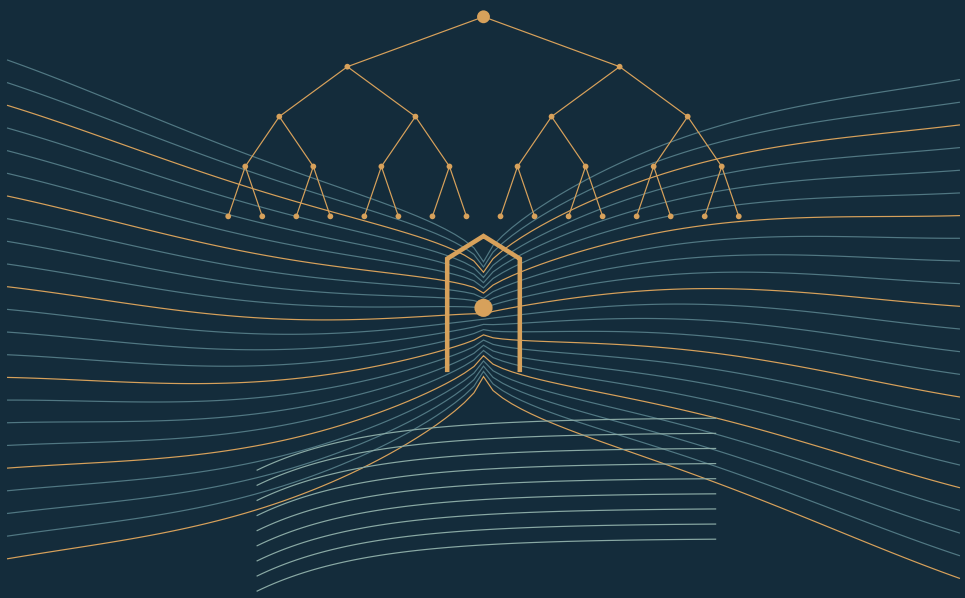
كتاب

SIBUNA

رياضيات حماية الويب وهندستها



العمل · الثقة · الحالة المحدودة

Vikrant Rathore، بمساعدة Ronak Rathore وKanak Rathore

الإصدار 1.0.0-beta.3 · Nov 13, 2026

المحتويات

المقدمة 6

كيف نقرأ هذا الكتاب 8

ثلاثة مسارات للقراءة 8

الاصطلاحات 9

الجزء الأول: تكلفة الطلب 10

القبول قرار اقتصادي 10

ما يثبته إثبات العمل 11

البحث عن التجزئة متغيرًا عشوائيًا 11

الجلسات توزع تكلفة العمل 12

الحالة المحدودة ميزانية ثانية 12

اللامتغيرات التي نحافظ عليها فيما يلي 13

الجزء الثاني: الأعمال السابقة وفضاء التصميم 14

من أين جاء Sibuna 14

ثلاث أفكار في ملف تنفيذي واحد 16

مقارنة الميزات 17

الجزء الثالث: أسس من أبحاث التشفير 23

Hashcash: رياضيات المستوى الأول 23

إثبات العمل المتسلسل: المستوى الثاني 24

المصادقة المتماثلة: التحديات والجلسات 28

الجزء الرابع: بروتوكول Sibuna 31

دورة التحدي 31

سجل التحدي عديم الحالة 33

ترتيب التحقق والاستخدام الواحد 34

ملف تعريف ارتباط الجلسة 35

الجزء الخامس: هندسة بلا تخصيص للذاكرة في Rust 38

حلقة الاتصال 38

حالة متزامنة بلا تخصيص 49

الجزء السادس: آلات الحالات، وأشجار البادئات، والجدار الناري الدلالي 59

مطابقة أنماط متعددة بمرور واحد 59

شجرة radix لعناوين IPv4 وIPv6 61

الجدار الناري الدلالي 62
محرك السياسة التصريحي 63
الجزء السابع: محرك إثبات العمل في المتصفح 66
مصدر واحد وهدفان 66
بروتوكول العامل 70
المعايرة 71
الصفحة الوسيطة 71
الجزء الثامن: التقييم التجريبي 73
المنهجية 73
أزمنة العمليات الأولية 78
مقارنة المنتجات الكاملة 81
زمن الاستجابة عند حمل معروض متساوٍ 83
Anubis وSibuna 85
التكلفة الإضافية لوحدة التحكم 90
ما لم يُقس 91
الجزء التاسع: التشغيل والنشر 94
الأوضاع والإعداد 94
لمحة عن وحدة التحكم 98
طوبولوجيات النشر 115
السياسة التصريحية 122
التخزين الدائم والعنقدة 123
التحزيم 127
الجزء العاشر: المرجع التفصيلي والتشخيص 132
نقاط النهاية 132
المقاييس 133
رموز الحالة 134
فهرس الأخطاء 135
مخطط السياسة 137
جداول التخزين 138
أهداف البناء 139
الجزء الحادي عشر: بطاقة مرجع سريع 141
البناء 141
التشغيل 141

وحدة التحكم (معاينة تُفعَّل اختياريًا) 142
الخيارات الأهم أولًا 142
نقاط النهاية تحت /__sibuna/ 143
رموز الحالة 143
الترويسات 144
هيكل ملف السياسة 144
التخزين (Edge) بلغة SQL 145
التشخيصات 145
أرقام جديرة بالحفظ 146
توجيه وكيل الدخول 146
حلول مختارة 148
الجزء الأول: التكلفة والاحتمال 148
الجزء الثاني: الأعمال السابقة 148
الجزء الثالث: التشفير 148
الجزء الرابع: البروتوكول 149
الجزء الخامس: الحالة والترتيب 149
الجزء السادس: الخوارزميات 149
الجزء السابع: محرك المتصفح 150
الجزء الثامن: التقييم 150
الجزء التاسع: التشغيل 150
الجزء العاشر: المرجع 151
واصل البحث 151
مسرد المصطلحات 152
المراجع 155
الفهرس 156

© 2026، Vikrant Rathore والمساهمون في Sibuna

رياضيات حماية الويب وهندستها.

المقدمة

يصل الطلب ومعه طريقة، ومسار، وبعض الترويسات، ومطالبة بانتباه الخادم. ولا يثبت أي من هذه البايتات أن تلبيته تستحق العناء. يضع Sibuna قرارًا بين الطلب والخادم الأصلي: أن يقبله، أو يرفضه، أو يطلب من مرسله إنجاز عمل يمكن التحقق منه. يتتبع هذا الكتاب ذلك القرار من نموذجه الرياضي إلى البايتات في الذاكرة.

للنظام وضعان. يوفر *Gate* القبول بإثبات العمل وضوابط محلية لصد الإغراق. ويضيف *Shield* فحص التطبيق وقرارات السياسة. ويستطيع كل منهما استخدام تخزين دائم للسياسة والسمعة وسجل الحوادث، محليًا على عقدة واحدة أو مشتركًا بين عقد عنقود؛ ونسمي هذا النشر *Edge*. الجلسة الصالحة تكفي للقبول، لكنها لا تعفي الطلب من الفحص. ومشاركة الحالة الدائمة عبر عنقود توزعها على عقده، لكنها لا تحول عداد معدل محليًا إلى حصة شاملة.

ليس السؤال الهندسي المحوري أن تكون التجزئة سريعة، بل أن يبقى المسار الكامل محدودًا حين يختار عميل غير موثوق المدخلات. كم بايتًا يستطيع المحلل أن يفحص؟ كم من الذاكرة يستطيع التحدي أن يستهلك؟ كم اتصالًا يستطيع العميل أن يبقيه مفتوحًا؟ أي عمليات الكتابة يجوز إعادة محاولتها؟ ماذا يبقى متاحًا بعد اختفاء عقدة أو قاعدة البيانات؟ يجب أن تسمي كل إجابة لامتغيرًا والحدَّ الذي يقف عنده معًا.

وتخدم الرياضيات الغرض نفسه. التكلفة المتوقعة ليست ضمانًا لزمن الاستجابة. وحجة أخذ العينات لالتزام ثابت ليست برهانًا ضد كل مُبرهِن متكيف. يمتص الطابور دفعة، لكنه لا يعوض مستهلكًا أبطأ على الدوام. نشتق نماذج صغيرة، ونحل أمثلة يدويًا، ثم نسأل أين يحيد التنفيذ عنها.

كُتب هذا الكتاب ليُستخدم بثلاث طرق. إذا قرأناه بالترتيب، فالأجزاء من الأول إلى السابع مقرر دراسي: يبدأ كل فصل بما ينبغي أن نقدر عليه بعده، ويحل مثالًا يدويًا، ويُختم بتمرين ولامتغير نشرحه بكلماتنا؛ وحلول مختارة في آخر الكتاب. والأجزاء من الثامن إلى العاشر مرجع: قياسات مع مصدرها، وكل خيار وكل بنية نشر، وكل نقطة نهاية وحالة وخطأ وجدول. والجزء الحادي عشر بطاقة من صفحتين للاستخدام عند الطرفية. ويُختم الكتاب بمسرد للمصطلحات وقائمة بالمراجع.

يضع الجزء الثاني Sibuna إلى جانب Anubis وSafeLine وCloudflare WAF، ميزةً بميزة. ويعرض الجزء الثامن قياسات Sibuna نفسه، من الأدوات الأولية المفردة إلى تشغيلات المنتج الكامل، ويقارن Sibuna بـ Anubis منتجين كاملين تحت مولد الحمل نفسه. وكلاهما مكتوب بحيث يستطيع القارئ التحقق من الادعاءات: تسمي المقارنة مصادرها، ويسمي كل قياس ملف النتائج المودَع في المستودع الذي نُسخ منه.

ينبغي أن يعرف القارئ أساسيات البرمجة، واللوغاريتمات، والاحتمال الشرطي. نقدم Rust من خلال الملكية وتخطيط البيانات، لا من خلال جولة في اللغة. يمكن قراءة الفصول بالترتيب: يسوّغ نموذج التكلفة البروتوكول، ويحدد البروتوكول الحالة، وتحدد الحالة تصميم التزامن والتخزين. وتحول الفصول اللاحقة تلك اللامتغيرات إلى قياسات وإجراءات تشغيل.

هذا كتاب عن التنفيذ، وليس ادعاءً بأن الألغاز الحسابية تقضي على الحركة الآلية. يستطيع العملاء شراء قدرة حسابية، وقد يتشارك عدة عملاء عنوانًا واحدًا، وصياغة التطبيقات أغنى من أي كاشف محدود. والنتيجة المفيدة نظام يمكن فحص تكاليفه وحدوده واختبارها وتغييرها دون إخفائها وراء شعار.

طبعة لإصدار *Sibuna 1.0.0-beta.3*، نوفمبر *2026*

في مستودع Sibuna الشيفرة المصدرية والتمارين والقياسات القابلة لإعادة الإنتاج.

كيف نقرأ هذا الكتاب

نبدأ بسؤال واحد: ما أقل القرارات الآمنة تكلفة يستطيع الخادم اتخاذه الآن؟ ونبقي هذا السؤال حاضرًا معنا في فصول البروتوكول والتنفيذ. العملية قليلة التكلفة التي تقبل الطلب الخطأ عيب؛ والعملية الصحيحة ذات التكلفة غير المحدودة عيب من نوع آخر.

السؤال	أين تتبلور الإجابة
ماذا يشتري اللغز؟	الأجزاء من الأول إلى الثالث: التكلفة، والاحتمال، وحدود البرهان.
كيف يقارَن Sibuna بغيره؟	الجزء الثاني: نسب الأفكار وجدول الميزات؛ الجزء الثامن: القياسات.
ماذا تعني الجلسة؟	الجزء الرابع: الروابط، وترتيب التحقق، والانتهاء، وإعادة الاستخدام.
أين تذهب الذاكرة؟	الجزءان الخامس والسادس: المخازن المؤقتة، والخيوط، والجداول، والآلات، والنشر.
كيف يحل المتصفح اللغز؟	الجزء السابع: رسائل العامل، والذاكرة، والمعايرة.
كيف نعرف أنه يعمل؟	الجزء الثامن: ثلاث منظومات قياس وما تستبعده كل منها.
كيف نشغله؟	الجزء التاسع: الخيارات، وبنى النشر، والسياسة، والتخزين، والعنقدة، والتحزيم.
ما معنى ذلك الخطأ؟	الجزء العاشر: نقاط النهاية، والحالات، والأخطاء، والمخطط؛ الجزء الحادي عشر: البطاقة.

ثلاثة مسارات للقراءة

- تعلم التصميم. الأجزاء من الأول إلى الرابع بالترتيب، ثم التتبع المحلول في الجزء السادس، ثم الجزء السابع. حلّ التمارين، ولا تقارن بالحلول في آخر الكتاب إلا بعد ذلك. وعُد إلى الجزء الخامس عندما يصل التتبع إلى جدول مشترك أو مقطع مستعار.
- تشغيل نشر. الجزء التاسع، ثم الجزء الحادي عشر، مع إبقاء الجزء العاشر مفتوحًا للبحث. اقرأ "مقارنة الميزات" في الجزء الثاني قبل اختيار الوضع، و"مقارنة المنتجات الكاملة" في الجزء الثامن قبل اختيار حدود العمال والاتصالات.
- تغيير الشيفرة. الجزءان الخامس والسادس، ومراسي المصدر في كل فصل، وسجلات SID التي تستشهد بها. يشرح الجزء الثامن أي منظومة قياس ستكشف تراجعًا فيما غيّرناه؛ ويسرد الجزء الثاني القيود التي يجب أن يحافظ عليها أي تغيير.

الاصطلاحات

يعرض المثال المحلول الحالات الوسيطة، لا الإجابة وحدها. ويطلب منا التمرين تغيير افتراض واحد. تقترح التلميحات خطوة أولى؛ وتجعل الحلول المختارة في آخر الكتاب الاستدلال قابلًا للتحقق. وتميز المخططات مسار الطلب من العمل الجاري في الخلفية. ويسمي مرسى المصدر التنفيذ الذي نفحصه عندما يبدو أن النص والشيفرة يختلفان. والإطارات المعنونة "التنفيذ" تسمي دالة وملفها؛ والإطارات المعنونة "اشرح اللامتغير" تطلب منا أن نعيد شرح الفكرة. في المعادلات، b هي صعوبة Hashcash بالبتات، وn عمق العمل المتسلسل، وt عدد عمليات الفتح، وp احتمال النجاح في كل محاولة، وK عدد المحاولات، وT فاصل الإصدار لمحدد المعدل، وτ تسامحه مع الدفعات، وQ سعة الطابور. الرمز محلي في قسمه ما لم يُذكر غير ذلك. والنانوثواني والطلبات في الثانية في الجزء الثامن قياسات من تشغيل مسمى. والأرقام في الأمثلة المحلولة مدخلات مختارة، لا ادعاءات قياس أداء. وتذكر النماذج الإحصائية افتراضاتها قبل استخلاص النتائج.

الجزء الأول

تكلفة الطلب

قبل اختيار خوارزمية، نحدد المورد الذي يُراد لها أن تحميه. يبني هذا الفصل النماذج الصغيرة التي يُقاس عليها كل قرار تصميم لاحق: عمل الخادم الأصلي، وعمل اللغز، وتوزيع تكلفة العمل على الجلسة، والذاكرة التي يستطيع عميل غير موثوق أن يجبر الخادم على الاحتفاظ بها.

الأهداف بنهاية هذا الفصل، ينبغي أن نستطيع كتابة المتباينة التي تقرر هل تسترد البوابة تكلفتها، وبيان ما يثبته إثبات العمل وما لا يثبته، واشتقاق القيمة المتوقعة للبحث عن التجزئة وذيله، وشرح كيف توزع الجلسات تكلفة العمل، وسرد اللامتغيرات الخمسة التي يحافظ عليها بقية الكتاب.

القبول قرار اقتصادي

لنفترض أن الخادم الأصلي يجري بحثًا في قاعدة البيانات وينشئ صفحة لكل طلب مقبول. ليكن c_o ذلك العمل، و c_g عمل البوابة لكل طلب، وr معدل الوصول. من دون بوابة، يجب أن يتحمل الخادم الأصلي rc_o وحدة عمل في الثانية. والبوابة التي تقبل نسبة a من الطلبات تغير ذلك الطلب على العمل إلى rac_o، بينما تنفق هي نفسها rc_g. ولا تكون البوابة مفيدة إلا عندما يزيد عمل الخادم الأصلي الذي توفره على تكلفتها هي وعلى التكلفة التي تفرضها على الزوار الشرعيين.

$$rc_g + rac_o < rc_o \quad \Leftrightarrow \quad c_g < (1-a)c_o.$$

هذا نموذج للسعة، وليس قياسًا. فهو يُغفل عرض النطاق، وحالة الاتصالات، والتخزين، وتأخير العميل. وقيمته أن كل حد مفقود ظاهر. قد تجعل إصابة ذاكرة التخزين المؤقت c_o صغيرة، وقد يجعلها استعلام مكلف كبيرة. ولا توجد نسبة عامة بين تكلفة صاحب الطلب وتكلفة الخادم.

مثال محلول: متى تسترد البوابة تكلفتها لنختر $c_o = 100$ وحدة عمل و$c_g = 1$. إذا رُفض نصف الطلبات، ينفق النظام ذو البوابة $100 \times 0.5 + 1 = 51$ وحدة لكل وصول بدل 100. وإذا لم يُرفض إلا طلب واحد من كل ألف، فإنه ينفق 100.9: تكلف البوابة في هذا النموذج أكثر مما توفر. هذه الأرقام المختارة ليست أزمنة مقيسة. إنها تبين لماذا يجب ذكر عبء العمل في كل ادعاء عن الأداء.

ما يثبته إثبات العمل

يثبت لغز العميل أن أحدًا ما وجد مدخلًا يستوفي قاعدة تحقق علنية. ولا يثبت أن صاحبه إنسان، ولا يثبت هويته، ولا حسن نيته. يستطيع صاحب الطلب استئجار قدرة حسابية، أو إعادة استخدام جلسة صالحة خلال عمرها، أو توزيع العمل على عدة أجهزة. يغير اللغز تكلفة القبول؛ أما ما يجوز للطلبات المقبولة أن تفعله فتقرره السياسة والفحص. الحدود المبنية على العناوين مكملة لذلك. فهي تقيد عنوانًا واحدًا، لكن العناوين المشتركة قد تمثل أناسًا كثيرين، وقد يستخدم صاحب طلب واحد عناوين كثيرة. لا اللغز ولا العنوان مصدر موثوق للهوية. لذلك يبقي Sibuna القبول والفحص والسمعة قرارات منفصلة.

البحث عن التجزئة متغيرًا عشوائيًا

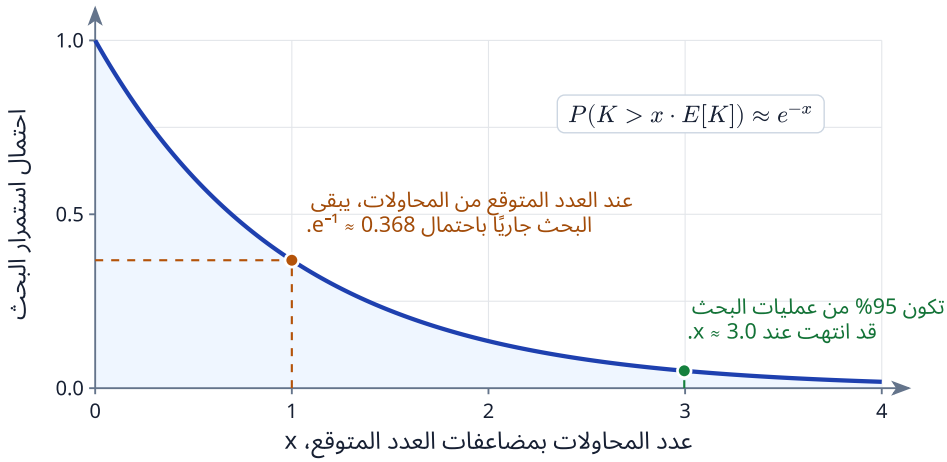
في خلاصة مثالية من 256 بت، يعطي اشتراط b بتًا صفريًا بادئًا احتمال نجاح $p = 2^{-b}$ لكل محاولة مستقلة. ليكن K عدد المحاولات حتى أول نجاح، بما فيها تلك المحاولة. عندئذ

$$P(K = k) = (1-p)^{k-1}p, \quad P(K > k) = (1-p)^k, \quad E[K] = 1/p = 2^b.$$

تنتج القيمة المتوقعة من مجموع الذيل:

$$E[K] = \sum_{k=0}^{\infty} P(K > k) = \sum_{k=0}^{\infty} (1-p)^k = 1/p.$$

تضاعف كل خطوة في الصعوبة العدد المتوقع للمحاولات. لكنها لا تعد بأن كل لغز سيستغرق ضعف الوقت. فبعض الألغاز ينجح من المحاولة الأولى، وبعضها يستغرق أطول من المتوسط بكثير. والعميل الذي يقيس حلًا واحدًا فقط قد قاس في الغالب هذه العشوائية.



الشكل 1 احتمال البقاء في البحث عن التجزئة: احتمال أن يظل البحث جاريًا بعد عدد من المحاولات يبلغ x ضعف العدد المتوقع، وفق تقريب العمل الكبير $P(K > x/p) \approx e^{-x}$.

مثال محلول: القيمة المتوقعة ليست موعدًا نهائيًا عند $b = 16$، العمل المتوقع $65{,}536$ تجزئة. واحتمال استمرار البحث بعد هذا العدد من المحاولات يساوي تقريبًا $e^{-1} = 0.368$. وعدد المحاولات عند المئين 95 هو $\lceil \ln(0.05) / \ln(1 - 2^{-16}) \rceil \approx 196327$، أي نحو ثلاثة أضعاف المتوسط. ينبغي لواجهة المستخدم أن تتحمل هذا التفاوت بدل أن تعلن الفشل عند وقت الإكمال المتوقع.

الجلسات توزع تكلفة العمل

لنفترض أن الجلسة تسمح بـ m طلبًا قبل انتهائها، وأن اللغز يكلف c_p، والتحقق يكلف c_v، وفحص الجلسة يكلف c_s. إذا أهملنا المحاولات غير الناجحة، فتكلفة البوابة الموزعة لكل طلب هي $c_v/m + c_s$، بينما تكلفة اللغز على العميل c_p/m. وإطالة عمر الجلسة تفيد الناس والعملاء الآليين على حد سواء. واختيار هذا العمر قرار سياسة، لا تحسين تشفيري.

تمرين 1.1. يحصل عميل على 100 طلب لكل جلسة، ويحل لغزًا يكلف مليون محاولة. ما العمل الموزع لكل طلب؟ وما الذي يتغير إذا شارك الجلسة مع عملية ثانية؟
تلميح: ميّز بين نموذج المحاسبة وما يرتبط به الرمز فعلًا.

الحالة المحدودة ميزانية ثانية

نقل العمل خارج مسار الطلب لا يجعله يختفي. لنفترض أن الحوادث تصل بمعدل λ، وأن خيط التخزين يحفظها بمعدل متوسط μ، وأن الطابور يتسع لـ Q سجلًا. عندما يكون $\lambda > \mu$، يعطي تقريب السوائل

$$t_{\text{fill}} \approx \frac{Q}{\lambda - \mu}.$$

الطابور الأكبر يشتري وقتًا، لكنه لا يصنع إنتاجية. يوزع التجميع في دفعات تكلفة المعاملات، لكن السجلات المحتجزة لا تزال تحتاج إلى ذاكرة وإلى معالجة في النهاية. تحت الحمل الزائد، يُسقط Sibuna سجلات الحوادث الجديدة ويحصيها. ويستمر في اتخاذ قرارات الطلبات من الحالة التي في الذاكرة. ويجب أن يراقب المشغل الأدلة المفقودة إلى جانب معدل نجاح HTTP.

تمرين 1.2. يتسع طابور لـ 512 سجلًا. معدل الوصول $1{,}000$/s ومعدل التصريف 600/s. كم من الوقت يستطيع طابور فارغ في البداية أن يمتص الفائض؟ ولماذا ليست الإجابة إلا تقريبًا؟
تلميح: استخدم الفرق بين المعدلين، ثم انظر في الدفعات المفاجئة وفي تثبيت المعاملات على دفعات.

اللامتغيرات التي نحافظ عليها فيما يلي

1. لا تستطيع الطلبات غير الموثوقة إنشاء حالة غير محدودة على الخادم.
2. رمز القبول الصالح لا يتخطى رفضًا صادرًا عن فحص التطبيق.
3. يبقى المخزن المؤقت حيًّا ما دامت أي مقطع (slice) مستعار منه باقية.
4. لا تستطيع إعادة محاولة الحفظ الدائم أن تكرر حادثًا أو أثره في السمعة.
5. لقطة السياسة المنشورة غير قابلة للتغيير حتى يتخلى عنها آخر قارئ لها.

يشتق بقية الكتاب الآليات التي تجعل هذه العبارات صحيحة، والاختبارات التي تكشف أي خرق لها. ولا يكون المسار السريع مفيدًا إلا ما دامت هذه العبارات صحيحة.

اشرح اللامتغير. صُغ متباينة البوابة $c_g < (1 - a)c_o$ بالكلمات، ثم اشرح لماذا لا يستطيع زمن استجابة التحقق المقيس وحده أن يخبر المشغل هل تستحق البوابة التشغيل.

الجزء الثاني

الأعمال السابقة وفضاء التصميم

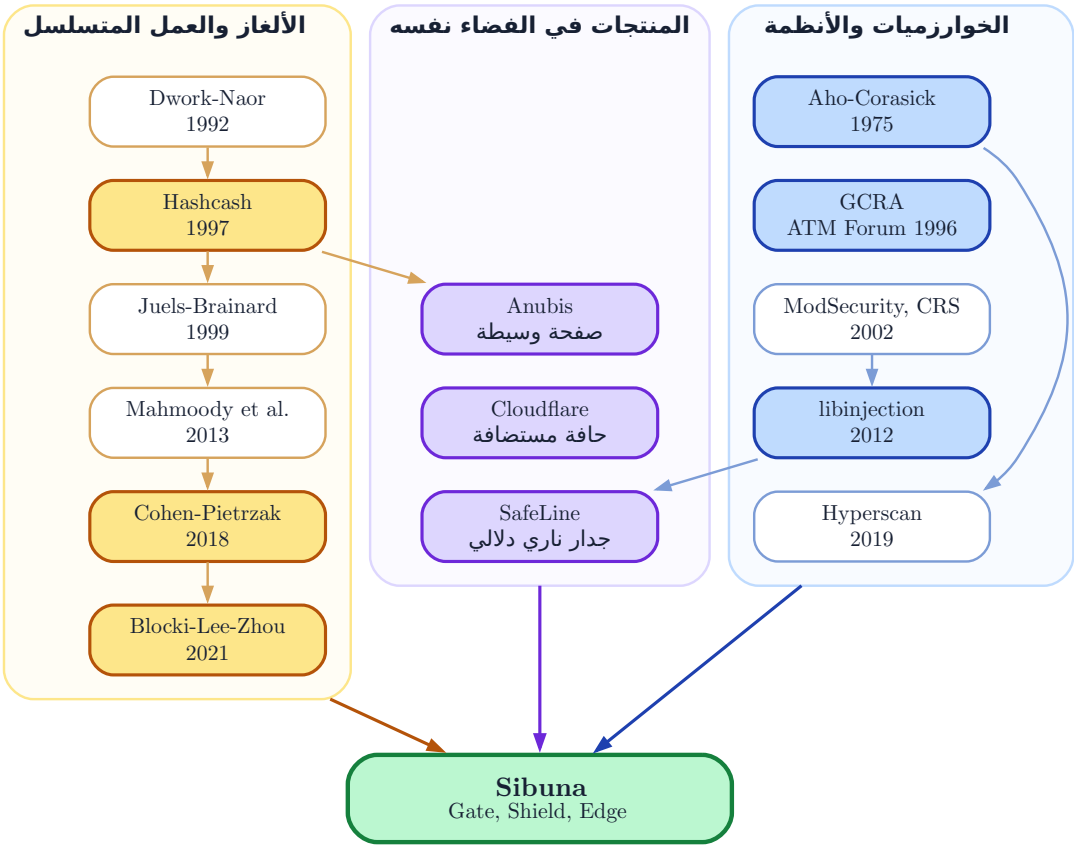
نضع *Sibuna* بين الأنظمة التي تعلم منها: ألغاز العملاء والعمل المتسلسل من أدبيات التشفير، وجدران حماية التطبيقات القائمة على التوقيعات والدلالة، والصفحات الوسيطة لإثبات العمل، وشبكات الحافة المستضافة. ثم نقارن، ميزةً ميزة، الأدوات الأربع التي يرجح أن يوازن المشغل بينها، ونسجل الحقائق التي تحققنا منها.

من أين جاء Sibuna

الأهداف بنهاية هذا الفصل، ينبغي أن نتمكن من تتبع نسب ألغاز العملاء من عام 1992 إلى بناءات العمل المتسلسل عام 2018، وتسمية الأفكار المعمارية الثلاث التي يجمعها Sibuna، ووضع Sibuna وAnubis وSafeLine وCloudflare WAF في فضاء تصميم واحد، وبيان ادعاءات المقارنة التي تحققنا منها وتلك التي قرأناها من الوثائق فقط.

ألغاز العملاء والعمل المتسلسل

اقترح بحث Dwork وNaor "Pricing via Processing" (CRYPTO 1992) أن تطلب الخدمة حسابًا متوسط الصعوبة سهل الفحص قبل أن تؤدي عملًا لصاحب الطلب. وجعل Hashcash الذي وضعه Back (1997، ونُشر وصفه مكتوبًا في 2002) اللغز عكسًا جزئيًا لتجزئة تُحدَّد صعوبته بعدد صحيح واحد، وطبّق Juels وBrainard ألغاز العملاء (NDSS 1999) على إغراق الاتصالات، حيث يجب ألا يتذكر الخادم شيئًا عن لغز لم يُحل. وتشترك الثلاثة في الشكل الذي يحتفظ به Sibuna: مسألة يختارها الخادم، وحل قابل لضبط تكلفته، وتحقق أرخص من الحل بمراتب عددية. البحث في التجزئة قابل للتوازي بالكامل، لذا ينتهي صاحب الطلب الذي يملك نوى أكثر أسرع. عرّف Mahmoody وMorano وVadhan (CRYPTO 2013) إثباتات العمل المتسلسل، التي يجب فيها على المُبرهِن إجراء سلسلة طويلة من الخطوات المترابطة مهما كان عدد نواه، وقدم Cohen وPietrzak (EUROCRYPT 2018) البناء البسيط القائم على رسم التجزئة، الذي لا يحتاج متحققه إلا إلى عمل $O(t \log N)$. وأثبت Blocki وLee وZhou (ITC 2021) سلامة البناء نفسه ضد المُبرهِنين الكموميين في نموذج الأوراكل العشوائي الكمومي. يشتق الجزء الثالث المستوى الثاني في Sibuna من هذا الخط البحثي؛ ويسجل SID 0006 الافتراضات.



الشكل 2 نسب الأفكار التي يجمعها Sibuna. العمود الأيسر هو تشفير ألغاز العملاء والعمل المتسلسل، والعمود الأيمن الخوارزميات والأنظمة التي يبني عليها Sibuna، والعمود الأوسط المنتجات في الفضاء نفسه؛ ويجري الزمن نحو الأسفل. وتدل الأسهم الرفيعة على انحدار مباشر، والمربعات المظلّلة هي الأفكار التي يأخذها Sibuna مباشرة. وقد استند تصميم تخزين سابق إلى Multi-Paxos (Lamport 1998؛ SID 0005)؛ أما الآن فيخزن Sibuna حالته في SQLite أو PostgreSQL (SID 0011).

جدران حماية التطبيقات

طابق الجيل الأول من جدران حماية تطبيقات الويب بايتات الطلب مع تعبيرات نمطية. وجعل ModSecurity (2002) ومجموعة OWASP Core Rule Set هذا النهج مفتوحًا وواسع الانتشار، وجعلا تكاليفه مألوفة أيضًا: مئات الأنماط ذات التراجع لكل طلب، ودرجات شذوذ تُضبط يدويًا، وذيل طويل من الإنذارات الكاذبة على المدخلات العادية. ثم استبدل libinjection (Hanson، 2012) بقوائم الأنماط مقسِّمًا (tokenizer) يسأل هل تُحلَّل السلسلة بوصفها SQL، فخفض التكلفة والإنذارات الكاذبة معًا. ونقل SafeLine من Chaitin هذه الفكرة الدلالية إلى منتج كامل، وبيّن Hyperscan (Wang وآخرون، NSDI 2019) إلى أي حد يمكن دفع المطابقة الحرفية المتجهة عندما تكون المدخلات طويلة. ويشرح الجزء السادس لماذا اختار Sibuna آلة كثيفة مع مقسِّمات بنيوية صغيرة، بدل التعبيرات النمطية أو محرك SIMD، للحقول القصيرة التي يفحصها.

الصفحات الوسيطة وشبكات الحافة

أصبحت الصفحة الوسيطة لإثبات العمل أمام مواقع الويب شائعة في 2025، حين جعلت حركة أدوات كشط المحتوى التي تغذي تدريب النماذج اللغوية تشغيل الخوادم الأصلية العادية باهظ التكلفة. وAnubis (Techaro) هو التنفيذ المفتوح الأشهر: وكيل عكسي مكتوب بلغة Go يقدم لغز JavaScript، ويوقع رمز جلسة، ويمرر بقية الحركة كما هي. ويحل وضع Gate في Sibuna المشكلة نفسها بتحدٍّ عديم الحالة، ورمز متماثل، ومستوى عمل ثانٍ؛ وتسجل المقارنة أدناه مواضع الاختلاف بينهما.

في الطرف الآخر من المقياس، تضع شبكات الحافة المستضافة مثل Cloudflare الفحص، وتحديد المعدل، وتقييم الروبوتات في مئات نقاط الحضور مع مستوى تحكم عالمي. يستعير نشر Edge في Sibuna هذا الشكل (مستوى مشترك للسياسة والسمعة يغذي لقطات محلية غير قابلة للتغيير) مع بقائه مستضافًا ذاتيًا، ولهذا تكون طبقة التخزين قاعدة بيانات يشغلها المشغل (ملف SQLite على عقدة واحدة، أو قاعدة بيانات PostgreSQL واحدة يتشاركها عنقود) يقرؤها خيط تخزين خارج مسار الطلب، بدل استدعاء خدمة.

ثلاث أفكار في ملف تنفيذي واحد

يجمع Sibuna تحدي عمل في المتصفح للقبول، وفحصًا محدودًا لحمولة التطبيق، ومستوى تحكم مشتركًا ينشر لقطات سياسة محلية. وGate وShield هما وضعا المنتج؛ أما التخزين والنشر الموزع فخياران لأي منهما.

الخاصية	Gate	Shield
القبول	تحقق أصلي من Hashcash أو العمل المتسلسل	نفسه
الجلسة	BLAKE3 ذو مفتاح؛ Ed25519 اختياري	نفسه
الضوابط المحلية	قواعد، ونطاقات CIDR، وGCRA، وحظر	نفسه
الفحص	معطل	توقيعات محدودة ومقسِّمات بنيوية
الحالة	سياسة وسمعة دائمتان اختياريتان، مشتركتان عبر العنقود	وأدلة الفحص الجنائية أيضًا

يُبنى برنامج الحل في المتصفح من مصادر الإثبات نفسها التي يُبنى منها المتحقق الأصلي. وتتجنب المطابقة الحرفية الكثيفة التراجع، وتبقي عمل المسح خطيًا في طول المدخل. وتعزل لقطات السياسة غير القابلة للتغيير معالجة الطلبات عن SQL وعن توفر قاعدة البيانات. ولا يثبت أي من هذه الخيارات أمثلية خوارزمية مطلقة: فالترويسات القصيرة، والأجسام الطويلة، وتنوع الأنماط، وعدد النوى، وحجم الذاكرة المخبئية كلها تغير الموازنات.

> **قيود التصميم المعتمدة**

1. لا تخصيص على الكومة في مسار الطلب؛ ولكل جدول سعة ثابتة وسلوك محدد عند الامتلاء.
2. إصدار التحدي لا يكتب شيئًا؛ ولا يشغل الذاكرة إلا حل دُفع ثمنه.
3. الجلسة تمنح القبول؛ لكنها لا تعفي الطلب أبدًا من الفحص ولا من رفض صريح.
4. كل رقم مُعلَن إما مقيس بأداة قياس مودَعة في المستودع، وإما موسوم بأنه نموذج.
5. تستشهد الخيارات التشفيرية بمنشور وبرهان، وتذكر موقفها من الحوسبة الكمومية.

مقارنة الميزات

تقارن الجداول الأربعة أدناه Sibuna بالأدوات الثلاث التي يكثر ذكرها معه. وقد قُرئت الحقائق عن المنتجات الأخرى من وثائقها العامة وملفات إصداراتها في سبتمبر 2026: Anubis 1.27.0 (رخصة MIT، بلغة Go)، وSafeLine community edition 9.x (شفرة إدارة بترخيص GPL-3.0 حول صور كاشف مغلقة، مع Docker Compose)، ووثائق المطورين الخاصة بـ Cloudflare WAF. وشُغِّل Anubis أيضًا على مضيف قياس الأداء؛ ولم يُشغَّل SafeLine ولا Cloudflare (يشرح الجزء الثامن السبب). وتعني الشرطة أن المنتج لا يقدم الميزة؛ ويعني اسم الخطة أن الميزة موجودة لكن في تلك الخطة فقط. ويذكر كل قياس لـ Sibuna ملف النتائج المودَع في المستودع الذي أُخذ منه. وقد روجع صف بصمة TLS في أكتوبر 2026 مقابل سجل تغييرات Anubis (1.21.0 و1.26.0)، وملاحظات إصدار SafeLine (8.10.0 و9.2.0)، وصفحة Cloudflare "JA3/JA4 fingerprint".

الجدول 3 مقارنة الميزات، القسم 1: النشر

الميزة	Sibuna	Anubis 1.27	SafeLine CE 9	Cloudflare WAF
طريقة التشغيل	ملف تنفيذي واحد؛ وكيل عكسي أو forward auth؛ ملف SQLite اختياري أو قاعدة بيانات PostgreSQL مشتركة	ملف تنفيذي واحد بـ Go؛ وكيل عكسي أو forward auth	سبع حاويات Docker (tengine، وdetector، وmgt، وluigi، وfvm، وchaos، وPostgreSQL)	شبكة مستضافة؛ يشير DNS إلى Cloudflare
اللغة وبيئة التشغيل	Rust، بلا جامع نفايات؛ وبلا حزم خارجية عند تعطيل التخزين	Go، مع جمع النفايات	وكيل وكاشف بـ C/C++، وإدارة بـ Go، وPostgreSQL	برمجيات حافة مملوكة
الرخصة	LGPL-3.0-or-later	MIT	شفرة إدارة بترخيص	خدمة مملوكة

الميزة	Sibuna	Anubis 1.27	SafeLine CE 9	Cloudflare WAF
			GPL-3.0؛ صور كاشف مغلقة	
الأثر على المضيف	ملف تنفيذي بحجم 35.7 MiB؛ و5.9 MiB من RSS في الخمول (`benchmarks/results/comparison-latest.json`)؛ و16.0 MiB على الأكثر ذروةً لـ RSS تحت الحمل (`benchmarks/results/product-latest.md`)	ملف تنفيذي بحجم 38.1 MiB؛ و21 MiB من RSS في الخمول؛ بيئة اختبار Linux، ووحدة التحكم معطلة	1 نواة CPU، و1 GB من RAM، و5 GB من القرص حدًا أدنى، وDocker 20.10+	لا شيء محليًا

الجدول 4 مقارنة الميزات، القسم 2: القبول والجلسات

الميزة	Sibuna	Anubis 1.27	SafeLine CE 9	Cloudflare WAF
القبول بإثبات العمل	Hashcash (على مستوى البتات) والعمل المتسلسل لـ Cohen-Pietrzak، يُختار لكل قاعدة؛ WebAssembly مع بديل JavaScript	Hashcash بـ SHA-256 بأنصاف بايت ست عشرية (`fast`، و`slow`)؛ وتحديا `metarefresh` و `preact` بلا عمل	تحدي JavaScript لمكافحة الروبوتات وCAPTCHA؛ غير مشروط بالعمل	Managed challenge، وJS challenge، وTurnstile؛ يصدر Bot Fight Mode تحديًا حسابيًا
حالة التحدي على الخادم	لا شيء حتى الحل؛ الوسوم المحلولة في جدول Robin Hood ثابت	خلفية التخزين: الذاكرة، أو bbolt، أو Valkey، أو S3	تديرها المجموعة البرمجية	تديرها Cloudflare

الميزة	Sibuna	Anubis 1.27	SafeLine CE 9	Cloudflare WAF
رمز الجلسة	وسم BLAKE3 ذو مفتاح، وملف تعريف ارتباط من 75 محرفًا مربوط بالعمل المدفوع؛ Ed25519 اختياري	JSON Web Token موقع بـ Ed25519؛ HS512 اختياري	ملف تعريف ارتباط تصدره المجموعة البرمجية	ملف تعريف الارتباط `cf_clearance`
موقف الرمز الافتراضي من الحوسبة الكمومية	متماثل: لا ينطبق إلا تسارع Grover التربيعي	Ed25519: تكسره خوارزمية Shor	غير موثق	غير موثق
لغة السياسة	قواعد JSON: المسار، وUser-Agent، والترويسات، ونطاقات CIDR لـ IPv4/IPv6، وبصمة JA4؛ ALLOW وDENY وCHALLENGE وWEIGH مع عتبات؛ قواعد ديناميكية من قاعدة البيانات	روبوتات YAML/ JSON: تعبيرات نمطية على UA والمسار والترويسات؛ CIDR؛ تعبيرات CEL؛ WEIGH مع عتبات	قواعد مخصصة تُعرّف في وحدة التحكم، وACLs، ومجموعات IP	تعبيرات لغة القواعد؛ *5 / 20 / 100 / 1000* قاعدة مخصصة حسب الخطة
بصمة TLS (JA4)	تطابق القواعد بصمة JA4 مطابقة تامة أو بالبادئة؛ يحسبها وكيل دخول موثوق ويمررها في ترويسة (`--ja4-header`)؛ ولا يحسبها Sibuna	غير موثق؛ يمكن للسياسات أن تشير إلى بصمة HTTP المسماة JA4H، التي يحسبها Anubis ويضيفها في الترويسة `X-Http-Fingerprint-JA4H`	تُحسب عند إنهاء TLS الخاص به؛ وتظهر في تفاصيل الهجوم؛ وتطابقها القواعد المخصصة (يساوي، لا يساوي)؛ وترفض قاعدة مدمجة البصمات المدرجة في معلوماته عن	بصمات JA3 وJA4 في قواعد WAF المخصصة والتحليلات مع *Bot Management (Enterprise)*

الميزة	Sibuna	Anubis 1.27	SafeLine CE 9	Cloudflare WAF
			بصمات JA4 الخبيثة	

الجدول 5 مقارنة الميزات، القسم 3: الفحص والحالة

الميزة	Sibuna	Anubis 1.27	SafeLine CE 9	Cloudflare WAF
فحص SQLi وXSS وRCE والاجتياز	Shield: آلة موسومة مع مقسِّمات بنيوية، وأول 8 KB من الجسم	-	محرك تحليل دلالي على فئات OWASP، وأيضًا SSRF وXXE وCRLF	مجموعات قواعد مُدارة *(Pro* وما فوق)؛ OWASP Core Ruleset؛ درجة الهجوم *(Business* وما فوق)
تحديد المعدل	GCRA لكل عميل، محلي للعقدة	- (يُعرض حمل النظام لقواعد CEL)	لكل IP ومسار وجلسة	*1 / 2 / 5 / 100* قاعدة حسب الخطة؛ مفاتيح IP فقط دون Enterprise
الحظر والسمعة	حظر المصيدة؛ شجرة سمعة تُحمّل من جدول السمعة المشترك عبر العنقود	DNSBL؛ ASN وGeoIP عبر Thoth *(مدفوع، مغلق)*	مجموعات IP وقوائم سوداء؛ معلومات التهديدات *(Pro)*	قوائم IP؛ قوائم مُدارة ودرجة الروبوت *(Enterprise)*
الأدلة الجنائية	SQLite أو PostgreSQL: بحث نصي كامل في الحوادث، وتجميع متجهي للحملات	مقاييس Prometheus فقط	سجل هجمات في PostgreSQL مع وحدة تحكم	Security Events؛ بعينات في خطة Free
العقد المتعددة	قاعدة بيانات PostgreSQL واحدة تتشاركها كل العقد للسياسة والسمعة؛ بذرة	مفتاح توقيع مشترك؛ مخزن Valkey مشترك	مجموعة واحدة لكل مضيف	Anycast عالمي

الميزة	Sibuna	Anubis 1.27	SafeLine CE 9	Cloudflare WAF
	مشتركة؛ تحقق من التحدي مثبت على العقدة المُصدرة			

الجدول 6 مقارنة الميزات، القسم 4: التشغيل

الميزة	Sibuna	Anubis 1.27	SafeLine CE 9	Cloudflare WAF
المقاييس	Prometheus على `/__sibuna/metrics`	Prometheus على منفذ منفصل	لوحات وحدة التحكم	لوحة معلومات وتحليلات
إنهاء TLS	- (وكيل الدخول)	- (وكيل الدخول)	**نعم**	**نعم**
وحدة تحكم الإدارة	تُفعَّل اختياريًا؛ SID 0007 بحالة Committed	-	**نعم**	**نعم**
مقيس على مضيف قياس الأداء	**نعم** (الجزء الثامن)	**نعم**	- (Docker فقط)	- (مستضاف)

تحمل ثلاثة فروق معظم الوزن في قرار الاختيار.

- أين يذهب العمل. يجعل Anubis وSibuna Gate العميل يدفع قبل أن يفعل الخادم الأصلي أي شيء. يفحص SafeLine وCloudflare أولًا ويتحديان انتقائيًا؛ وموقفهما الافتراضي هو القبول ثم التصفية. يفعل Sibuna Shield الأمرين بهذا الترتيب: قد يرفض الفحص طلبًا يحمل جلسة صالحة بالفعل.
- ما يتذكره الخادم. يحفظ Anubis حالة التحدي في مخزن (الذاكرة، أو bbolt، أو Valkey، أو S3) ويوقع JSON Web Token بـ Ed25519؛ ويصدر Sibuna سجلًا ذاتي المصادقة ولا يتذكر إلا الوسوم المحلولة في جدول ثابت. يحفظ SafeLine سجلات الهجمات في PostgreSQL؛ ويحفظ Cloudflare كل شيء، مع رؤية بالعينات في خطة Free.
- ما يجب على المشغل تشغيله. Sibuna ملف تنفيذي واحد مع ملف SQLite اختياري، أو قاعدة بيانات PostgreSQL يتشاركها عنقود؛ وAnubis ملف تنفيذي واحد بلغة Go؛ وSafeLine سبع حاويات خلف برنامج Docker الخدمي على مضيف Linux بنواة واحدة على الأقل، وغيغابايت واحد من الذاكرة، وخمسة غيغابايتات من القرص؛ وCloudflare تغيير في DNS واشتراك.

> **حدود المنتج الحالية** يترك Sibuna إنهاء TLS الوارد لوكيل النشر، ولا يقيّم الروبوتات بنموذج مدرَّب ولا ينشر توقيعات مدفوعة. وهو يطابق بصمة JA4 التي يحسبها وكيل دخول موثوق؛ ولا يستطيع حسابها بنفسه

(SID 0014، الجزء التاسع). وتوفر وحدة التحكم الاختيارية لوحات معلومات تتطلب المصادقة، وإثراءً ببيانات البلدان، وتحرير السياسات، والتحقيق، وإدارة العنقود. ويجري البحث عن البلد وعمل التخزين خارج تصنيف الطلبات. حالة SID 0007 هي Committed، وتجتاز وحدة التحكم مسارات عملها في المتصفح (SID 0010). أما معيار الاجتياز الرسمي لعزل الأداء، مع ثماني لوحات معلومات نشطة تحت الحمل، فلم يُقس بعد؛ ويسجل الجزء الثامن ما قيس.

تمرين 2.1. اشرح لماذا تقلل ترجمة مصدر متحقق واحد إلى شفرة أصلية وإلى WebAssembly من انحراف البروتوكول. ما الخصائص التي ما زالت تتطلب اختبارًا مستقلًا أو مراجعة تشفيرية؟

تمرين 2.2. بالاعتماد على جداول المقارنة وحدها، اكتب أصغر نشر يمنح موقعًا ثابتًا (a) بوابة إثبات عمل، و(b) فحصًا لحقن SQL، و(c) حظرًا على مستوى العنقود. ما الصفوف التي فرضت كل اختيار؟
تلميح: ابدأ بصفّي "طريقة التشغيل" و"العقد المتعددة".

اشرح اللامتغير. صف كيف يقسم Gate وShield والتخزين المشترك الاختياري مهام القبول والفحص وتوزيع السياسة. ثم سمِّ صفًا واحدًا من جداول المقارنة نود التحقق منه مجددًا قبل نقله إلى عميل، وبيّن كيف نتحقق منه.

الجزء الثالث

أسس من أبحاث التشفير

نشتق نماذج التكلفة لمستويَي إثبات العمل في *Sibuna*، ونعرض إثبات العمل المتسلسل لـ *Cohen-Pietrzak* مع حجته الأمنية، ونبرر مصادقة التحديات والجلسات بتجزئة ذات مفتاح. يسجل *SID 0006* الافتراضات والمراجع.

Hashcash: رياضيات المستوى الأول

الأهداف بنهاية هذا الفصل، ينبغي أن نتمكن من تحديد العمل المتوقع والتباين واحتمال الذيل للغز Hashcash على مستوى البتات، وأن نشرح لماذا تحد تكلفة تحقق الخادم من أفضلية المهاجم، وأن نصف الحدين اللذين يدفعان إلى مستوى ثانٍ.

التعريف والتكلفة

لغز المستوى الأول في Sibuna لسلسلة التحدي C والصعوبة b بت هو: أوجد $N \in \mathbb{N}$ بحيث تبدأ SHA-256 $(C \,\|\, ":" \,\|\, \mathrm{dec}\,(N))$ بـ b بتات صفرية. كتابة القيمة المؤقتة بالنظام العشري مقصودة؛ فهي تبقي تنسيق النقل متطابقًا بين برنامج الحل بـ WebAssembly، وبديل JavaScript، والمتحقق الأصلي، من دون خطوة ترميز ثنائي.

وحدة الصعوبة هي محاولة تجزئة متوقعة، وليست استدعاءً لدالة الضغط. قد يشغل المدخل عدة كتل SHA-256. وقد تغير الحوسبة المسبقة للبادئة العمل لكل قيمة مؤقتة. تظل قاعدة النقل هي شرط الملخص؛ ويجب أن تتفق التنفيذات على المسألة وترميز القيمة المؤقتة، لا على تكلفة CPU تقديرية.

للملخصات المثالية المستقلة، يتبع عدد المحاولات X توزيعًا هندسيًا مع $p = 2^{-b}$. وسيطه $\lceil \ln(1/2) / \ln(1-p) \rceil$، ويقارب $2^b \ln 2$ عندما تكون p صغيرة، وتباينه $(1-p)/p^2$. يفحص المتحقق ملخصًا مرشحًا واحدًا بدل إجراء البحث. أما التحليل، ومصادقة التحدي، وفحص البصمة، وتسجيل الحل في مجموعة الحلول المُنفَقة، والاستجابة، فعمل إضافي. ولا يمكن عكس توقيت عملية أولية لاستنتاج قدرة الخادم على تحمل الإغراق.

حدان للبحث

الحد الأول هو التباين: البحث الطويل لا يعني بالضرورة تعطل العامل. والثاني هو التوازي: يمكن إجراء محاولات مستقلة لقيم مؤقتة في آن واحد. تقلل النوى الإضافية زمن البحث المنقضي حتى عندما يظل العدد الإجمالي المتوقع للمحاولات ثابتًا. لذا تهم إنتاجية العتاد سياسة القبول. ولا يستطيع معدل تجزئة معلن ولا قياس أداء واحد أن يحد أفضلية كل صاحب طلب.

تمرين 3.0. باستخدام ذيل التوزيع الهندسي، قدّر احتمال الحاجة إلى أكثر من 4.6×2^b محاولة. لماذا لا يتغير العدد المتوقع للمحاولات عند تقسيم فضاء القيم المؤقتة بين عدة عمال؟
تلميح: احسب مجموع المحاولات، لا الثواني المنقضية.

إثبات العمل المتسلسل: المستوى الثاني

الأهداف نفهم إسناد التسميات في Cohen-Pietrzak، ولماذا يفرض الحساب المتسلسل، وما يفحصه المتحقق، وحد أخذ العينات المشروط $(1-\alpha)^t$ وحدوده، ونتيجة ما بعد الكم، وحد ذاكرة المُبرهِن، وبيانات المعايرة التي تستند إليها خريطة بتات العمل.

لماذا العمل المتسلسل

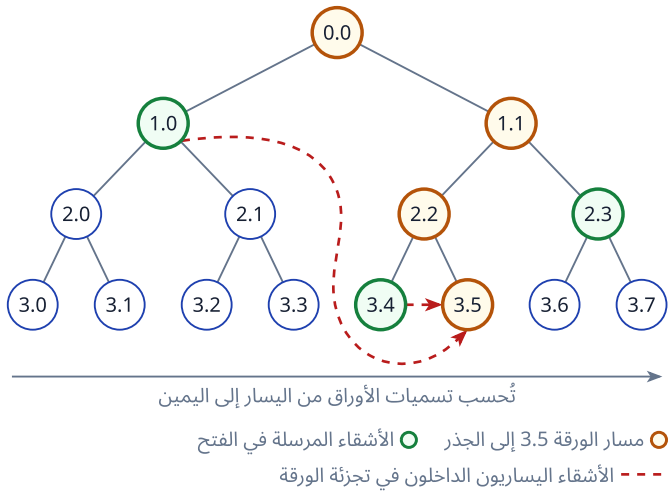
يُدخل بناء العمل المتسلسل اعتماديات بين الحسابات: فالتسمية اللاحقة تحتاج إلى قيمة نتجت قبلها. ويعالج ذلك كمية مختلفة عن مجموع العمل، هي عمق الحساب. وتظل الألغاز المستقلة قابلة للتنفيذ بالتوازي، ويظل زمن التجزئة في العتاد مهمًا.

يتبع Sibuna البناء القائم على التجزئة الذي وصفه Cohen وPietrzak. يحتوي رسمه على $2^{n+1}-1$ تسمية عند العمق n، لكن العدد الثابت من التسميات لا يعني زمنًا حتميًا في المتصفح: فقد تمتد التسمية على عدة كتل ضغط، وقد تعيد عمليات الفتح حساب أشجار فرعية، وتضيف الجدولة تفاوتًا. ولا يجوز أن تتحول العبارات الأمنية المستمدة من تحليلات نموذج الأوراكل إلى ادعاء مطلق بشأن كل تنفيذ ملموس أو كل خصم.

البناء

ليكن n هو العمق، ولننظر إلى الشجرة الثنائية الكاملة التي تُسمى عقدها بسلاسل ثنائية لا يتجاوز طولها n (الجذر ε، والابنان $v0$ و$v1$، والأوراق بطول n). تحمل كل عقدة v تسمية من 32 بايت $\ell_v = H(\chi, v, \text{parents}\ (v))$، حيث χ هي مسألة التحدي وH هي SHA-256 بعد استيعاب χ مرة واحدة (حيلة البادئة نفسها في برنامج حل Hashcash):

- تجزئ العقدة الداخلية ابنيها: $\ell_v = H(\chi, v, \ell_{v0}, \ell_{v1})$؛
- تجزئ الورقة $u = u_1 \ldots u_n$ الأشقاء اليساريين لأسلافها حيثما اتجه مسار الجذر إلى اليمين: $\ell_u = H(\chi, u, \{\ell_{u_1 \ldots u_{i-1} 0} : u_i = 1\})$.



الشكل 3 الشجرة بعمق 3: يرسل فتح الورقة 3.5 تسمية الورقة وتسميات الأشقاء الثلاثة؛ والحواف المتقطعة هي اعتماديات الأشقاء اليساريين التي تجعل إسناد التسميات متسلسلًا.

حواف الأشقاء اليساريين هي ما يجعل الحساب متسلسلًا: لا يمكن إسناد تسمية إلى الورقة u قبل أن تكتمل كل شجرة فرعية على يسارها، فيُفرض اجتياز بالعمق أولًا بترتيب لاحق (post-order)، وتعتمد تسمية الجذر $\varphi = \ell_\varepsilon$ على جميع التسميات البالغ عددها $2^{n+1} - 1$ بالتسلسل.

عمليات الفتح. بعد حساب φ، يشتق المُبرهِن t ورقة بطريقة Fiat-Shamir، $\gamma_i = H(\chi \,\|\, \varphi \,\|\, i) \bmod 2^n$، ويرسل لكل منها تسمية الورقة وتسميات الأشقاء البالغ عددها n على مسارها إلى الجذر. يعيد المتحقق حساب الورقة من الأشقاء الذين هم آباؤها، ويجزئ صعودًا عبر المسار، ويتحقق من أنه يصل إلى φ. تكلفة المتحقق $O(tn)$ حساب تسمية بمساحة عمل محدودة. وقد يمتد حساب التسمية الواحدة على عدة كتل تجزئة؛ فعدّ التسميات ليس عدًّا لاستدعاءات دالة الضغط.

> **التنفيذ: `posw::verify_opening`** يعيد حساب ورقة مفتوحة من تسميات أشقائها ويتبع المسار حتى الجذر. في `crates/pow/src/posw.rs` (الحزمة `sibuna-pow`).

```
fn verify_opening(base: &Sha256, n: u8, phi: &Label, gamma: u32, opening:
&[Label]) -> bool {
    let Some((leaf, siblings)) = opening.split_first() else {
        return false;
    };
    // تمتد `siblings` من العمق n صعودًا إلى العمق 1؛ وبعد عكسها تقترن بالأعماق
.1..=n
    let mut hasher = node_hasher(base, n, gamma);
    for (d, sibling) in (1..=n).zip(siblings.iter().rev()) {
        if turns_right(gamma, n, d) {
            hasher.update(sibling);
        }
```

```
    }
    if hasher.finalize() != *leaf {
        return false;
    }
    let mut current = *leaf;
    for (d, sibling) in (1..=n).rev().zip(siblings) {
        let mut parent = node_hasher(base, d - 1, gamma >> (n - d + 1));
        if turns_right(gamma, n, d) {
            parent.update(sibling);
            parent.update(&current);
        } else {
            parent.update(&current);
            parent.update(sibling);
        }
        current = parent.finalize();
    }
    current == *phi
}
```

أخذ العينات والسلامة وحدودهما

لننظر في التزام ثابت فيه نسبة سيئة α يمكن كشفها كشفًا مستقلًا. إذا أخذت كل عملية فتح عينتها بتوزيع منتظم وباستقلال، فإن احتمال تفويت كل المواضع السيئة في t عملية فتح هو $(1-\alpha)^t$. يوضح هذا الحساب البسيط لماذا تفيد زيادة عمليات الفتح، لكنه وحده ليس برهانًا أمنيًا لبروتوكول عمل متسلسل.

مثال محلول: حد مشروط لأخذ العينات مع نسبة سيئة ثابتة $\alpha = 0.1$ و$t = 16$ عملية فتح مستقلة، يكون احتمال تفويت كل المواضع السيئة $0.9^{16} \approx 0.185$. ومع 32 عملية فتح يبلغ نحو 0.034. تفترض المقدمة أن الالتزام ثُبِّت قبل اختيار العينات. فإذا استطاع المُبرهِن تجربة جذور كثيرة، أو اختيار الالتزام الذي يكشفه، أو جعل حالات الفشل مترابطة، فإن هذا الحساب لا يأخذ تلك الاستراتيجية في الحسبان.

لبناء العمل المتسلسل المشار إليه تحليل أمني في نموذج أوراكل. ومع ذلك يجب التحقق من أن فصل المجالات في Sibuna، وخريطة المعلمات، والاقتطاع، والتنفيذ تفي بافتراضات البناء. تجعل عينات Fiat-Shamir عمليات الفتح قابلة لإعادة الإنتاج من الالتزام؛ لكن قابلية إعادة الإنتاج وحدها لا تثبت مقاومة التجربة المتكررة (grinding). ولا ندعي دون قيد أن الغش لا يكون أرخص أبدًا.

تمرين 3.1. إذا جرب مهاجم g التزامًا مستقلًا، يُقبل كل منها باحتمال q، فاشتق احتمال قبول واحد منها على الأقل. ما التكاليف الغائبة عن هذا التعبير؟
تلميح: احسب أولًا احتمال فشل الالتزامات g كلها.

ذاكرة المُبرهِن

يخزن مُبرهِن ساذج كل تسمية (وعددها $2^{n+1}-1$). يحتفظ مُبرهِن Sibuna في المرور الأول بمكدس لتسميات الأشقاء اليساريين للمسار الحالي ($n+1$ تسمية)، ولا يبقي إلا أعلى $m=\min(n,10)$ مستوى ($2^{m+1}-1$ تسمية، وبحد أقصى 64 KB). لفتح ورقة، يعيد حساب الشجرة الفرعية لسلف الورقة عند العمق m، أي $2^{n-m+1}-1$ تجزئة، مع تهيئة المكدس من المستويات المحفوظة. التكلفة الإجمالية للمُبرهِن $(2^{n-m+1}-1)\,t-1+2^{n+1}$ تجزئة، ومساحة العمل كلها، بما فيها الإثبات الناتج، أقل من 100 KB: يشترك تبويب المتصفح والاختبار الأصلي في النوع Workspace نفسه (`sibuna_pow::posw::Workspace`).

المعلمات ليست أزمنة

يربط Sibuna بتات العمل بالعمق باستخدام $n=b-3$، مع حصره في المجال المدعوم $[4,24]$. وهذا اصطلاح لمقياس العمل، وليس ضمانًا لتساوي الزمن الفعلي بين الخوارزميتين. فالورقة قد تجزئ عدة تسميات آباء، بينما يختبر Hashcash قيمة مؤقتة مرة بعد مرة. وتؤثر خيارات المترجم والمتصفح والمعالج والمعلمات كلها في تكاليفهما النسبية. يقيس قياس أداء العمليات الأولية المودَع في المستودع التحقق؛ ولا يثبت زمن الحل في المتصفح.

حجم الإثبات $32(1+t(n+1))$ بايت. عند العمق 13 مع ست عشرة عملية فتح، يبلغ $32(1+16\times14)=7200$ بايت. وزيادة عدد عمليات الفتح تزيد حجم الإثبات وعمل المتحقق معًا، حتى إذا ظلت التكلفة الغالبة لبناء الشجرة ثابتة.

تمرين 3.2. احسب حجم الإثبات عند العمق 17 مع 16 عملية فتح. ثم ضاعف عدد عمليات الفتح. أي حد يتضاعف، وأي حد يظل ثابتًا؟

اختيار البناء

المقارنة المناسبة هي ميزانية الموارد، لا سمعة الخوارزمية. سجّل ذاكرة المُبرهِن، وعمل المتحقق في أسوأ الحالات، وبايتات الإثبات، والافتراضات اللازمة للخاصية المعلنة. قد يفيد البحث المستنزف للذاكرة (memory-hard)، لكن المتحقق الذي يضطر إلى تكرار عمل مكلف يمنح المهاجم موضعًا آخر لاستنزاف موارد الخادم. وتخدم بناءات التأخير القابلة للتحقق العلني نموذج ثقة مختلفًا عن نموذج بوابة تقوم على سر مشترك.

يحتفظ Sibuna بتحقق تجزئة قليل التكلفة، وبخيار عمل متسلسل قائم على التجزئة. يحتاج الأخير إلى بايتات إثبات أكثر وتحليل أدق للمعلمات. لا يلغي أي من الخيارين ضرورة رفض الإثباتات المشوهة قبل الدخول في تحقق مكلف.

اشرح اللامتغير. اشرح الفرق بين شجرة المصادقة، التي تثبت انتماء تسمية إلى التزام، وحواف الاعتماد، التي تقيد طريقة حساب التسميات.

المصادقة المتماثلة: التحديات والجلسات

الأهداف نشرح لماذا تكون الدالة شبه العشوائية ذات المفتاح هي الأداة الأولية الصحيحة عندما ينتمي المُصدر والمتحقق إلى مجال الثقة نفسه، ونذكر حد التزوير لوسم من 128 بت، ونصف سجل التحدي عديم الحالة وحد مجموعة الحلول المُنفَقة، ونقرأ جدول اشتقاق المفاتيح.

المُصدر هو المتحقق

يتيح التوقيع الرقمي لأي أحد يحمل المفتاح العام أن يتحقق، وهو الأداة المناسبة عندما يجب ألا يتمكن المتحققون من إصدار الرموز. متحقق Sibuna هو المُصدر (أو عنقود يتشارك بذرة واحدة). يوفر رمز مصادقة الرسالة المبني من دالة شبه عشوائية المصادقة داخل مجال الثقة هذا. ولا يوفر التحقق العلني: كل من يحمل مفتاح MAC يستطيع أيضًا إصدار الرموز. يقارن فصل قياسات الأداء عمليات الرموز المنفذة.

تمهيدية 3 (حد التزوير) عند نمذجة BLAKE3 في الوضع ذي المفتاح بوصفه PRF مع وسم من 16 بايت، يزور خصم يجري q استعلام تحقق رمزًا صالحًا باحتمال لا يتجاوز $q \cdot 2^{-128} + \varepsilon_{\mathrm{PRF}}$. عند مليون استعلام في الثانية يكون ذلك 2^{-108} في الثانية.

ولنتيجة ثانية أهمية على المدى البعيد: يعتمد Ed25519 على اللوغاريتمات المتقطعة في زمرة منحنى إهليلجي، وتكسرها خوارزمية Shor؛ أما التجزئة ذات المفتاح فتعتمد على دالة التجزئة وحدها، ولا تعطي الخوارزميات الكمومية ضدها أكثر من تسارع تربيعي. يحتفظ Sibuna بـ Ed25519 خيارًا (`--token-scheme ed25519`) لعمليات النشر التي تحتاج إلى متحققين لا يستطيعون إصدار الرموز.

جدول اشتقاق المفاتيح

تُشتق من بذرة رئيسية واحدة من 32 بايت (من `--secret-file` أو `SIBUNA_SECRET` أو قيمة عشوائية جديدة) مفاتيح الأغراض كلها باستخدام BLAKE3 ذي مفتاح وسلسلة مجال: مفاتيح الرمز، والتحدي، وEd25519، والبصمة، ومفتاح المتطلبات الذي يختم تذاكر المتطلبات (الجزء الرابع).

التنفيذ: `Keys::derive` يشتق مفاتيح الأغراض الخمسة من البذرة. في `crates/challenge/src/keys.rs` (الحزمة `sibuna-challenge`). يصفّر `Keys` و`Seed` بايتاتهما عند إسقاطهما، ويُحجب خرج `Debug` الخاص بهما.

```
pub fn derive(seed: &Seed) -> Keys {
    Keys {
        token: subkey(seed, "sibuna/token/v1"),
        challenge: subkey(seed, "sibuna/challenge/v1"),
        ed25519_seed: subkey(seed, "sibuna/ed25519/v1"),
        fingerprint: subkey(seed, "sibuna/fingerprint/v1"),
```

```
        requirement: subkey(seed, "sibuna/requirement/v1"),
    }
}

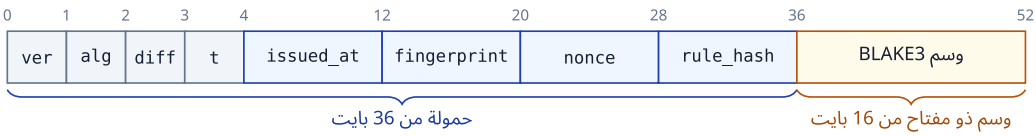
fn subkey(seed: &Seed, purpose: &str) -> [u8; 32] {
    blake3::keyed_hash(seed.as_bytes(), purpose.as_bytes())
}
```

في العنقود، يضع `Keys::bind_node` مكان مفتاح التحدي ناتجَ BLAKE3 ذي مفتاح، مفتاحه مفتاح التحدي نفسه، محسوبًا على `sibuna/challenge-node/v1` ومعرّف العقدة؛ فيجب أن يعود التحدي إلى العقدة التي أصدرته، بينما تظل رموز الجلسة وتذاكر المتطلبات صالحة على كل عضو.

يمنع فصل المجالات إعادة الاستخدام العرضية بين الأغراض؛ واختراق البذرة الرئيسية يخترق كل مفتاح مشتق. لا يحتاج العنقود إلا إلى الاتفاق على بذرة واحدة.

تحديات عديمة الحالة

معرّف التحدي



52 بايت في المجموع، تُرسَل في 70 محرفًا بترميز base64 الآمن في URL

رمز الجلسة



56 بايت في المجموع، تُرسَل في 75 محرفًا بترميز base64 الآمن في URL

الشكل 4 تنسيقات النقل لمعرّف التحدي ورمز الجلسة. الأرقام فوق كل سجل هي إزاحات البايتات.

معرّف التحدي سجل ذاتي المصادقة: الإصدار، والخوارزمية، والصعوبة، وعدد عمليات الفتح، ووقت الإصدار، وبصمة العميل ذات المفتاح، وقيمة مؤقتة مشتقة من PRF، وتجزئة قاعدة السياسة التي طلبت التحدي، يتبعها وسم من 16 بايت بمفتاح التحدي. وإصداره لا يكتب شيئًا. يفك التحقق ترميز السجل، ويفحص الوسم في زمن ثابت، ويقارن العمر بمدة صلاحية التحدي، ويقارن البصمة بالعميل المرسل، وعندئذ فقط يفحص الإثبات.

تمهيدية 4 (لا تنمو الحالة إلا بالعمل المدفوع) لتكن S مجموعة وسوم التحديات التي يتذكرها البرنامج الخدمي. لا يدخل وسم إلى S إلا بعد قبول إثبات صالح له، ويخرج منها بعد مدة صلاحية التحدي. لذا $|\,S\,| \le r \cdot \mathrm{TTL}$ عندما تحد r معدل الحلول المختلفة المقبولة خلال تلك الفترة. ولا يُخصَّص وسم عند جلب

تحدٌّ مجاني. وللتنفيذ أيضًا سعة ثابتة، وقد يرفض حلًا صالحًا عند امتلائه. هذا حد للحالة، وليس حدًا أدنى لاستراتيجية الحساب لدى كل خصم.

مجموعة الحلول المُنفَقة 16 شريحة بعنونة Robin Hood المفتوحة على الوسوم ذات 16 بايت (الجزء الخامس). والبصمة تجزئة ذات مفتاح لعنوان العميل وUser-Agent، مع فصل الأطوال كي لا يتصادم مدخلان بالضمّ، ومع مفتاح كي لا يمكن حساب بصمات عملاء آخرين دون اتصال (offline).

تمرين 3.3. تحمل حمولة الرمز مستوى العمل، و`timestamp`، و`expiry`، و`rule_hash`، و`fingerprint`. اشرح ما يمنعه كل حقل لو حُذف، ولماذا يجب أن يغطي الوسم الحقول الخمسة كلها.

اشرح اللامتغير. يقترح زميل تحويل جميع الرموز إلى Ed25519 "لأنه أقوى". اشرح حجة مجال الثقة، وفرق التكلفة، واعتبارات ما بعد الكم.

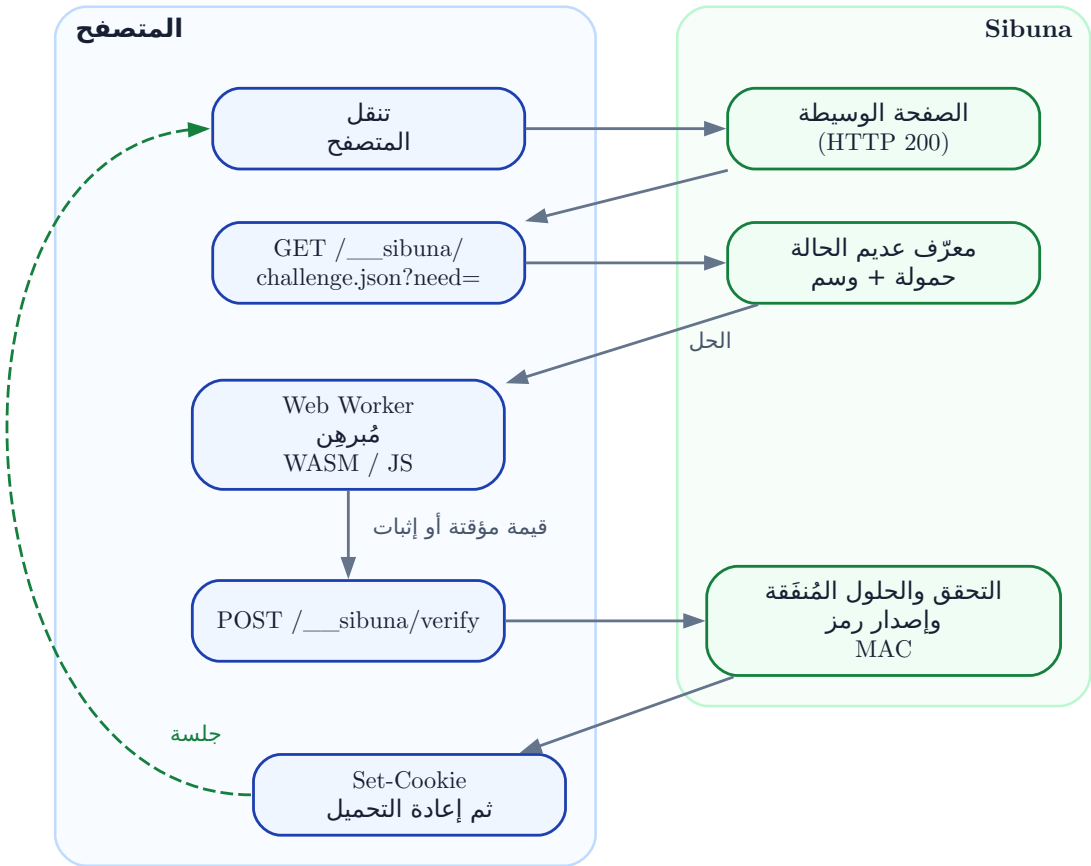
الجزء الرابع

بروتوكول Sibuna

نحدد بروتوكول النقل كما هو منفَّذ: دورة الصفحة الوسيطة، وسجل التحدي عديم الحالة، وتنسيقي الحل، وترتيب التحقق، وفرض الاستخدام لمرة واحدة، وملف تعريف ارتباط الجلسة.

دورة التحدي

الأهداف بنهاية هذا الفصل، سنستطيع تتبع المتصفح من أول طلب إلى إنشاء جلسة، وتسمية كل نقطة نهاية وكل حقل JSON مشارك، وشرح سبب حمل التحدي لقرار السياسة الخاص بالمسار المحمي.



الشكل 5 دورة التحدي. تجري الخطوات التي على اليسار في المتصفح، والتي على اليمين في Sibuna.

1. يصل تنقل بلا جلسة صالحة إلى محرك السياسة، فيُصنَّف `CHALLENGE`. ولأن الطلب يقبل `text/html`، يجيب البرنامج الخدمي بالحالة `200` مع الصفحة الوسيطة والترويسة `X-Sibuna-Status: CHALLENGE`؛ أما عميل API الذي لا يقبل HTML فيتلقى `401` مع جسم JSON يحمل عنوان `challenge` جاهزًا للجلب. وفي الحالتين تحمل الاستجابة تذكرة متطلبات: التحدي الذي يفرضه هذا القرار (الخوارزمية، وبتات العمل، وعدد عمليات الفتح) وتجزئة القاعدة، مختومة لهذا العميل بوسم BLAKE3 ذي مفتاح بمفتاحها المشتق الخاص، وصالحة مدة عمر تحدٍّ واحد.
2. تجلب الصفحة الوسيطة `GET /__sibuna/challenge.json?path=<original URL>&need=<ticket>`. فإذا كانت التذكرة صالحة حددت التحدي: فالعمل يحدده القرار الذي طلبه، ولا يُعاد حسابه من عنوان URL يبلغ عنه العميل ومن ترويسات الجلب نفسه، وهي تختلف عن ترويسات التنقل (`Accept`، و`Sec-Fetch-*`). ومن دون تذكرة صالحة (عميل API يتجاهل العنوان، أو صفحة بقيت مفتوحة بعد انقضاء العمر) يقيّم الخادم عنوان URL المُبلَّغ عنه، مقسومًا إلى مسار واستعلام بالدالة نفسها التي تقرأ سطر الطلب، بعد وضع المسار في الصيغة القياسية التي يستخدمها القبول (SID 0009)؛ ويُرفض العنوان الأطول من 8 KiB بالحالة `414` ولا يُقتطع أبدًا، ويُرفض المسار الذي ليست له صيغة قياسية بالحالة `400`. والتذكرة استرشادية: يظل القبول يفحص مستوى عمل الجلسة، فلا تكلف التذكرة المعادة أو المزورة حاملها إلا عملًا.
3. الاستجابة هي سجل التحدي:

```
{"id":"AQEND...70
chars","algorithm":"posw","difficulty":13,"challenges":16,"expires_at":1757241234}
```

بالنسبة إلى `hashcash`، يمثل `difficulty` عدد البتات، ويساوي `challenges` القيمة `0`.

4. يحله Web Worker بوحدة WebAssembly، أو بمُبرهِن JavaScript إذا لم تتوفر WebAssembly (الجزء السابع)، ويرسل إما

```
{"challenge_id":"...","nonce":"90766"}
```

أو

```
{"challenge_id":"...","proof":"<base64url, 32(1 + t(n+1)) bytes>"}
```

إلى `POST /__sibuna/verify`.

5. عند النجاح يجيب الخادم بـ `200 {"status":"ok"}` مع ترويسة `Set-Cookie`، وتعيد الصفحة تحميل موقعها الأصلي. يحمل كل طلب لاحق ملف تعريف الارتباط، ويُقبل قبل تقييم السياسة.

> **لماذا يرافق المسار التحدي** تُقدَّم الصفحة الوسيطة عند العنوان المحمي، فيعرف المتصفح موقعه. ويتيح إرسال المسار للخادم اختيار الصعوبة والخوارزمية الخاصتين بكل قاعدة (قد يتطلب مسار الدفع 20 بت عمل

من العمل المتسلسل، بينما تتطلب مدونة 16 بت من Hashcash)، ويربط تجزئة القاعدة بالرمز، حيث تنقلها `X-Sibuna-Rule-Hash` إلى الخادم الأصلي.

سجل التحدي عديم الحالة

الأهداف اقرأ مرمِّز التحدي، وافهم غرض كل حقل، وانظر كيف تُشتق القيمة المؤقتة لكل تحدٍّ من دون استدعاء نظام لجلب العشوائية في المسار الساخن.

التنفيذ: `Coordinator::create_challenge_with_spec` ينشئ حمولة من 36 بايت، ويضع لها وسمًا بمفتاح التحدي، ويرمز البايتات الـ 52 بترميز base64url إلى معرف من 70 محرفًا. في `crates/challenge/src/coordinator.rs` (الحزمة `sibuna-challenge`).

```
pub fn create_challenge_with_spec(
    &self,
    client_ip: &[u8],
    user_agent: &[u8],
    now: u64,
    spec: ChallengeSpec,
    rule_hash: u64,
) -> Challenge {
    self.adaptive.observe(now.saturating_mul(1000));
    let effective = ChallengeSpec {
        difficulty: spec.difficulty.saturating_add(self.adaptive.bump()),
        ..spec
    };
    let fingerprint = self.fingerprint(client_ip, user_agent);
    let difficulty = effective.parameter();
    let challenges = match effective.algorithm {
        Algorithm::Hashcash => 0,
        Algorithm::Posw => effective.posw_challenges,
    };

    let mut payload = [0u8; ID_PAYLOAD_LEN];
    payload[0] = ID_VERSION;
    payload[1] = effective.algorithm.to_byte();
    payload[2] = difficulty;
    payload[3] = challenges;
    payload[4..12].copy_from_slice(&now.to_le_bytes());
    payload[12..20].copy_from_slice(&fingerprint.to_le_bytes());
    payload[20..28].copy_from_slice(&self.next_nonce(now,
fingerprint).to_le_bytes());
    payload[28..36].copy_from_slice(&rule_hash.to_le_bytes());
    let mut raw = [0u8; ID_RAW_LEN];
    raw[..ID_PAYLOAD_LEN].copy_from_slice(&payload);
```

```
    raw[ID_PAYLOAD_LEN..].copy_from_slice(&self.challenge_tag(&payload));

    Challenge {
        id: encoded::encode(&raw),
        algorithm: effective.algorithm,
        difficulty,
        challenges,
        expires_at: now.saturating_add(u64::from(self.challenge_ttl)),
    }
}
```

المعرّف من النوع `Encoded<70>`: يُحفظ النص في مصفوفة بطوله تمامًا، فلا يخصص إصدار التحدي أي ذاكرة. وتحوّل `ChallengeSpec::parameter` بتات العمل إلى وحدة المستوى نفسه: بتات Hashcash (64 على الأكثر)، أو عمق PoSW (`difficulty - 3`، محصورًا بين 4 و24).

تستحق ثلاثة تفاصيل الانتباه:

- الصعوبة التكيفية. يحتفظ المنسق بمتوسط متحرك مرجح أسيًا لمعدل إصدار التحديات في ثلاث قيم ذرية. فوق خط أساس قدره 50 تحديًا في الثانية، تضيف كل مضاعفة للمعدل بت عمل واحدًا، بحد أقصى ستة. يدفع الإغراق تكلفة تزداد أسيًا، بينما تحتفظ الحركة الهادئة بالتكلفة الأساسية.
- القيمة المؤقتة خرج دالة PRF، بمفتاح التحدي على عداد والساعة والبصمة. وهي فريدة (بفضل العداد) وغير قابلة للتنبؤ (بفضل المفتاح)، فلا يستطيع الخصم حساب حلول مسبقًا لمعرفات لم تُصدر له، ولا يستدعي المسار الساخن نظام التشغيل أبدًا للحصول على العشوائية.
- الوسم يغطي الصعوبة. لا يستطيع العميل خفض الصعوبة بتعديل السجل: فسيفشل الوسم قبل فحص أي إثبات.

ترتيب التحقق والاستخدام الواحد

> **الأهداف** افهم لماذا تسبق الفحوص الرخيصة الإثبات، ولماذا لا يُسجَّل التحدي مُنفَقًا إلا بعد ثبوت صلاحية الإثبات، وكيف يُحسم تقديمان متزامنان للحل نفسه.

> **التنفيذ: `Coordinator::verify_and_mint`** يفك ترميز المعرّف ويصادقه، ويفحص العمر والارتباط بالعميل، ويتحقق من الإثبات للمستوى المسجل، ويسجل الوسم مُنفَقًا، ويصدر رمز الجلسة. في `crates/challenge/src/coordinator.rs` (الحزمة `sibuna-challenge`).

```
pub fn verify_and_mint(
    &self,
    challenge_id: &[u8],
    solution: Solution<'_>,
    client_ip: &[u8],
    user_agent: &[u8],
    now: u64,
```

```
) -> Result<Verified, VerifyError> {
    let record = self.decode(challenge_id)?;
    let expires_at =
record.issued_at.saturating_add(u64::from(self.challenge_ttl));
    if now > expires_at || record.issued_at >
now.saturating_add(CLOCK_SKEW_SECONDS) {
        return Err(VerifyError::ChallengeExpired);
    }
    if record.fingerprint != self.fingerprint(client_ip, user_agent) {
        return Err(VerifyError::FingerprintMismatch);
    }
    check_solution(challenge_id, &record, solution)?;
    self.spent.mark_spent(&record.tag, expires_at, now).map_err(|error| match
error {
        SpendError::DoubleSpend => VerifyError::DoubleSpendAttempt,
        SpendError::Full => VerifyError::StoreFull,
    })?;
    let work = WorkLevel {
        algorithm: record.algorithm.to_byte(),
        bits: record.algorithm.work_bits(record.difficulty),
    };
    Ok(Verified {
        token: self.mint_token(now, record.rule_hash, record.fingerprint, work),
        algorithm: record.algorithm,
        difficulty: record.difficulty,
        challenges: record.challenges,
        ttl_seconds: self.token_ttl,
    })
}
```

الترتيب مقصود. فالوسم والعمر والارتباط فحوص ثابتة الزمن ترفض المدخلات العبثية قبل أي تجزئة. ويجب ألا يستهلك الإثبات الخاطئ التحدي، لذا لا يُكتب في مجموعة الحلول المُنفَقة إلا بعد نجاح `check_solution`. وإذا تحقق خيطان من الحل الصالح نفسه تسابقا على `mark_spent`؛ فيرتّبهما قفل الشريحة، ويفوز واحد منهما بالضبط، ويتلقى الآخر `VerifyError::DoubleSpendAttempt`. ويرفض الشرط `issued_at > now + 60` (قيمة `CLOCK_SKEW_SECONDS` هي 60) السجلات الصادرة عن عقدة تتقدم ساعتها بأكثر من دقيقة، وإلا لأطالت هذه السجلات عمر التحدي.

تظهر الأخطاء للعميل بالحالة `400` مع تشخيص بأسلوب Elm (الجزء العاشر): يتحول كل `VerifyError` إلى `sibuna_core::errors::ErrorKind`، ويكون شرحه جسم الاستجابة. وتتحقق اختبارات الطرف إلى الطرف من وجود `DOUBLE SPEND` و`FINGERPRINT MISMATCH` و`WRONG SOLUTION TYPE` و`MALFORMED CHALLENGE` في جسم الاستجابة.

ملف تعريف ارتباط الجلسة

> **الأهداف** اقرأ الرمز القائم على التجزئة ذات المفتاح، وافهم سمات ملف تعريف الارتباط، واشرح الارتباط بالعميل.

التنفيذ: `token::mac::mint` و`token::mac::verify` يسلسل الحمولة ذات 40 بايت، ويلحق بها أول 16 بايت من BLAKE3 ذي المفتاح محسوبًا عليها، ويتحقق بمقارنة ثابتة الزمن. في `crates/challenge/src/token/mac.rs` (الحزمة `sibuna-challenge`). تستدعيه `Coordinator::verify_cookie` ببصمة العميل.

```
pub fn verify(
    key: &[u8; 32],
    text: &[u8],
    now: u64,
    expected_fingerprint: Option<u64>,
) -> Result<Payload, TokenError> {
    if text.len() != TOKEN_LEN {
        return Err(TokenError::InvalidTokenLength);
    }
    let raw: [u8; RAW_LEN] =
        encoded::decode::<RAW_LEN,
TOKEN_LEN>(text).ok_or(TokenError::InvalidEncoding)?;
    let (payload, given) = raw.split_at(PAYLOAD_LEN);
    let mut bytes = [0u8; PAYLOAD_LEN];
    bytes.copy_from_slice(payload);
    // زمن ثابت: الخروج المبكر يكشف للمزوّر عدد بايتات الوسم الصحيحة.
    if !ct::eq(&tag(key, &bytes), given) {
        return Err(TokenError::InvalidTokenSignature);
    }
    let payload = Payload::from_bytes(&bytes)?;
    payload.check(now, expected_fingerprint)?;
    Ok(payload)
}
```

يُفحص الوسم قبل تحليل الحمولة: لا تُرفض حمولة من إصدار آخر بالخطأ `TokenError::InvalidEncoding` إلا بعد التأكد من أصالتها، ولا تصل الحمولة المزورة إلى المحلل أبدًا.

يُصدر ملف تعريف الارتباط على النحو التالي

```
Set-Cookie: __sibuna_token=<75 chars>; Path=/; Max-Age=86400; HttpOnly;
    SameSite=Lax[; Secure]
```

تبقيه `HttpOnly` بعيدًا عن نصوص الصفحة البرمجية، وتمنع `SameSite=Lax` إعادة الإرسال عبر المواقع مع السماح بالتنقل في المستوى الأعلى، وتُضاف `Secure` مع `--secure-cookie` عندما يُنهى TLS أمام البرنامج الخدمي. ولأن الحمولة تحمل البصمة ذات المفتاح للعنوان وUser-Agent، يفشل ملف تعريف الارتباط المنسوخ إلى جهاز آخر مع `TokenError::TokenBoundAddressMismatch`؛ ويختبر اختبار الطرف إلى الطرف "the cookie is bound to the client identity" هذا المسار تحديدًا.

تبدأ الحمولة ببايت إصدار، ثم مستوى العمل الذي دفعه حاملها: الآلية (Hashcash أو PoSW) وبتات العمل المحلولة فعلًا، بما فيها أي زيادة تكيفية مع الحمل. ولا تستوفي الجلسة تحديًا إلا عندما يبلغ مستواها ما يتطلبه المسار، محسوبًا من صعوبة القاعدة بالحصر نفسه الذي يطبقه التحدي ذاته. والمستويات مرتبة لا مسماة: ملف تعريف الارتباط

المكتسب عند 24 بت يغطي أيضًا كل مسار من 16 بت، أما ملف 16 بت المقدَّم إلى مسار من 24 بت فيُحال إلى الصفحة الوسيطة الأقوى، ويحل الملف الجديد محل القديم. ويبقى `rule_hash` هوية التدقيق للقاعدة التي أصدرت التحدي؛ يُبلَّغ عنه إلى الخادم الأصلي، ولا يُستخدم أبدًا للقبول. ولا تُسقط الجلسة أبدًا نتائج WAF ولا الرفض الصريح. ويختبر اختبار الطرف إلى الطرف "a session earned on a cheaper route does not admit a route that demands more work" الاتجاهين كليهما.

العناوين الممررة خلف وكيل الدخول، يصل عنوان العميل في `X-Forwarded-For`. لا يثق Sibuna بهذه الترويسة إلا مع `--trust-forwarded` (وهو ضمني في وضع forward-auth). يجب أن يبقيه الوكيل العكسي المكشوف مباشرة للإنترنت معطلًا، وإلا استطاع أي عميل اختيار ارتباطه ودلو حد المعدل الخاص به.

تمرين 4.1. ارسم تسلسل متصفحين خلف عنوان NAT واحد بقيم User-Agent متطابقة. ما الفحوص التي يشتركان فيها، وما التي لا يشتركان فيها، وماذا يحتاج مهاجم على ذلك NAT لإعادة استخدام ملف تعريف ارتباط المتصفح الآخر؟

اشرح اللامتغير. اشرح لمشغل لماذا لا "يتذكر" البرنامج الخدمي التحديات المعلقة، ولماذا يعد ذلك خاصية أمنية لا اختصارًا.

الجزء الخامس

هندسة بلا تخصيص للذاكرة في Rust

نتتبع اتصالًا من قبوله إلى الاستجابة عبر مخزن مؤقت واحد ثابت الحجم، وندرس هياكل البيانات المتزامنة التي تتيح لكثير من خيوط الاتصالات مشاركة الحالة دون تخصيص في الكومة على مسار الطلب. وحيث تحتاج سلامة الذاكرة في العادة إلى آليات خاصة بها *(*عدادات القراء، والساحات، وجداول الواصفات، وسلاسل التفكيك*)*، نبين كيف توفرها ملكية *Rust* بدلًا منها.

حلقة الاتصال

الأهداف بنهاية هذا الفصل، ينبغي أن نتمكن من شرح كيف يخدم مخزن واحد من 64 KiB اتصال keep-alive كاملًا، وكيف يُحدَّد موقع الرأس والجسم دون نسخ، ولماذا يحصل كل اتصال على خيط محدود خاص به، وكيف يعرف الوكيل تأطير الخادم الأصلي حتى يبقى اتصال العميل المُمرَّر مفتوحًا، وأي هذه الضمانات يثبتها لنا مدقق الاستعارة.

خيط لكل اتصال، بحد أعلى

يتشارك عمال استقبال الاتصالات (`--workers`، واحد لكل وحدة CPU افتراضيًا، و64 على الأكثر) مقبس الاستماع. يُسلَّم كل اتصال مُستقبَل إلى خيط خاص به مع مكدس من ميغابايت واحد، وتعود حلقة استقبال الاتصالات مباشرة إلى `accept`. الخيط الذي ينهي اتصاله يتوقف وينتظر ثم يأخذ الاتصال التالي، لأن إنشاء خيط وإنهاءه لكل اتصال كان يعطل الخيوط الأخرى (الجزء الثامن، SID 0002). أما البديل، أي خدمة الاتصال حتى نهايته على خيط الاستقبال، فكان التصميم الأول: كانت قياساته جيدة مع عميل واحد وسيئة مع أربعة وستين، لأن ستين منهم انتظروا عاملًا حتى ينهي حصته من 256 طلبًا، ونما زمن الاستجابة في الذيل مع كل عميل منتظر. مع خيط لكل اتصال، يبقى p99 عند 64 اتصالًا تحت ميلي ثانية واحدة في كل عبء عمل للمنتج الكامل، من 305 us إلى 857 us (`benchmarks/results/product-latest.md`؛ الجزء الثامن).

عدد خيوط الاتصالات محدود بـ `--max-connections` (1,024 افتراضيًا، و8,192 على الأكثر). بعد هذا الحد، تجيب حلقة استقبال الاتصالات بـ `503 Service Unavailable` على المقبس الجديد وتغلقه دون أن تطلق أي خيط، وتُحصي الحدث في `sibuna_overloaded_total`.

التنفيذ: `accept_loop` تقبل الاتصالات، وتفحص كلًّا منها مقابل حد الاتصالات، وتطلقه على خيط خاص به من مجمع المهام. في `cmd/sibuna/src/server.rs` (الحزمة `sibuna`).

```
fn accept_loop(server: &Arc<Server>, listener: &TcpListener) {
    while !server.is_stopping() {
        let stream = match listener.accept() {
            Ok((stream, _)) => stream,
            // ... نفدت الواصفات أو الذاكرة: انتظر 1 ms بدل الدوران
        };
        let limit = server.max_connections.load(Ordering::Relaxed);
        if server.connections.fetch_add(1, Ordering::Relaxed) >= limit {
            server.connections.fetch_sub(1, Ordering::Relaxed);
            server.metrics.overloaded.bump();
            let text = "Service Unavailable: connection limit reached";
            reject(server, &stream, text);
            continue;
        }
        set_no_delay(&stream);
        let task_server = Arc::clone(server);
        let launched = server.tasks.launch(move || {
            let _counted = Counted(&task_server);
            crate::connection::serve(&task_server, &stream);
        });
        // ... إخفاق الإطلاق يعيد المكان ويُحصي حالة حمل زائد
    }
}
```

`Counted` حارس يعيد `Drop` الخاص به مكان الاتصال ضمن الحد أيًّا كانت طريقة انتهاء الخيط، بالعودة أو بالذعر. يستنزف مسار 503 ما أرسله العميل قبل أن يغلق جانبه من المقبس: إغلاق مقبس فيه بايتات غير مقروءة يُرسل إعادة ضبط (reset)، وإعادة الضبط تُسقط 503.

تأتي الخيوط من `TaskPool` (`cmd/sibuna/src/tasks.rs`)، وهو جدول ثابت من مواضع قابلة للضم (join). يحتفظ الخيط المنتهي بمكدسه حتى يُضَم، لذلك يُضَم الموضع عندما يؤخذ مرة أخرى، وتأخذ المهمة الجديدة أدنى موضع منتهٍ بدل تقديم مؤشر دوار. فالمؤشر الدوار لن يبلغ موضعًا منتهيًا مرة أخرى إلا بعد لفة كاملة على الجدول، تاركًا آلاف الخيوط المنتهية مقيمة في الذاكرة، فتنمو الذاكرة مع عدد الاتصالات المخدومة بدل عدد الاتصالات المفتوحة. ويُعلِّم حارس ثانٍ، `Finished`، الموضعَ حرًّا عند انتهاء المهمة، فيحرر الاتصال الذي يذعر موضعه أيضًا.

تُحَد الذاكرة بالطريقة نفسها. يحمل خيط الاتصال مخزن القراءة من 64 KiB ومخزن الكتابة من 16 KiB على مكدسه، ومساحة عمل للوكيل من 40 KiB في الكومة، تُخصَّص مرة واحدة عند بدء الاتصال. وبذلك تكون أسوأ حالة عددًا معروفًا لا دالة في عدد المقابس التي يستطيع عميل فتحها. ويغلق جامع الاتصالات الخاملة (أدناه) الاتصالات التي تتوقف عن الإرسال، فلا يستطيع عميل بطيء احتجاز خيط بعد `--idle-timeout`.

اختيار مخصص الذاكرة

يثبت Rust 1.100 الأجزاء الأساسية من واجهة `Allocator` ومعاملات مخصص الذاكرة لكل من `Vec` و``Box`` (ملاحظات الإصدار). يمكننا اختيار مخصص لمجموعة واحدة من دون استبدال مخصص العملية. لا توفر لنا هذه الواجهة تخصيصات إضافية في مسار الطلب: فالطلبات على الاتصالات المهيأة لا تخصص ذاكرة أصلا، كما يتحقق `cmd/sibuna/tests/steady_state.rs` و``crates/net/tests/steady_state.rs``. يخصص الوكيل مخازنه الثلاثة مرة واحدة لكل اتصال ويعيد استخدامها. كذلك يخصص المحدد ومجموعة البراهين المستهلكة

جدوليْهما مرة واحدة؛ وتسمح التخصيصات المصفرة ببقاء الصفحات غير المستخدمة خارج الذاكرة المقيمة، كما يتحقق `crates/store/tests/resident.rs`.

عند دراسة تغيير المخصص سنقيس أولا بناء لقطات السياسة وتحليل JSON في وحدة التحكم. تخصص مجموعاتها المؤقتة ذاكرة خارج مسار الطلب، لكن مخصص المجموعة لا يتحكم تلقائيا في تخصيصات عناصرها. تملك شجرة JSON أيضا السلاسل ذات محارف الهروب من خلال `Cow<str>`، الذي تكون صيغته المالكة `String`؛ وليس لـ``String`` معامل مخصص في Rust 1.100. وتبقى `Arc::new_in` غير مستقرة في هذا الإصدار (توثيق الواجهة). تظل المحركات المنشورة مشتركة عبر `Arc` حتى يحررها آخر قارئ، ولذلك لا يمكن لإعادة البناء تصفير ذاكرة ما زال محرك أقدم يستخدمها. يجب أن تحافظ ساحة الذاكرة على هذه الأعمار وحدود مدخلات المحلل.

نبقي المخصص العام الحالي `WipeOnFree` لوحدة WebAssembly الخاصة بوحدة التحكم في `cmd/console-ui/src/abi.rs`. يمحو الكتل المحررة، ومنها الكتلة القديمة بعد إعادة التخصيص. اختيار `System` مباشرة لمجموعة سيتجاوز هذا المحو. لا تحتوي وحدة WebAssembly لإثبات العمل على مخصص كومة، ويستخدم Argon2 بالفعل مساحة عمل يملكها المستدعي ويعيد استخدامها. لذلك نبقي المخصصات الحالية والتوافق مع Rust 1.94 حتى يبرر حمل مقاس تغييرا. يجب أن يحسن البديل زمن الاستجابة أو الذاكرة المقيمة مع الحفاظ على المحو والمعالجة المحدودة للمدخلات ومسار الطلب الخالي من التخصيص.

مخزن مؤقت واحد لكل اتصال

كل تخصيص ديناميكي على مسار الطلب أداة يمكن استغلالها لحجب الخدمة: تجزؤ على مدى أيام، وتنافس على قفل المُخصِّص بين الخيوط، وتضخيم من مهاجمين يصوغون طلبات تزيد التخصيص إلى أقصاه. قاعدة Sibuna مطلقة: التصنيف والتحقق والتسليم إلى الوكيل تجري بلا أي تخصيص في الكومة. يملك معالج الاتصال مخزن قراءة من 64 KiB ومخزن كتابة من 16 KiB طوال عمر المقبس؛ وكل ما عدا ذلك مقاطع منهما.

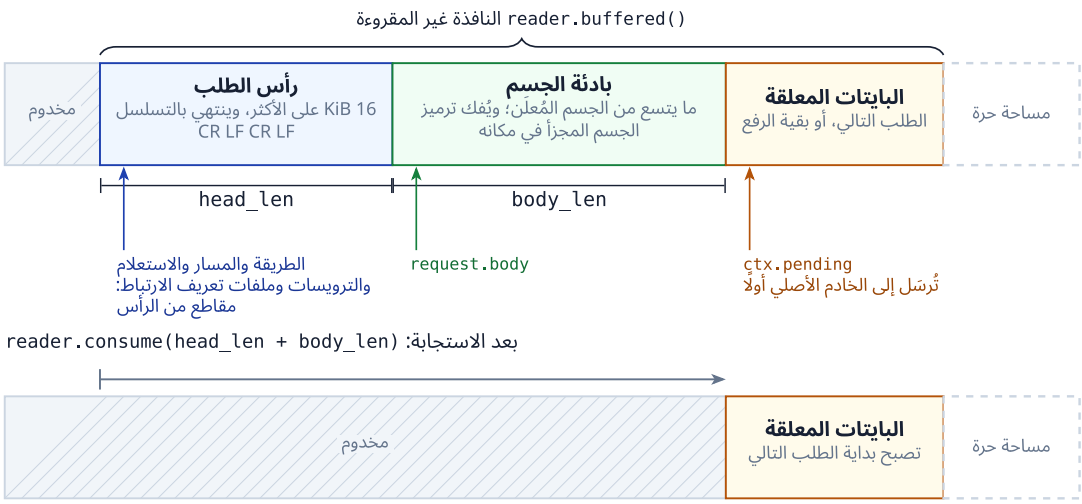
> **التنفيذ: `connection::serve`** تسجل الاتصال لدى جامع الاتصالات الخاملة، وتهيئ المخازن ومساحة عمل الوكيل وذاكرة المحرك المؤقتة مرة واحدة، وتخدم حتى 4,096 طلبًا. في `cmd/sibuna/src/connection.rs` (الحزمة `sibuna`).

```
pub fn serve(server: &Server, stream: &TcpStream) {
    let Some(lease) = server.idle.register(stream) else { return };
    // ...
    let mut read_buffer = [0u8; MAX_REQUEST_BYTES]; // 64 KiB
    let mut write_buffer = [0u8; WRITE_BUFFER]; // 16 KiB
    let mut reader = Reader::new(stream, &mut read_buffer[..]);
    let mut peer = PeerText::default();
    peer.format(stream.peer_addr().ok());
    let mut link = Link {
        server,
        stream,
        writer: Writer::new(stream, &mut write_buffer[..]),
        workspace: Workspace::new(),
        cache: EngineCache::new(),
```

```
        activity: lease.activity(),
        last: false,
    };
    for served in 0..MAX_REQUESTS_PER_CONNECTION {
        link.last = served + 1 == MAX_REQUESTS_PER_CONNECTION;
        if !serve_one(&mut reader, &mut link, peer.as_bytes()) {
            break;
        }
        lease.touch();
    }
}
```

تملأ read_head المخزن حتى يظهر السطر الفارغ، باحثةً فقط في البايتات التي وصلت منذ البحث الأخير، وترفض الرؤوس التي تزيد على 16 KiB بـ 431. تنتج parse_request (الحزمة sibuna-http) قيمة Request<'a> تكون طريقتها ومسارها واستعلامها وترويساتها وملفات تعريف الارتباط فيها مقاطع من المخزن. يُملأ الجسم المُعلَن بقدر ما يتسع ويُقتطَع بعد الرأس. ثم تُقدِّم reader.consume(head_len + body_len) القارئ، فيبدأ طلب keep-alive التالي بداية نظيفة؛ ولا ينقل القارئ البايتات غير المقروءة إلى المقدمة إلا عند حاجته إلى مساحة. يُفك ترميز الجسم المجزأ (chunked) في مكانه خلف الرأس (SID 0009): يُقتطَع تأطير كل خطوة من المخزن، فتبقى البادئة المفكوكة متصلة للفحص. المسارات الداخلية محددة الطول وتُبقي الاتصال مفتوحًا. والطلبات المُمرَّرة عبر الوكيل تُبقيه مفتوحًا أيضًا، بشرط أن تكون استجابة الخادم الأصلي مؤطَّرة وأن يكون الجسم قد خُزِّن بالكامل؛ وتبين الأقسام التالية كيف يقرر الوكيل ذلك.

مخزن القراءة لاتصال واحد، وسعته 64 KiB



الشكل 6 مخزن القراءة للاتصال أثناء خدمة طلب واحد. كل حقل من حقول الطلب مقطع منه.

```
fn serve_one<W: Write>(
    reader: &mut Reader<'_, &TcpStream>,
    link: &mut Link<'_, W>,
    peer: &[u8],
```

```
) -> bool {
    let head_len = match read_head(reader) {
        Head::Complete(length) => length,
        Head::TooLarge => { /* 431 "Request head exceeds 16 KB" */ return false; }
        Head::Closed | Head::Failed => return false,
    };
    let framing = {
        let bytes = reader.buffered();
        let head = bytes.get(..head_len).unwrap_or_default();
        let request = match parse_request(head) {
            Ok(request) => request,
            Err(error) => return reject_head(link, error),
        };
        let declared = request.content_length().unwrap_or(0);
        // ... Expect: 100-continue
        if declared == 0 && !request.chunked {
            let upload = Upload::declared(&request);
            let pending = bytes.get(head_len..).unwrap_or_default();
            let keep = dispatch(link, request, pending, upload, 0, peer);
            reader.consume(head_len);
            return keep;
        }
        (declared, request.chunked)
    };
    serve_with_body(reader, link, peer, head_len, framing)
}
```

ما يثبته مدقق الاستعارة تبقى مقاطع الطلب صالحة فقط حتى الملء التالي للمخزن. يعبر Sibuna عن هذه القاعدة بنوع لا بتعليق: الطلب من النوع Request<'a>، و'a استعارة مشتركة لمخزن القارئ. يحتاج ملء المخزن أو استهلاكه إلى &mut reader، وهو ما يرفضه المترجم ما دام طلب Request مستعار منه لا يزال قيد الاستخدام. في serve_one يُنقل الطلب إلى dispatch، ولا يجوز تشغيل reader.consume إلا بعد عودة ذلك الاستدعاء؛ وكتابتهما بالترتيب المعاكس لا تُترجَم.

تفسر القاعدة نفسها خيارين يبدوان غريبين أول الأمر. الأول أن serve_with_body تحلل الرأس مرة ثانية. يجب أن ينهي التحليل الأول استعارته قبل تخزين الجسم، لأن تخزين جسم مجزأ ينقل بايتات داخل المخزن؛ ويستعير التحليل الثاني المخزن بعد استقراره. التحليل حتمي ويستغرق نحو 133 ns لطلب متصفح مع ملف تعريف ارتباط (zero_copy_request_and_cookie، 132.5 ns، في benchmarks/results/primitives-latest.md)، لذلك يكلف التحليل الثاني قليلًا ويعطي الحكم نفسه. والثاني أن الوكيل يستعير الطلب المحلَّل والبايتات المعلقة ويجب ألا يعيد ملء مخزن الاتصال، فيقرأ بايتات العميل الإضافية في مخزن رفع خاص به. ولأنه لا يستطيع إعادة ما قرأه، لا يُحتفَظ باتصال العميل إلا عندما يكون الجسم قد خُزِّن بالكامل.

إعادة كتابة الرأس في الوكيل

لا يُمرَّر الرأس حرفيًا. تعيد write_request_head (crates/net/src/request.rs، الحزمة sibuna-net) إصدار سطر الطلب وكل ترويسة إلى كاتب الخادم الأصلي، مُسقطةً الحقول الخاصة بكل قفزة

(Connection وKeep-Alive وProxy-Connection وTransfer-Encoding وTE وTrailer وUpgrade وحقول تفويض الوكيل)، وأي حقل تمرير أو تدقيق وارد (X-Forwarded-For وX-Real-IP وForwarded و X-Forwarded-Proto وX-Forwarded-Host وX-Forwarded-Port وX-Sibuna-Status وX-Sibuna-Rule)، وExpect، وContent-Length، وكل حقل تسميه ترويسة Connection في الطلب. ثم تضيف مجموعة التدقيق. يُولَّد تأطير الجسم ولا يُنسخ أبدًا:

```
for header in request.headers() {
    if is_hop_by_hop(header.name) || nominated(request, header.name) {
        continue;
    }
    out.write_all(header.name)?;
    out.write_all(b": ")?;
    out.write_all(header.value)?;
    out.write_all(b"\r\n")?;
}
match upload {
    Upload::None => {}
    Upload::Length(length) => {
        out.write_all(b"Content-Length: ")?;
        write_decimal(out, *length)?;
        out.write_all(b"\r\n")?;
    }
    Upload::Chunked(_) => out.write_all(b"Transfer-Encoding: chunked\r\n")?,
}
```

يُعاد إصدار Content-Length. والجسم المجزأ الذي انتهى داخل مخزن الاتصال يُفك ترميزه في مكانه ويُرسل مع طوله. أما الأطول منه فيُعلَن بـ Transfer-Encoding: chunked ويُعاد تجزيئه قراءةً بقراءة (SID 0009): تصل كل قراءة إلى مخزن الرفع للوكيل من 16 KiB، ويُفك ترميزها هناك في مكانها، وتخرج جزءًا واحدًا بصيغة قياسية، فلا يرى الخادم الأصلي أبدًا أحجام أجزاء العميل ولا امتداداته ولا حقوله الختامية (trailers). مجموعة التدقيق:

```
Connection: keep-alive
X-Forwarded-For: <client>
X-Forwarded-Proto: <scheme>
X-Real-IP: <client>
X-Sibuna-Status: PASS
X-Sibuna-Rule: session | robots-txt | ip/cidr-trie | ...
```

تُرحَّل الأجسام الأكبر من المخزن على قطع من 16 KiB من العميل إلى كاتب الخادم الأصلي (relay_body) قبل بث الاستجابة عائدةً. يؤكد اختبار الوحدة proxy_head_rewrite_drops_hop_by_hop_headers_and_injects_audit_fields في crates/net/src/request.rs أن إعادة الكتابة تُسقط X-Forwarded-For المزيف وتحقن حقول التدقيق.

تعيش مخازن الوكيل الثلاثة، أي مخزن القراءة من الخادم الأصلي (رأس استجابة واحد، 16 KiB)، ومخزن الكتابة إلى الخادم الأصلي (8 KiB)، ومخزن الرفع (16 KiB)، في Workspace يخصصها الاتصال مرة واحدة ويعيد استخدامها لكل طلب. تجعل مساحة العمل عمر هذه المخازن عمر الاتصال لا عمر التبادل، ويستعيرها قارئ الخادم الأصلي وكاتبه لتبادل واحد في كل مرة.

ترحيل استجابة الخادم الأصلي

كان الوكيل الأول يبث بايتات الخادم الأصلي حتى يغلق الخادم الأصلي ثم يغلق العميل: تأطير صحيح بلا تحليل، لكن بثمن اتصال TCP جديد لكل طلب مُمرَّر. وتحت مولد حمل، ظهر هذا الثمن في صورة عشرات الآلاف من المقابس في `TIME_WAIT`، وفي واجهة الاسترجاع (loopback) في صورة منافذ مؤقتة مستنفدة. يقرأ الوكيل الآن رأس الخادم الأصلي (16 KiB على الأكثر) ويصنف الجسم وفق قواعد RFC 9112:

> **التنفيذ: `parse_response_head`** تعيد الحالة، وواحدًا من أربعة أنواع من التأطير (بلا جسم في HEAD وxxx1 وو204 و304؛ أو `Content-Length`؛ أو ترميز نقل مجزأ؛ أو جسم ينتهي بالإغلاق)، وما إذا كان الخادم الأصلي يُبقي الاتصال. في `crates/net/src/response.rs` (الحزمة `sibuna-net`).

```rust
pub enum Framing { None, Length(u64), Chunked, UntilClose }

pub fn parse_response_head(
    head: &[u8],
    head_request: bool,
) -> Option<ResponseHead> {
    let status = status_code(head)?;
    let mut fields = FramingFields {
        keep_alive: head.get(7) == Some(&b'1'),
        ..Default::default()
    };
    for line in crlf_lines(head).skip(1) {
        if line.is_empty() {
            break;
        }
        fields.read_line(line)?; // Transfer-Encoding, Content-Length, Connection
    }
    let mut framing = fields.length.map_or(Framing::UntilClose, Framing::Length);
    if head_request || status / 100 == 1 || status == 204 || status == 304 {
        framing = Framing::None;
    } else if fields.encoded {
        framing =
            if fields.chunked { Framing::Chunked } else { Framing::UntilClose };
    }
    let mut keep_alive = fields.keep_alive;
    if framing == Framing::UntilClose || fields.closing {
        keep_alive = false;
    }
    if fields.nominates_framing {
        return None;
    }
    Some(ResponseHead { status, framing, keep_alive })
}
```

يرفض المحلل ما يجب ألا يرحِّله الوكيل: سطر حالة معيبًا، واسم حقل ليس رمزًا لغويًا (token)، وبايت تحكم في قيمة، وأطوالًا متعارضة، وترميز `chunked` مكررًا أو غير أخير، وحقل `Connection` يسمي `Content-Length` أو `Transfer-Encoding`. عمله خطي في طول الرأس، ولا يذعر أبدًا.

يُعاد إصدار الرأس إلى العميل مع استبدال ترويسات Connection الخاصة بالخادم الأصلي بقرار Sibuna نفسه، ويُرحَّل الجسم بدقة: copy_exact للطول، وجزءًا بجزء (سطر الحجم، والبيانات، والذيول) للترميز المجزأ، و copy_remaining للحالة القديمة. الحالة القديمة وحدها تغلق العميل. يُبنى كل ترحيل على خطوة نسخ واحدة: مرِّر كل ما يحمله المصدر بالفعل، ولا تفرِّغ الوجهة ثم تقرأ مرة واحدة إلا عندما لا يحمل شيئًا. وبذلك يُسلَّم تيار من الكتابات الصغيرة كما يُنتَج، بينما يظل النقل الكبير ينقل مخزنًا كاملًا في كل قراءة وكتابة.

```
let keep = client_keep_alive && parsed.framing != Framing::UntilClose;
write_client_head(client, head, keep).map_err(client_head_error)?;
origin.consume(head_length);
match parsed.framing {
    Framing::None => {}
    Framing::Length(length) => copy_exact(origin, client, length, progress)?,
    Framing::Chunked => relay_chunked(origin, client, progress)?,
    Framing::UntilClose => copy_remaining(origin, client, progress)?,
}
client.flush().map_err(|_| ProxyError::ClientFailed)?;
Ok(Relayed { client_keep: keep, origin_reusable: parsed.keep_alive })
```

يستغرق تحليل رأس خادم أصلي من ثمانية حقول 322 ns، ويستغرق طلب GET بنمط keep-alive مع استجابة من 1 KiB مُمرَّرة عبر واجهة الاسترجاع 29.8 us (قيس في أكتوبر 2026؛ يوثق crates/net/src/lib.rs هذا القياس).

مجمع اتصالات الخادم الأصلي

يخبر تحليل التأطير الوكيل أيضًا ما إذا كان مقبس الخادم الأصلي قابلًا للاستخدام مرة أخرى: استجابة HTTP/1.1 بلا Connection: close (أو استجابة HTTP/1.0 مع keep-alive) استُهلك جسمها بالكامل تترك المقبس عند حد طلب نظيف. تدخل هذه المقابس مجمعًا ثابتًا من 256 اتصالًا خاملًا بالخادم الأصلي، ويأخذ الطلب المُمرَّر التالي واحدًا منها بدل الاتصال من جديد. من دون المجمع، يفتح كل طلب مُمرَّر اتصالًا جديدًا بالخادم الأصلي، ويحتجز كل اتصال مغلق منفذه المحلي في TIME_WAIT؛ وتحت الحمل، تنفد المنافذ المؤقتة سريعًا لدى وكيل على واجهة الاسترجاع، وتنهار إنتاجيته.

المجمع تحت قفل واحد يجعل كل طلب مُمرَّر يتنافس على ذلك القفل. لذلك يُقسَّم المجمع (crates/net/src/pool.rs) إلى شرائط: 16 شريطًا في كل منها 16 مقبسًا، وكل شريط Mutex<Vec<Arc<TcpStream>>> على سطر ذاكرة مخبئية (cache line) خاص به. يُسنَد إلى الخيط شريط أصلي بالتناوب الدوري عند أول استخدام، فيعيد المقابس إليه ويأخذها منه أولًا، ثم ينظر في الشرائط الأخرى، دون انتظار قفل محجوز، قبل أن يتصل من جديد.

```
pub fn take(&self) -> Option<Arc<TcpStream>> {
    if self.is_stopping() {
        return None;
    }
    let home = home_stripe();
    if let Some(socket) =
        self.stripe(home).and_then(|stripe| lock(&stripe.sockets).pop())
```

```
    {
        return Some(socket);
    }
    (1..STRIPES).find_map(|offset| {
        let stripe = self.stripe((home + offset) % STRIPES)?;
        try_lock(&stripe.sockets)?.pop()
    })
}
```

قد يكون الخادم الأصلي قد أغلق المقبس المُجمَّع أثناء خموله. يلاحظ الوكيل ذلك بإحدى طريقتين، كتابة فاشلة أو نهاية تيار قبل أول بايت من الاستجابة، وفي الحالتين لم يصل شيء إلى العميل بعد، فيغلق المقبس ويعيد المحاولة مرة واحدة على اتصال جديد، لا على مقبس مُجمَّع آخر أبدًا: بعد فترة خمول قد يكون كل مقبس مُجمَّع قديمًا، والنسخة الأولى من إعادة المحاولة هذه، التي كانت تأخذ مقبسًا مُجمَّعًا ثانيًا، أجابت بـ 502 على أول طلب بعد كل فترة هدوء. لا يُسمح بإعادة المحاولة إلا لطريقة آمنة (GET وHEAD وOPTIONS) خُزِّن جسمها بالكامل. فغياب الرد ليس دليلًا على أن الخادم الأصلي لم ينفذ تعديلًا، والجسم الذي رُحِّل على قطع لا يمكن إرساله مرة أخرى؛ وفي الحالتين يحصل العميل على 502.

```
let safe = matches!(request.method, Method::Get | Method::Head | Method::Options);
let buffered = exchange.upload.is_buffered(request.body.len());
let retryable = safe && buffered;
for attempt in 0..2 {
    // إعادة المحاولة تتصل دائمًا من جديد: بعد فترة خمول قد يكون كل مقبس
    // في المجمع قديمًا، ومقبس قديم ثانٍ سيُفشل الطلب.
    let pooled = if attempt == 0 { pool.take() } else { None };
    // ... اتصل إن لم يكن في المجمع مقبس، وألحِقه بالنشاط، ثم بادِل
    match outcome {
        Ok(relayed) => {
            if relayed.origin_reusable {
                pool.give(origin);
            }
            return Ok(relayed.client_keep && buffered);
        }
        Err(error) => {
            drop(origin);
            let stale = was_pooled
                && retryable
                && matches!(
                    error,
                    ProxyError::UpstreamClosed | ProxyError::UpstreamWriteFailed
                );
            // ... أي خطأ آخر يُبلَّغ عنه فورًا
        }
    }
}
```

ترحِّل اختبارات الوحدة سلاسل بايتات ثابتة عبر الدوال نفسها وتؤكد أن الخرج مطابق بايتًا ببايت باستثناء ترويسة الاتصال (crates/net/src/relay/tests.rs). ويرسل اختبار من طرف إلى طرف للبرنامج الخدمي طلبين مُمرَّرين على مقبس عميل واحد ويتحقق، عبر ترويسة تسلسل يضيفها الخادم الأصلي الوهمي (stub)، من

أن الطلب الثاني أعاد استخدام اتصال الخادم الأصلي المُجمَّع. ويعيد اختبار متجهات في `crates/net/tests/` تشغيل سيناريوهات وكيل مسجلة (طلب عميل خام واستجابة خادم أصلي جاهزة) ويقارن كل بايت يتلقاه الخادم الأصلي والعميل.

> **لماذا يكون مقبس الخادم الأصلي `Arc` واصف المقبس عدد صحيح صغير تعيد النواة استخدامه فور إغلاقه.** يجب أن يقاطع الإيقاف تبادلًا جاريًا، لكن يجب ألا يمس أبدًا واصفًا أُعيد استخدامه في هذه الأثناء لاتصال آخر. يمكن لجدول من الواصفات النشطة تحت قفل أن يحمي من ذلك، بثمن قفل في كل تبادل. لا يحتاج Sibuna إلى جدول كهذا: مقبس الخادم الأصلي `Arc<TcpStream>`. يحمل المجمع مقبضًا، ويحمل التبادل مقبضًا، ويحمل `Activity` الخاص بالاتصال نسخة منه أثناء جريان التبادل. يُغلق الواصف عند إسقاط آخر مقبض، فلا يمكن أن يُغلق، أو يُعاد استخدامه، ما دام أي أحد يستطيع الوصول إليه.

مهل الخمول

يسجل كل اتصال عميل مقبسه في `IdleTable` (`crates/net/src/idle.rs`) ويتلقى `Lease`. يحمل `Activity` الخاص بعقد الإيجار هذا الوقت الذي تحركت فيه البايتات آخر مرة على الاتصال. يحدِّثه البرنامج الخدمي بعد كل طلب مخدوم؛ ويحدِّثه الوكيل في كل قراءة من الخادم الأصلي ولكل 16 KiB من الرفع، ويُلحق به مقبس الخادم الأصلي للتبادل الجاري. يستيقظ خيط الجامع كل ربع أقصر مهلة (بين 200 ms وثانية واحدة) ويغلق كل اتصال صامت لمدة أطول من مهلته، مع مقبس الخادم الأصلي الملحق به، فيرى الخيط المتوقف في انتظار أي من الجانبين نهاية التيار ويحرر موضعه.

لذلك تقيس المهل الصمت، لا طول الاستجابة أبدًا: التنزيل الكبير الذي يظل يتحرك لا يُقطع أبدًا. أما بايتات الرفع فتأتي من الجانب غير الموثوق ولا تحدِّث الطابع الزمني إلا لكل 16 KiB، فيُطلب من العميل 16 KiB على الأقل في كل فترة خمول، وهذا هو الدفاع ضد الأجسام البطيئة الذي توفره مهل الطلبات ذات المعدل الأدنى.

للجدول قفل صغير لكل موضع، ولا قفل على مستوى الجدول كله. يحمل نسخة `try_clone` من كل مقبس عميل و`Arc` لكل مقبس خادم أصلي ملحق، للسبب المذكور أعلاه. يلغي `Lease` تسجيل اتصاله في `Drop`، ويبقى الموضع المحصود مملوكًا لعقد الإيجار الخاص به حتى ذلك الحين، فلا يستطيع اتصال قديم أبدًا تحرير موضع أخذه اتصال جديد.

الاتصالات المُرقّاة

بعد `101` مُتحقَّق منها، يُرحَّل الاتصال المُرقّى إلى WebSocket في الاتجاهين. يجري كل اتجاه على خيط خاص به مع مخزن ثابت واحد: يحمل خيط الاتصال نفسه البايتات من الخادم الأصلي إلى العميل، ويحمل خيط محدود النطاق (scoped) بمكدس من 128 KiB البايتات من العميل إلى الخادم الأصلي. كان يمكن لخيط واحد أن يخدم الاتجاهين باستخدام `poll(2)`، لكن المكتبة القياسية لـ Rust لا توفرها؛ وخيطان مع مهل للمقابس يُبقيان الترحيل خاليًا من `unsafe`. لذلك يحمل الاتصال المُرقّى خيطين، فقد يبلغ عدد خيوط الاتصالات ضعف `--max-connections` عندما تكون كل الاتصالات مُرقّاة.

يتشارك الاتجاهان مهلة خمول واحدة (`--websocket-idle-timeout`، 300 ثانية افتراضيًا): البايتات المتحركة في أي اتجاه تحدِّث طابعًا زمنيًا واحدًا. يحمل كل انتظار على مقبس مهلة تساوي الوقت المتبقي؛ والانتظار الذي

تنتهي مهلته يعيد قراءة الطابع المشترك ثم إما ينتظر مرة أخرى للمدة الباقية أو ينهي الترحيل. عندما يبلغ أحد الاتجاهين نهاية التيار، يُغلق جانب الإرسال في المقبس الآخر (إغلاق نصفي)؛ وعندما يفشل اتجاه، يُغلق المقبسان كلاهما فينتهي الاتجاه الآخر أيضًا. قبل تمرير الترقية عبر الوكيل، يحرر الاتصال محرك السياسة المخزَّن لديه مؤقتًا، فلا يُبقي اتصال قد يعيش ساعات محركًا قديمًا حيًّا.

الإيقاف

يوقف `Running::stop`، وكذلك `Drop` لـ `Running`، البرنامج الخدمي بترتيب ثابت. يرفع علم الإيقاف ويوقظ كل عامل قبول بالاتصال بالمستمع، ويضم العمال، ويغلق كل مقبس عميل مسجل ومقبس الخادم الأصلي الملحق به (`IdleTable::shutdown`)، ويوقف مجمع الخادم الأصلي، ويضم كل خيط اتصال، ويضم الجامع، ويغلق المقابس المُجمَّعة.

تصل SIGINT وSIGTERM إلى هذا التسلسل عبر Rust العادية. قبل أن يطلق البرنامج الخدمي أي خيط، يحجب الإشارتين كلتيهما (ويرث كل خيط لاحق القناع) ويخصص خيطًا واحدًا لـ `sigwait`، الذي يسجل الطلب ويوقظ الخيط الرئيسي فورًا. لا تجري أي شيفرة في معالج إشارة، ولا يستطلع أي خيط علمًا. تحمل وحدة الإشارات (`cmd/sibuna/src/signals.rs`) شيفرة `unsafe` الوحيدة في البرنامج الخدمي: تصريحات الدوال القليلة من مكتبة C التي لا تغلفها `std`.

> **التحرير عبر `Drop`** يجب أن تُكتب استدعاءات التنظيف الصريحة بعكس ترتيب الإعداد، ويجب أن تُقرأ كل عودة مبكرة في ضوئها. في Sibuna يحرر كل مورد نفسه في `Drop`: يوقف `Daemon` و`Running` الخادم، ويلغي `Lease` تسجيل اتصاله، ويعيد `Counted` مكانًا في حد الاتصالات، ويحرر `Finished` موضع مهمة. يُدرج المترجم كل تحرير على كل مسار خارج من النطاق، بما في ذلك الذعر الذي يفك المكدس عبره، فلا يستطيع أي مسار أن ينسى تحريرًا.

ما قسناه

يعرض الجدول Shield في وضع الوكيل العكسي على معالج Ryzen 9 5950X واحد: البرنامج الخدمي مثبت على أربع وحدات معالجة، و`wrk` بخيطين و64 اتصالًا على ثماني وحدات معالجة أخرى، وتشغيلات من خمس ثوانٍ، والوسيط من ثلاثة. الصفوف منسوخة من `benchmarks/results/product-latest.md`؛ ويضم الجزء الثامن السجل الكامل.

الجدول 7 Shield في وضع الوكيل العكسي، من `benchmarks/results/product-latest.md`

عبء العمل	الحالة	req/s	p50 / p99 (us)	زمن المعالج لكل طلب (us)
مقبول بجلسة، مُمرَّر عبر الوكيل	200	138,965	857 / 438	28.57

عبء العمل	الحالة	req/s	p50 / p99 (us)	زمن المعالج لكل طلب (us)
خاضع لتحدٍّ (الصفحة الوسيطة)	200	195,188	328 / 162	13.73
مسار ملفات ثابتة مسموح، مُمرَّر عبر الوكيل	200	145,673	746 / 415	27.27
مرفوض من WAF (مع جلسة)	403	209,507	306 / 151	12.17

يُبقي `wrk` 64 طلبًا قيد التنفيذ، لذلك تصطف الطلبات على المسارات المُمرَّرة المحدودة بالمعالج في انتظار وحدات المعالجة الأربع، ويكون p99 هناك ذيل المُجدوِل لا تكلفة طلب واحد. أما المولد ذو الحلقة المفتوحة، الذي يرسل بمعدل ثابت ويقيس زمن كل طلب من موعد إرساله المقصود، فيقيس زمن الاستجابة عند حمل معروض متساوٍ. للطلبات المقبولة في وضع الوكيل العكسي، يبلغ p99 قيمة 65 us عند 10,000 طلب في الثانية و100 us عند 100,000؛ وعند حمل معروض قدره 150,000 تتشبع وحدات المعالجة الأربع، فيتحقق 134,344 طلبًا في الثانية، وينمو الطابور إلى ثوانٍ (`benchmarks/results/latency-latest.md`). يخدم اتصال keep-alive دافئ كل قرار دون تخصيص في الكومة: يُحصي `cmd/sibuna/tests/steady_state.rs` تخصيصات البرنامج الخدمي، ولا يجد `crates/net/tests/steady_state.rs` أي تخصيص على مدى 500 طلب keep-alive مُمرَّر.

تمرين 5.1. يرسل عميل رفعًا من 200 KB. تتبَّع البايتات التي تعيش في المخزن ذي 64 KiB، والبايتات التي ترحِّلها `relay_body` عبر مخزن الرفع، ولماذا يجب أن يكتب الوكيل الرأس إلى الخادم الأصلي قبل الترحيل.

حالة متزامنة بلا تخصيص

الأهداف نحلل أقفال الشرائح، ومجموعة Robin Hood للحلول المُنفَقة، ومحدِّد GCRA، وجدول الحظر ذا الإصدارات، وحلقة الحوادث MPSC، والنشر بأسلوب القراءة والنسخ والتحديث (read-copy-update) لمحرك السياسة باستخدام `Arc` وذاكرة مؤقتة تُفحَص بالجيل.

يخصص كل مخزن في `sibuna-store` سعته الثابتة مرة واحدة عند بنائه، ولا يخصص بعد ذلك أبدًا ولا يُحجب على قفل عام. ولا يحتاج أي منها إلى منظف في الخلفية: تُستعاد الحالة المنتهية في مكانها.

الأقفال والتقسيم إلى شرائح

الأقسام الحرجة في جداول Sibuna بطول عشرات التعليمات، أقصر بكثير من رحلة ذهاب وإياب إلى futex في النواة. تُقسَم الجداول إلى 16 شريحة وفق تجزئة المفتاح، فتتنافس عمليتان باحتمال 1/16، وتقع كل شريحة على سطر ذاكرة مخبئية خاص بها، فلا تتشارك الأقفال المتجاورة السطر نفسه زيفًا (false sharing).

تستخدم الجداول `std::sync::Mutex`: عملية مقارنة وتبديل واحدة عند غياب التنافس، ودوران قصير أثناء عمل الحامل، ثم انتظار futex. البديل قفل دوار ذري ذو حالتين مع تلميح حلقة الدوران في حلقة الانتظار. طلب منا SID 0010 أن نختار بينهما بالقياس. يوفر القفل الدوار نحو نانوثانية لكل عملية بلا تنافس، لكنه يخسر تحت التنافس:

الجدول 8 إنتاجية أقفال الشرائح على Ryzen 9 5950X (أكتوبر 2026)

عبء العمل	`Mutex`	قفل دوار
مجموعة الحلول المُنفَقة، 16 خيطًا	10.8 M/s	8.6 M/s
مجموعة الحلول المُنفَقة، 64 خيطًا (أكثر من وحدات المعالجة)	11.2 M/s	6.9 M/s
GCRA على ستة عناوين ساخنة، 16 خيطًا	44.9 M/s	19.3 M/s
GCRA على ستة عناوين ساخنة، 64 خيطًا	37.4 M/s	6.5 M/s

يشغِّل خادم الحافة خيوط اتصالات أكثر من وحدات المعالجة، وقفل الدوران الذي يُستبَق حامله يجعل كل منتظر يحرق حصته الزمنية، لذلك نبقي على القفل (mutex). يُدخَل القفل المسموم بدل نشر التسميم: لا تذعر أي عملية وهي تحمل شريحة، وتبقى الشريحة صالحة بنيويًا حتى لو ذعرت إحداها.

مجموعة Robin Hood للحلول المُنفَقة

لا يدخل مجموعة الحلول المُنفَقة إلا الإثباتات المقبولة (الجزء الثالث). لها 16 شريحة، كل منها مصفوفة بعنونة مفتوحة من 4,096 موضعًا من النوع `Option<Entry>`، حيث المُدخَل `{ tag: [u8; 16], expires_at: u64, dist: u8 }`. يزيح الإدراج بأسلوب Robin Hood (Celis، 1986) أي شاغل أقرب إلى موضعه الأصلي من المفتاح الوارد، وهذا يُبقي أطوال السبر متقاربة بإحكام؛ ويتوقف البحث حالما يصادف مُدخَلًا أقرب إلى موضعه الأصلي مما هو إلى موضعه.

> **التنفيذ: `Table::insert`** تُدرج وسمًا مُنفَّقًا مع وقت انتهائه، مستعيدةً المواضع المنتهية في طريقها. في `crates/store/src/spent.rs` (الحزمة `sibuna-store`).

```
fn insert(
    &mut self,
```

```
    tag: &Tag,
    expires_at: u64,
    now: u64,
) -> Result<(), SpendError> {
    if !self.can_insert(tag, now) {
        return Err(SpendError::Full);
    }
    let mut carry = Entry { tag: *tag, expires_at, dist: 0 };
    let mut idx = Self::home(tag);
    while carry.dist < MAX_PROBE {
        let slot = &mut self.entries[idx];
        let Some(e) = slot else {
            *slot = Some(carry);
            return Ok(());
        };
        // الكتابة فوق شاغل منتهٍ آمنة فقط إذا لم تكن المسافة الجديدة
        // أصغر، فتظل المفاتيح التي تسبر متجاوزةً هذا الموضع تجتاز
        // اختبار التوقف المبكر.
        if e.is_expired(now) && carry.dist >= e.dist {
            *e = carry;
            return Ok(());
        }
        if carry.dist > e.dist {
            std::mem::swap(&mut carry, e);
            // المُدخَل المنتهي المُزاح يُسقَط ببساطة.
            if carry.is_expired(now) {
                return Ok(());
            }
        }
        idx = (idx + 1) % SHARD_CAPACITY;
        carry.dist += 1;
    }
    Err(SpendError::Full)
}
```

قاعدة الموضع المنتهي هي الجزء الدقيق: لا يجوز الكتابة فوق مُدخل منتهٍ في مكانه إلا بمفتاح يبعد عن موضعه الأصلي بقدر بعده على الأقل، وإلا فإن مفتاحًا لاحقًا يسبر متجاوزًا الموضع سيتوقف مبكرًا ويضيع. أما المُدخلات المنتهية التي تُزاح فتُسقط ببساطة. ولا حاجة إلى خيط منظف. يحاكي الإدراج أولًا مسار الإزاحة دون تغيير أي شيء (`can_insert`) ويتحقق من أن الإزاحة المحدودة يمكن أن تنجح دون فقد وسم حي.

> **اللامتغير** الإدراج المرفوض لا يطرد وسمًا حيًّا أبدًا. لذلك يُعثَر دائمًا على وسم أُنفِق ولم تنتهِ صلاحيته، ويمكن للرفض (`SpendError::Full`) أن يحجب الخدمة، لكنه لا يتيح أبدًا إعادة استخدام حل.

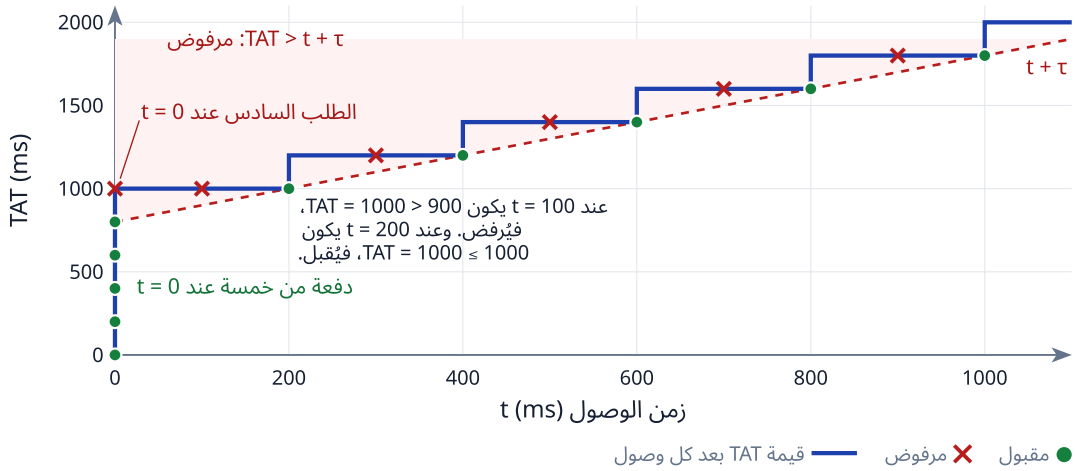
الموضع من النوع `Option<Entry>`. الموضع الفارغ لا يحوي وسمًا يُقرأ، فلا يوجد علم `occupied` قد ينسى قارئ فحصه، ولن تُترجَم `match` حتى تُعالَج الحالتان كلتاهما. ونوع الخطأ دقيق بالقدر نفسه: يمكن أن تفشل مجموعة الحلول المُنفَقة بطريقتين (`DoubleSpend` و`Full`)، و`SpendError` يسميهما بالضبط.

GCRA: تحديد المعدل بعدد صحيح واحد

خوارزمية Generic Cell Rate Algorithm (ATM Forum، 1996) هي صيغة الجدولة الافتراضية للدلو المثقوب. تخزن لكل عميل وقت وصول نظريًا واحدًا (TAT). مع فاصل الإصدار $T = \max(1, \lceil W/N \rceil)$ لحد موجب N في كل نافذة W، والتسامح مع الدفعات $\tau = (N-1)T$:

$$\text{arrival at } t \text{ conforms} \iff \text{TAT} \le t + \tau, \quad \text{then TAT} \leftarrow \max(\text{TAT}, t) + T.$$

مبرهنة (حد دلو الرموز) بعد أول وصول مطابق عند t_1، يحقق الوصول المطابق رقم k المتباينة $t_k - t_1 \ge (k-1)T - \tau$. ومن ثم تحتوي أي فترة طولها L على $N + \lfloor L/T \rfloor$ وصولًا مطابقًا على الأكثر: دفعة من N، ثم وصول واحد لكل T. لا يوجد أثر لحدود نافذة ثابتة يتجاوز الدفعة المحددة، ولا سجل لكل طلب.



الشكل 7 القبول في GCRA لحد قدره 5 في الثانية: دفعة من خمسة طلبات، ثم طلب كل 200 ms. يُقبل الطلب الواصل إذا كانت قيمة TAT التي يجدها على الخط $t + \tau$ أو تحته.

مثال محلول: خمسة طلبات في الثانية، حسابًا باليد لنأخذ $N = 5$ و$W = 1000$ ms، فيكون $T = 200$ ms و$\tau = 800$ ms، ولنفترض أن عميلًا كان خاملًا يرسل ستة طلبات عند $t = 0$. يجد الأول خلية فارغة: $\text{TAT} = \max(0, 0) = 0 \le 800$، فهو مطابق و$\text{TAT} \leftarrow 200$. تطابق الأربعة التالية بالطريقة نفسها وتترك $\text{TAT} = 1000$. يجد السادس $1000 > 0 + 800$ فيُرفض مع `Retry-After` قدره $\text{TAT} - \tau - t = 200$ ms. عند $t = 199$ يظل الطلب يجد $1000 > 999$ فيُرفض؛ وعند $t = 200$ يجد $1000 \le 1000$، فيطابق، ويترك $\text{TAT} = 1200$. يتحقق اختبار الوحدة `gcra_admits_exactly_the_burst_then_drains_at_the_emission_rate` في `crates/store/src/limiter.rs` من هذه الأعداد بالضبط.

التنفيذ: `arrive` تطبق وصولًا واحدًا على خلايا شريحة مقفولة وتعيد القرار وتأخير إعادة المحاولة والميزانية المتبقية. في `crates/store/src/limiter.rs` (الحزمة `sibuna-store`).

```
fn arrive(
    cells: &mut [Cell; SLOTS_PER_SHARD],
    key: u64,
    now_ms: u64,
    limits: Limits,
) -> Decision {
    let interval = limits.emission_interval();
    let tau = limits.burst_tolerance();
    let Some(cell) = locate(cells, key, now_ms, tau) else {
        return Decision {
            limited: true,
            capacity_exhausted: true,
            retry_after_ms: interval,
            remaining: 0,
        };
    };
    let tat = cell.tat_ms.max(now_ms);
    let horizon = now_ms.saturating_add(tau);
    if tat > horizon {
        return Decision {
            limited: true,
            capacity_exhausted: false,
            retry_after_ms: tat - tau - now_ms,
            remaining: 0,
        };
    }
    cell.tat_ms = tat.saturating_add(interval);
    // ... remaining = min(rate, floor((t + tau - TAT) / T) + 1)
}
```

تأخذ `check` الخاصة بالشريحة القفل وتستدعي `arrive` مع الخلايا المقفولة؛ ويحرر الحارس القفل عند عودة `check`. حجم الخلية 16 بايت (مفتاح وTAT) في 16 شريحة، في كل منها 512 موضعًا مع نافذة سبر من 16 موضعًا، أي نحو 128 KiB إجمالًا. الخلية التي يكون TAT فيها أقدم من $t - \tau$ قد استُنزفت وتُستعاد في الحال. وتُحسب ترويسة `Retry-After` من الحساب نفسه. نوافذ السبر المشبعة ترفض العملاء الجدد بدل طرد حالة حصص نشطة، والقرار يذكر ذلك (`capacity_exhausted`)، فلا يصعِّد المستدعي رفض التشبع إلى حظر. والمعدل المضبوط على صفر يرفض الطلبات. يخدم محدِّد واحد عدة قواعد: مفتاح الخلية تجزئة Wyhash لعنوان العميل مع بذرة هي نطاق القاعدة، فيكون للعميل نفسه في نطاقات مختلفة ميزانيات مستقلة.

جدول الحظر ذو الإصدارات

يستخدم الحظر 4,096 موضعًا بحقول ذرية للإصدار والمفتاح ووقت الانتهاء، بعنونة مفتوحة وفق تجزئة ذات مفتاح للعنوان المحظور، مع نافذة سبر من 8 مواضع. يحيط الكاتب، الذي يُسلسَل مع الكتاب الآخرين بقفل (mutex)، كل

تحديث بزيادتين في الإصدار؛ ولا يأخذ القراء أي قفل، ولا يقبلون إلا إصدارًا زوجيًا مستقرًا، ويعيدون المحاولة عند التغييرات المتزامنة.

> **التنفيذ: `BanTable::ban` و`BanTable::read_slot`** إحاطة الكاتب ولقطة القارئ المتسقة. في `crates/store/src/ban_table.rs` (الحزمة `sibuna-store`).

```
pub fn ban(&self, banned: &[u8], until: u64, now: u64) {
    let key = slot_key(banned);
    let _writers = self.lock_writers();
    let victim = &self.slots[self.victim(key, now)];
    // المفتاح ووقت الانتهاء قيمة منطقية واحدة: أحِط الاستبدال
    // بإصدار فردي حتى لا يجمع القراء مفتاحًا قديمًا مع وقت انتهاء
    // المفتاح التالي.
    victim.version.fetch_add(1, Ordering::SeqCst);
    victim.until.store(until, Ordering::SeqCst);
    victim.key.store(key, Ordering::SeqCst);
    victim.version.fetch_add(1, Ordering::SeqCst);
}

fn read_slot(&self, slot: &Slot) -> (u64, u64) {
    loop {
        let before = slot.version.load(Ordering::SeqCst);
        if before & 1 != 0 {
            std::hint::spin_loop();
            continue;
        }
        let key = slot.key.load(Ordering::SeqCst);
        let until = slot.until.load(Ordering::SeqCst);
        if before == slot.version.load(Ordering::SeqCst) {
            return (key, until);
        }
    }
}
```

يمنع الاتساق التسلسلي خلط هوية قديمة بوقت انتهاء بديل: ينتمي كل وصول إلى ترتيب كلي واحد، وهذا يُبقي الحجة بسيطة. لا يأخذ القراء قفلًا، لكنهم قد ينتظرون كاتبًا وليسوا خالين من الانتظار (wait-free). لهذه العمليات الذرية تكلفة حقيقية، تُقاس بصورة غير مباشرة في أداة اختبار HTTP بدل افتراض أنها مجانية. عندما تمتلئ نافذة السبر، يُكتب فوق موضع منتهٍ، وإلا فوق الموضع الذي ينتهي أولًا. ترفع `clear` كل حظر لكنها تترك المفاتيح في أماكنها، لأن موضعًا مُفرَّغًا قد يخفي مفتاحًا متصادمًا يقع لاحقًا في سلسلة سبر.

حلقة الحوادث MPSC

عندما يرفض وضع Shield طلبًا أو تنطلق المصيدة، يُنسخ الحادث إلى سجل ثابت من نحو 1.3 KB (عنوان العميل، وUser-Agent، والطريقة، والمسار، والفئة، والحمولة، وكل منها مقطوع عند سعته) ويُدفع إلى حلقة محدودة متعددة المنتجين وحيدة المستهلك (تصميم Vyukov الذي يسم المواضع بأرقام تسلسل، 512 موضعًا، `BoundedQueue<IncidentRecord, 512>`). المنتجون خيوط الاتصالات؛ والمستهلك خيط التخزين. الحلقة الممتلئة

تُسقط أحدث سجل بدل حجب استجابة، وهي سياسة فقد صريحة لا ضمان ديمومة. ويجعل عداد الإسقاط هذا الفقد قابلًا للملاحظة.

```
/// ينسخ `incident` إلى الحلقة؛ والحلقة الممتلئة تُسقطه وتُحصي الإسقاط.
pub(crate) fn record(&self, incident: &Incident<'_>) {
    if self.queue.push(IncidentRecord::from_incident(incident)).is_err() {
        self.metrics.incidents_dropped.bump();
    }
}
```

بعد نقل سجل إلى دفعة معلقة، يحتفظ به خيط التخزين، مع الصفوف المُعدَّة، حتى يُؤكَّد الإيداع (commit)؛ ولا يتخلص منه بصمت عند خطأ في قاعدة البيانات، وإعادة المحاولة تكتب الدفعة نفسها.

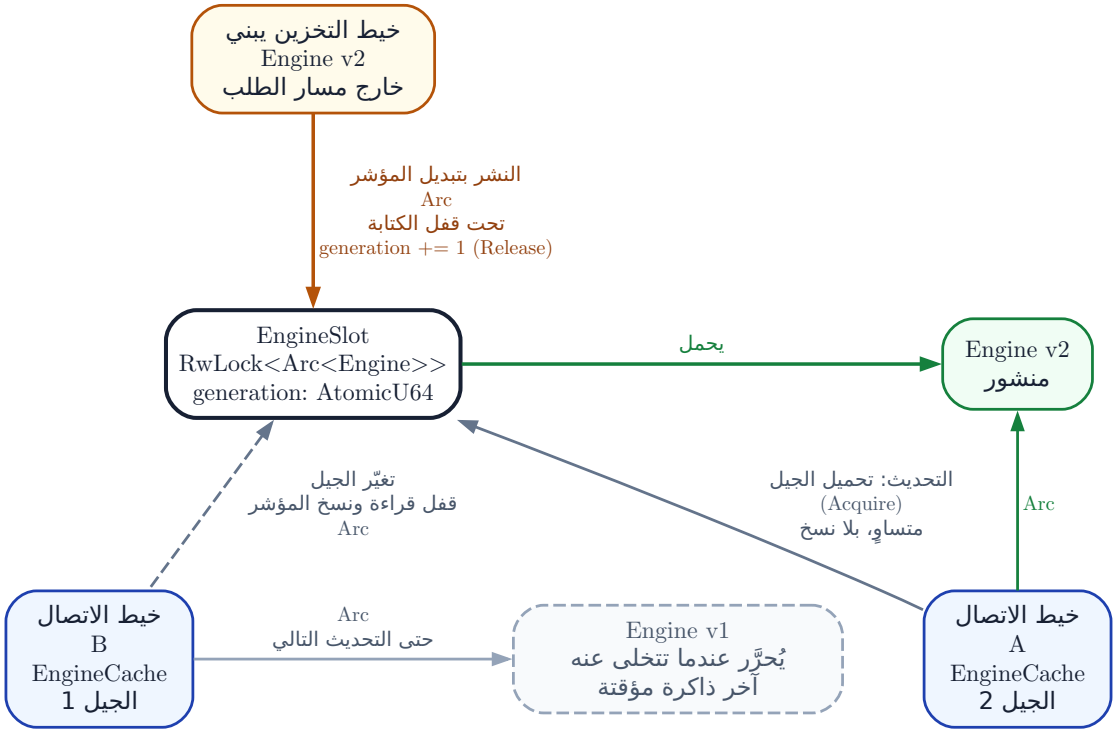
يحمل كل موضع رقم تسلسل يخبر المنتجين ما إذا كان الموضع حرًّا ويخبر المستهلك ما إذا كان ممتلئًا، فلا يأخذ أي من الجانبين قفلًا. لا يمكن كتابة حلقة كهذه بلغة Rust الآمنة: يلمس المنتجون والمستهلك مواضع مختلفة من مصفوفة واحدة في الوقت نفسه، وبروتوكول التسلسل وحده، لا نظام الأنواع، هو الذي يثبت أي خيط يجوز له لمس أي موضع. لذلك تحفظ الوحدة `crates/store/src/ring.rs` قيمها في مواضع `UnsafeCell<MaybeUninit<T>>`، وتسمي كل كتلة `unsafe` خطوة البروتوكول التي تسوِّغها:

```
if diff == 0 {
    // Relaxed: لا تفصل CAS إلا في أي منتج يملك الموضع `pos`.
    match self.tail.0.compare_exchange_weak(
        pos, pos.wrapping_add(1), Ordering::Relaxed, Ordering::Relaxed,
    ) {
        Ok(_) => {
            // SAFETY: منحت CAS هذا الخيطَ الموضع `pos`، و
            // `sequence == pos` يدل على أن الموضع فارغ؛ ولا يستطيع منتج آخر
            // المطالبة بـ `pos`، ولا يقرؤه المستهلك قبل التخزين بترتيب
            // Release أدناه.
            unsafe { (*slot.value.get()).write(value) };
            // Release: ينشر القيمة المكتوبة للمستهلك.
            slot.sequence.store(pos.wrapping_add(1), Ordering::Release);
            return Ok(());
        }
        Err(current) => pos = current,
    }
} else if diff < 0 {
    return Err(value);
}
```

اصطلاحان من اصطلاحات الحلقة يُعبَّر عنهما بالأنواع لا بالتعليقات. على الحلقة الممتلئة تعيد `push` القيمة `Err(value)`، فتعيد السجل إلى المستدعي ليقرر أيُحصيه أم يُسقطه؛ أما `false` المجردة فكانت ستضيعه. للحلقة مستهلك واحد، و`pop` تأخذ `&mut self`، ويجب على الشيفرة المشتركة أن تطالب أولًا بـ `Consumer` الفريد، فالمستهلك الثاني لا يُترجَم أو يُرفض في وقت التشغيل.

نشر المحرك بأسلوب القراءة والنسخ والتحديث

يُعاد بناء محرك السياسة خارج المسار الساخن عند تغير السياسات. يجب ألا ترى الطلبات أبدًا محركًا نصف مبني، ويجب ألا يُحرَّر محرك أبدًا ما دام طلب ما يقرؤه. ينشر Sibuna كل محرك بوصفه `Arc<Engine>` في `EngineSlot`، ويقرؤه كل اتصال عبر `EngineCache`:



الشكل 8 نشر المحرك. ينشر خيط التخزين محركًا كاملًا ويقدِّم الجيل؛ ويفحص كل اتصال الجيل بتحميل acquire واحد، ولا ينسخ Arc المنشور إلا بعد نشر جديد. يُحرَّر المحرك القديم عندما تتخلى عنه آخر ذاكرة مؤقتة تحمله.

> **التنفيذ: `EngineSlot::publish` و`EngineCache::refresh`** نشر محرك كامل، ورؤية اتصال واحد له. في `cmd/sibuna/src/slot.rs` (الحزمة `sibuna`).

```rust
pub struct EngineSlot {
    current: RwLock<Arc<Engine>>,
    generation: AtomicU64,
}

pub fn publish(&self, engine: Engine) {
    let fresh = Arc::new(engine);
    let previous = {
        let mut current =
            self.current.write().unwrap_or_else(PoisonError::into_inner);
        std::mem::replace(&mut *current, fresh)
```

```
    };
    self.generation.fetch_add(1, Ordering::Release);
    // يُسقَط المحرك القديم خارج القفل (إذا كان هذا آخر حامل له).
    drop(previous);
}

pub struct EngineCache {
    generation: u64,
    engine: Option<Arc<Engine>>,
}

pub fn refresh(&mut self, slot: &EngineSlot) -> &Engine {
    let generation = slot.generation();
    if self.generation != generation {
        // إذا وقع نشر بين التحميل أعلاه وهذا النسخ، وُسم محرك أحدث
        // بجيل أقدم؛ والتحديث التالي يحمِّل من جديد ببساطة.
        self.generation = generation;
        self.engine = None;
    }
    self.engine.get_or_insert_with(|| slot.load())
}
```

يُخزَّن الجيل بترتيب release بعد أن يستقر المحرك في مكانه، ويُحمَّل بترتيب acquire: القارئ الذي يرى جيلًا جديدًا يرى أيضًا المحرك المخزَّن قبله. السباق المذكور في التعليق غير ضار. إذا وقع نشر بين تحميل القارئ للجيل ونسخه، يخزن القارئ المحرك الأحدث مؤقتًا تحت رقم الجيل الأقدم، ويلاحظ تحديثه التالي عدم التطابق فيحمِّل مرة أخرى. قد يتأخر القارئ محركًا واحدًا لطلب واحد؛ لكنه لا يستطيع أبدًا أن يرى محركًا نصف مبني، لأن الموضع لا يُخزَّن فيه إلا محركات كاملة.

> **اللامتغير** كل محرك يقرؤه طلب كامل، ولا يُحرَّر أي محرك ما دام أي طلب أو ذاكرة مؤقتة لاتصال تحمله. يصح الأول لأن المحرك يُبنى بالكامل قبل أن تخزنه publish؛ ويصح الثاني لأن كل قارئ يحمل Arc، ولا يُسقط المحرك إلا مع آخر Arc له.

لماذا ذاكرة مؤقتة وليس قفلًا ببساطة؟ طلب منا SID 0010 أن نقيس تكلفة القراءة مع 16 خيطًا ونختار بين RwLock<Arc<Engine>> وبديل بلا أقفال بلغة Rust الآمنة. القراءة عبر القفل تأخذ كلمة القفل وعداد المراجع، وهما سطرا ذاكرة مخبئية تكتب فيهما كل نواة. أما الجيل فيُكتب مرة واحدة لكل نشر ويُقرأ فقط في غير ذلك، فتحتفظ كل نواة بنسختها منه:

الجدول 9 تكلفة قراءة واحدة للمحرك، Ryzen 9 5950X، و16 خيطًا تقرأ في حلقة (أكتوبر 2026)

القارئ	خيط واحد	16 خيطًا
حارس القراءة لـ RwLock فقط	4.2 ns	752 ns
قراءة RwLock ونسخ Arc	7.6 ns	833 ns
فحص الجيل مع Arc مخزَّن مؤقتًا	0.8 ns	1.8 ns

تحت التنافس تكون القراءة المخزَّنة مؤقتًا أرخص من القفل بأكثر من 400 مرة، لذلك يستخدم مسار الطلب الذاكرة المؤقتة، ولا يستخدم `RwLock` إلا على المسار البطيء للتحديث. القياس هو الاختبار المُتجاهَل `read_cost_under_contention` في `cmd/sibuna/src/slot.rs`. ويحدِّث كل استخدام على مسار الطلب الذاكرة المؤقتة، فيرى الطلب المحرك الحالي عند تلك الخطوة. والمحرك نفسه صغير: 209,052 بايت، أي نحو 204 KiB، من جداول مضغوطة في الكومة (`engine_bytes` في `benchmarks/results/primitives-latest.json`)، مع سعة لـ 128 قاعدة، وأسماء قواعد من 128 بايت، و8,192 عقدة في الشجرة.

لماذا `Arc` وليس عدادات القراء يحتفظ البديل ذو الذاكرة الثابتة بموضعين للمحرك، لكل منهما عداد قراء وساحة تخصيص خاصة به. يثبِّت الطلب موضعًا بزيادة عداده ثم إعادة فحص مؤشر الموضع، ويجب أن تكون العمليتان الذريتان كلتاهما متسقتين تسلسليًا لسد السباق الذي يبدِّل فيه كاتب بين تحميل القارئ وزيادته. يبدِّل الباني المؤشر ثم يدور حتى ينصرف قراء الموضع القديم، قبل أن يعيد ضبط ساحة ذلك الموضع ويبني فيها من جديد. لذلك فإن القارئ الذي يحتفظ بموضع طوال إعادة البناء يعطِّل الباني، وكان الخادم الأصلي البطيء سيعطل النشر التالي ما لم يحرر كل طلب موضعه قبل بدء عمليات الإدخال والإخراج في الوكيل.

عد المراجع في `Arc` لا يحتاج إلى شيء من هذا. يبني خيط التخزين `Engine` جديدًا من كل لقطة وينشره؛ لا يُعاد استخدام أي شيء، فلا موضع احتياطيًا، ولا إعادة ضبط لساحة، ولا دوران: `publish` لا تنتظر القراء أبدًا. يعيش المحرك القديم ببساطة حتى يتخلى عنه آخر حامل له، والخادم الأصلي البطيء لا يستطيع أن يؤخر إلا تلك اللحظة، لا النشر التالي أبدًا. ينسخ الحكم اسم القاعدة المطابقة إلى مخزن من 128 بايت لينهي استعارة: يستعير القرار المحرك عبر الذاكرة المؤقتة للاتصال، ويحتاج مسار الاستجابة سياق الاتصال كاملًا قابلًا للتعديل. والاتصال المُرقّى، الذي قد يعيش ساعات، يحرر محركه المخزَّن مؤقتًا قبل تمريره عبر الوكيل.

تمرين 5.2. باستخدام مبرهنة GCRA، احسب أكبر عدد من الطلبات يستطيع عميل واحد تمريره في أول 3 ثوانٍ بعد خموله، مع `--rate-limit 100 --rate-window 10`. قارن ذلك بعداد نافذة ثابتة من 10 ثوانٍ وبسجل نافذة منزلقة حقيقي.

اشرح اللامتغير. اشرح، بمثال من ثلاثة مواضع، لماذا لا يجوز الكتابة فوق مُدخل Robin Hood منتهٍ إلا بمفتاح مسافته كبيرة بالقدر نفسه على الأقل.

الجزء السادس

آلات الحالات، وأشجار البادئات، والجدار الناري الدلالي

نفحص الخوارزميات التي تتيح التصنيف في أقل من ميكروثانية: آلة *Aho-Corasick* الموسومة، وشجرة *radix* لعناوين *IPv4/IPv6*، ومقسِّمات *WAF* الدلالي القائمة على فئات البايتات، ومحرك السياسة التصريحي مع احتساب الدرجات بـ *WEIGH*.

مطابقة أنماط متعددة بمرور واحد

الأهداف بنهاية هذا الفصل، ينبغي أن نتمكن من اشتقاق ضمان الزمن الخطي لآلة Aho-Corasick، وشرح تخطيط الجدول الكثيف وطي حالة الأحرف عبر فئات البايتات، وتعليل عدم اعتماد محرك SIMD للنصوص الحرفية.

Aho-Corasick بوصفها آلة DFA كثيفة

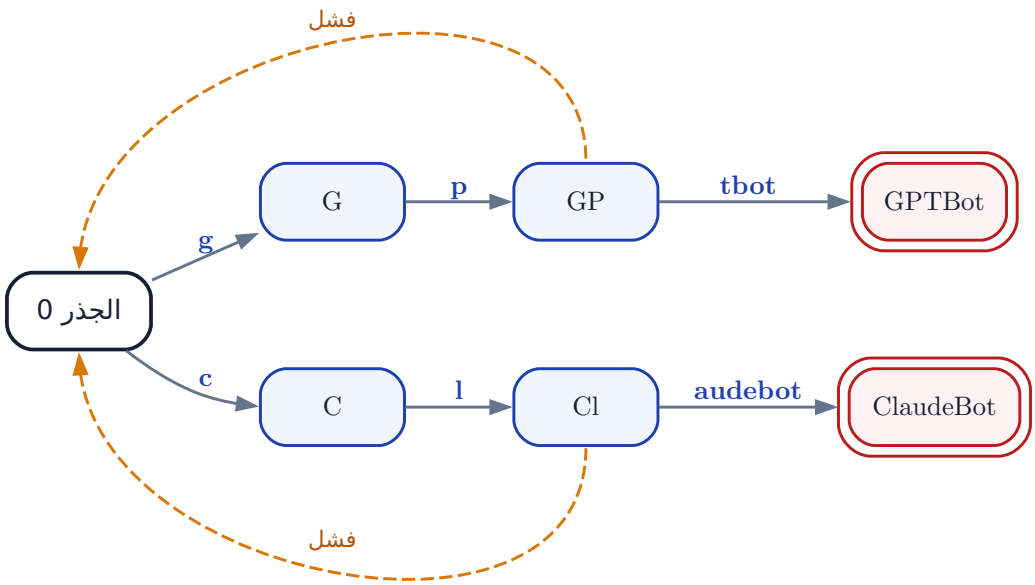
لدى Sibuna مجموعتان من الأنماط الحرفية: نحو أربعين توقيعًا للروبوتات ونحو مئة توقيع للهجمات. ومسح حقل بمقارنته بـ N من الأنماط نمطًا نمطًا يكلّف $O(NM)$؛ أما آلة Aho و Corasick (1975) فتمسحه مرة واحدة في $O(M)$ بعد بناء شجرة بادئات مع روابط الفشل. يدمج Sibuna روابط الفشل في جدول الانتقال وقت البناء، فيصير المسح تحميلًا واحدًا من الجدول لكل بايت من المدخل، بلا تفرعات تتوقف على مجموعة الأنماط. وتوجد الآلة في الحزمة (crate) `sibuna-patterns`:

```
/// يتقدم من `state` (إزاحة صف) عبر `byte`.
fn step(&self, state: usize, byte: u8) -> usize {
    self.table[state + usize::from(self.classes[usize::from(byte)])] as usize
}

pub fn find_first(&self, haystack: &[u8]) -> Option<Match> {
    let mut state = 0;
    for (index, &byte) in haystack.iter().enumerate() {
        state = self.step(state, byte);
        if state >= self.first_match {
            let id = self.first_output_in(self.match_state(state),
self.all_tags)?;
            return Some(self.report(id, index + 1));
        }
    }
```

```
    None
}
```

تُخزَّن الحالة على هيئة إزاحة صفها في جدول مسطح واحد، فتكون الخطوة عملية جمع واحدة وتحميلًا واحدًا. وتُرقَّم الحالات بحيث تأتي كل حالة مطابقة بعد سائر الحالات؛ فيصير كشف المطابقة مقارنة واحدة مع `first_match`، بلا جدول ثانٍ يُحمَّل في الحلقة الداخلية.



الشكل 9 حالات شجرة البادئات وحواف الفشل لتوقيعين

ميزانية حالات الآلة خيار يُضبط وقت التشغيل. يحجز محرك السياسة على الأكثر جذرًا واحدًا مضافًا إليه مجموع أطوال التوقيعات كلها، وهو حد أعلى آمن لعدد حالات شجرة البادئات. ويتجنب ذلك الجداول الثابتة المفرطة الحجم التي عُرفت سابقًا، مع الإبقاء على انتقال كثيف لكل فئة بايت. يحمل كل نمط وسم فئة صغيرًا (من 0 إلى 63، فتكون مجموعة الوسوم كلمة واحدة من 64 بت). ويستخدم طي حالة الأحرف جدول فئات بايتات من 256 مدخلًا يُبنى مع الآلة: تشترك حالتا الحرف الواحد في فئة واحدة، وتشترك البايتات التي لا تظهر في أي نمط في فئة واحدة، فيكون عرض صف الجدول عدد بايتات الأنماط المختلفة زائد واحد بدل 256. ويسجل الجزء الثامن أحجام المحرك الدقيقة والمقارنة مع البحث المتسلسل عن السلاسل الفرعية، وهو بحث يستخدم الأنماط نفسها ويكتفي بأي مطابقة.

التعبيرات النمطية ومحركات SIMD والاختيار بينها

وازنّا بين ثلاث عائلات لمرحلة التوقيعات:

- التعبيرات النمطية ذات التراجع (على طريقة مجموعات القواعد ذات التراجع) تشغّل مئات الأنماط لكل طلب وتخصص ذاكرة لكل مطابقة؛ وهي أبطأ الخيارات وأكثرها عرضة للإنذارات الكاذبة.
- محركات *SIMD* للنصوص الحرفية (Hyperscan وVectorscan) ترشّح مسبقًا كتلًا من 16 إلى 32 بايتًا بأقنعة أنصاف بايت قائمة على الخلط (خوارزمية "Teddy")، ثم تتحقق من المرشحين بآلات حالات. وهي تتفوق على

المدخلات الطويلة مع مجموعات حرفية كبيرة. وثمنها تبعية C++ بحجم عدة ميغابايتات، وقاعدة بيانات أنماط تُبنى وقت الترجمة، وغياب هدف WebAssembly، ولا ميزة لها على مدخلات من مئة بايت: يُمسح الحقل كله في زمن أقل من إخفاق واحد في الذاكرة المخبئية.

- آلة *Aho-Corasick* كثيفة مع مقسِّمات بنيوية صغيرة، على نهج libinjection، تعطي سلوكًا خطيًا دقيقًا في جدول ثابت، ولا تحتاج إلى شيء خارج الملف التنفيذي.

حقول Sibuna قصيرة، لذا تفوز العائلة الثالثة في الذاكرة والسرعة معًا؛ ويسجل الجزء الثامن نحو 1.56 ns لكل بايت على جسم بحجم 8 KiB (يستغرق المسح الدلالي 12,784.3 ns، `8kb_body_semantic_scan` في `benchmarks/results/primitives-latest.md`). هذا هو عبء العمل الوحيد الذي قد يفيد فيه ترشيح SIMD مسبق، ويبقيه SID 0006 بندًا مفتوحًا.

شجرة radix لعناوين IPv4 وIPv6

> **الأهداف** نتتبع بحثًا عن أطول بادئة في الشجرة ذات 128 بت، ونشرح اختصار جذر IPv4، ونرى كيف تؤدي الشجرة أيضًا دور مخزن السمعة الذي يغذيه العنقود.

شجرة ثنائية واحدة على مفاتيح من 128 بت تخدم العائلتين: تُدرَج بادئات IPv6 كما هي، وتُحوَّل بادئات IPv4 إلى `::ffff:0:0/96`، فتصير `/24` مسارًا بعمق 120. وتُحفظ العقد في `Vec` مسطح واحد مفهرس بـ `u32`، والفهرس صفر هو الجذر و"لا عقدة فرعية" معًا، فيكون البحث 128 تحميلًا متعاقبًا على الأكثر، يعتمد كل منها على سابقه، بلا تتبع للمؤشرات. والشجرة هي `PrefixTrie` في الحزمة `sibuna-cidr`.

المرور بمستويات البادئة المحوَّلة الستة والتسعين في كل بحث عن IPv4 يكلف 96 تحميلًا متعاقبًا قبل الوصول إلى أول بت من عنوان IPv4. لذا تسجل الشجرة العقدة التي على العمق 96 عند التهيئة، وتبدأ منها عمليات البحث عن IPv4:

```
let (mut node, mut depth, mut best) = if is_ipv4_mapped(key) {
    let start = self.ipv4_root;
    let value = self.nodes[start as usize].value.as_ref();
    (start, IPV4_DEPTH, value.or(self.nodes[0].value.as_ref()))
} else {
    (0, 0, self.nodes[0].value.as_ref())
};
```

يكلف تصنيف عنوان IPv4 مدة 32.0 ns وعنوان IPv6 مدة 79.3 ns (`ipv4_cidr_classification` و `ipv6_cidr_classification` في `benchmarks/results/primitives-latest.md`). يستشير المحرك الشجرة مرتين: أحكام `deny`/`allow` النهائية قبل جدول القواعد، كي يتغلب الحظر على مستوى العنقود على كل قاعدة، وأحكام `challenge` بعده.

الجدار الناري الدلالي

> **الأهداف** نفهم المسح المسبق لفئات البايتات، وآلة التوقيعات الموسومة، ومقسِّم SQL (tokenizer) مع كشف العبارات الصحيحة دائمًا (tautology)، وفحوص بنية HTML والصدفة، والتطبيع، وانضباط الإنذارات الكاذبة الذي يقتضي أن يمر طلب المتصفح الحقيقي دون مساس.

درس في الإنذارات الكاذبة

طابقت النسخة الأولى من WAF قائمة مسطحة من السلاسل الفرعية مع كل ترويسة. احتوت القائمة على */*، وكل متصفح يرسل `Accept: text/html,...,*/*;q=0.8`، فصُنِّف كل متصفح حقيقي على أنه حقن SQL وتلقى 403. وقد شكّل هذا الدرس التصميم: لا يجوز للكاشف أن يُطلق إنذارًا إلا بناءً على البنية، وصارت مجموعة الاختبارات تمرر عبر المحرك طلب Chrome كاملًا، بما فيه نص تعليق يحتوي على `select` و`--` وفاصلة عليا، وتتأكد من أنه يخرج نظيفًا.

فئات البايتات في مرور واحد

يُمسح كل حقل مفحوص مرة واحدة لجمع فئات البايتات الموجودة فيه: علامات الاقتباس، و=، و>، وفواصل الصدفة، و%، وNUL، ومحفزات التطبيع، والمسافات المزدوجة. ويستشير كل كاشف هذه الفئات بدل إعادة المسح، ولا يجري المرور الثاني على النص المطبَّع إلا عند رؤية محفز. والمسح بحث في جدول من 256 مدخلًا لكل بايت، في وحدة WAF من `sibuna-policy`. تُبقي أربعة مراكِمات سلاسل OR قصيرة، ولا يُبحث عن زوجي البايتات (`/*` ومسافتين متجاورتين) إلا إذا لم يكن أي بايت مفرد قد طلب التطبيع بالفعل:

```
pub(crate) fn scan(text: &[u8]) -> Classes {
    let (chunks, remainder) = text.as_chunks::<8>();
    // تُبقي أربعة مراكِمات سلاسل OR قصيرة، فتصدر التحميلات متتالية بلا انتظار.
    let mut lanes = [0u16; 4];
    for chunk in chunks {
        lanes[0] |= TABLE[usize::from(chunk[0])] | TABLE[usize::from(chunk[4])];
        lanes[1] |= TABLE[usize::from(chunk[1])] | TABLE[usize::from(chunk[5])];
        lanes[2] |= TABLE[usize::from(chunk[2])] | TABLE[usize::from(chunk[6])];
        lanes[3] |= TABLE[usize::from(chunk[3])] | TABLE[usize::from(chunk[7])];
    }
    let mut bits = lanes[0] | lanes[1] | lanes[2] | lanes[3];
    for &byte in remainder {
        bits |= TABLE[usize::from(byte)];
    }
    // لا تهم الأزواج إلا ما دام أي بايت مفرد لم يطلب التطبيع بعد.
    if bits & CANONICAL == 0
        && ((bits & SPACE != 0 && has_pair(text, b' ', b' '))
            || (bits & (SLASH | STAR) == SLASH | STAR && has_pair(text, b'/',
b'*')))
    {
        bits |= CANONICAL;
    }
```

```
    Classes { bits }
}
```

الكواشف

- التوقيعات. توجد التوقيعات القوية لكل الفئات في آلة موسومة واحدة؛ والإصابة رفض deny نهائي باسم waf:sqli أو waf:xss أو waf:path-traversal أو waf:rce.
- حقن *SQL*. يبحث ماسح كلمات بمرور واحد عن كل كلمة أبجدية رقمية (سبعة بايتات على الأكثر) في جدول الكلمات المفتاحية، وعند and / or يفحص وجود عبارة صحيحة دائمًا من الشكل literal = literal؛ فيكون id=1 or 1=1 نهائيًا. وفيما عدا ذلك فبايت الاقتباس شرط لازم، ويجب أن تبلغ درجة الأدلة 4 على الأقل من: علامة الاقتباس (1)، وكلمتين مفتاحيتين على الأكثر (2)، وعلامة تعليق (1)، و= (1). ويكون admin'-- نهائيًا بمفرده. أما "it's a group order from the shop" فدرجته 3 فيمر.
- البرمجة النصية عبر المواقع *(XSS)*. يتطلب <؛ ويُطلق إنذارًا عندما يجتمع وسم يمكنه استضافة نص برمجي مع سمة =on<event> في موضع السمات. ولا تُحتسب onboarding=1.
- حقن الأوامر. يتطلب فاصلًا؛ ويُطلق إنذارًا عندما يلي ; أو | أو && أو)$ أو علامةَ الاقتباس المائلة (backtick) أو سطرًا جديدًا اسمُ أمر معروف. ويمر status ok; done | next.
- اجتياز المسارات. توقيعات (/..، وfg2%..، وetc/passwd/، وid_rsa، ...) إضافة إلى NUL و%00.

تُتخطى الترويسات البنيوية التي تحتوي قواعدها مشروعًا على علامات الاقتباس والنجوم (Accept*، و Content-Type، و*-Sec، و*-If، ...)؛ ويُفحص المسار، والاستعلام، وUser-Agent، والترويسات الأخرى، وأول 8 KB من الجسم، وتُطبَّع الحقول التي تحتوي على ترميزات النسبة المئوية، أو علامات الجمع، أو فواتح التعليقات، أو مسافات قابلة للطي في مخزن مؤقت على المكدس بحجم 8 KB ثم تُفحص مجددًا، فيتحول 1%27/**/UnIoN/**/SeLeCt و/e%252e%252f إلى التوقيعات الخام.

التكلفة

تعتمد تكلفة الفحص على الحقول والبايتات المقدمة إلى المحرك، لا على عدد الطلبات وحده. يعرض فصل قياسات الأداء حالات منفصلة لـ Gate وShield وجسم بحجم 8 KB. حد 8 KB حد للفحص، وليس ادعاءً بأن بقية الملف المرفوع آمنة. يجب على وكيل الدخول الذي يستخدم forward-auth أن يمرر معلومات الطلب ذات الصلة؛ فالبايتات المحذوفة لا يمكن أن تفحصها هذه العملية.

محرك السياسة التصريحي

> **الأهداف** نقرأ نموذج القواعد، وترتيب التقييم، واحتساب الدرجات بـ WEIGH مع العتبات، وقواعد صياغة سياسة JSON.

القاعدة اقتران لمعايير اختيارية (نمط المسار، ونمط User-Agent، وما يصل إلى أربع ترويسات، وما يصل إلى ثماني كتل CIDR لـ IPv4/IPv6، وبصمة JA4 واحدة أو بادئة لها) مع إجراء `ALLOW` أو `DENY` أو `CHALLENGE` أو `WEIGH`، وصعوبة اختيارية لكل قاعدة ببتات العمل والخوارزمية، و`weight` ذي إشارة. ترتيب التقييم:

1. WAF الدلالي (Shield فقط): المخالفة `deny` نهائي.
2. شجرة السمعة: بادئات `deny` أو `allow` نهائية.
3. القواعد بالترتيب: تعيد أول مطابقة نهائية؛ وتتراكم مطابقات `WEIGH`.
4. الدرجة: المجاميع السالبة `allow`؛ والمجاميع $\text{deny} \geq 40$؛ والمجاميع $\text{challenge} \geq 10$ مع بت عمل إضافي لكل 5 نقاط فوق العتبة، بحد أقصى ستة.
5. مسارات التجاوز الثابتة، ثم أحكام `challenge` من الشجرة، ثم آلة الروبوتات.
6. الإجراء الافتراضي، `challenge` ما لم يقل ملف السياسة غير ذلك.

تقبل القواعد المدمجة `/.well-known/*` و`/favicon.ico` و`/robots.txt` و`/__sibuna/*`، وترفض عملاء `CF-Worker` و`Amazonbot`، وتتحدى كل User-Agent يحمل `Mozilla`: فتزوير سلسلة المتصفح لا يكلف شيئًا، وقبولها يجعل البوابة شكلية.

```
{
  "default_action": "CHALLENGE",
  "waf": true,
  "thresholds": { "challenge_at": 10, "deny_at": 40, "bits_step": 5 },
  "ip_rules": { "10.0.0.0/8": "ALLOW", "2001:db8::/32": "DENY" },
  "rules": [
    { "name": "internal-api", "path": "/api/*", "headers": { "X-Api-Key": ".*" },
    "action": "ALLOW" },
    { "name": "checkout", "path": "/checkout/*", "action": "CHALLENGE",
      "challenge": { "difficulty": 20, "algorithm": "posw" } },
    { "name": "headless", "user_agent": "Headless", "action": "WEIGH", "weight":
30 },
    { "name": "no-language", "headers": { "Accept-Language": "" },
    "action": "WEIGH", "weight": -5 }
  ]
}
```

تتبع تفصيلي: هجوم مرمَّز مع جلسة صالحة

لننظر في الطلب `GET /search?q=%2527%2520OR%25201%253D1` من عميل يحمل جلسة صالحة. يفصل محلل HTTP أولًا المسار عن الاستعلام دون نسخ بايتاتهما. ثم تُطبَّق الحدود المحلية، ثم يفحص Shield الطلب. تَسِم علامة النسبة المئوية الحقل للتطبيع. فيكشف مرور فك ترميز أول `%27%20OR%201%3D1`؛ ويكشف مرور ثانٍ `' OR 1=1 .`

يرى المقسِّم الآن علامة اقتباس، والكلمة المفتاحية `OR`، ومعاملين حرفيين متساويين. والرفض نتيجة سياسة نهائية، ولا تحوله الجلسة الصالحة إلى قبول: ففحص الجلسة لا يحسم إلا قرار التحدي. ينسخ خطاف التخزين سجل حادث محدود الحجم؛ ويجري SQL لاحقًا. ولا يحتاج مسار الاستجابة إلى الاحتفاظ بلقطة المحرك أثناء خمول الاتصال بالخادم الأصلي.

تمرين 6.1. استبدل الحمولة بـ `O'Reilly` ثم بـ `order=1`. لماذا ينبغي ألا يثبت أي منهما وحده حقن SQL؟ أعطِ مثالًا على إنذار كاذب قد ينتجه كاشف يعتمد على السلاسل الفرعية وحدها.

تمرين 6.2. اكتب مدخلات جدول فئات البايتات اللازمة لإضافة كاشف لحقن LDAP (أنماط مثل `)(|(`)، واشرح أي بتات الفئات الحالية يمكنه إعادة استخدامها.

اشرح اللامتغير. اشرح لمراجِع لماذا كان `Accept: */*` إنذارًا كاذبًا، وما الخاصية البنيوية التي يشترطها كاشف SQL الجديد، وكيف تتقاسم الآلة والمقسِّم العمل.

الجزء السابع

محرك إثبات العمل في المتصفح

ننظر في وحدة *WebAssembly* التي تُترجَم من حزمة المتحقِّق الخاصة بالخادم نفسه، وفي بروتوكول *Web Worker*، وفي المُبرهِنات البديلة المكتوبة بلغة *JavaScript*، وفي الصفحة الوسيطة.

مصدر واحد وهدفان

> **الأهداف** بنهاية هذا الفصل، ينبغي أن نتمكن من شرح كيف تُبنى وحدة المتصفح من حزمة `sibuna-pow` الخاصة بالخادم نفسه، وتسمية الدوال المصدَّرة، ووصف الذاكرة التي يحتاجها مُبرهِن العمل المتسلسل في تبويب المتصفح.

ربط البناء

يترجم برنامج الحل في المتصفح المصادر نفسها التي يتحقق بها الخادم. `cmd/sibuna-pow-wasm` حزمة مكتبة `no_std` من النوع `cdylib`، تُبنى للهدف `wasm32-unknown-unknown`، ولا تعتمد إلا على حزمتي مساحة العمل `sibuna-pow` و`sibuna-crypto`:

```
# cmd/sibuna-pow-wasm/Cargo.toml
[lib]
crate-type = ["cdylib", "rlib"]

[dependencies]
sibuna-crypto = { workspace = true }
sibuna-pow = { workspace = true }

# Cargo.toml جذر مساحة العمل): تُبنى الوحدة بأصغر حجم ممكن
[profile.release.package.sibuna-pow-wasm]
opt-level = "z"
debug = false
strip = true
```

يضمِّن البرنامج الخدمي الوحدة في ملفه. يشغِّل سكربت البناء (`cmd/sibuna/build.rs`) أمرًا متداخلًا يكتب في `cargo rustc -p sibuna-pow-wasm --target wasm32-unknown-unknown --profile wasm --compress-relocations`. ولا يقفل البناء المتداخل إلا مجلدات مجلد الأهداف لمساحة العمل (`target/wasm32-unknown-unknown/wasm/`). ويتلقى الربط النهائي الخيار ملف تعريف البناء `wasm`، فلا ينتظر البناء الخارجي أبدًا. ويتحقق السكربت من أن الناتج يبدأ ببايتات WebAssembly السحرية، ثم ينسخه إلى مجلد مخرجات البناء.

أما SIBUNA_POW_WASM=<path> فيضمِّن بدلًا من ذلك وحدة مبنية مسبقًا، للبناءات التي لا يتوفر فيها هدف WebAssembly. ثم يُدرج البرنامج الخدمي البايتات في ملفه التنفيذي:

```
/// برنامج الحل في المتصفح، يبنيه `build.rs` من `cmd/sibuna-pow-wasm`.
pub const WASM_SOLVER: &[u8] =
    include_bytes!(concat!(env!("OUT_DIR"), "/sibuna-pow.wasm"));
/// عامل الويب (Web Worker) الخاص ببرنامج الحل.
pub const WORKER_JS: &str = include_str!("../../../web/worker.js");
```

لا يوجد تنفيذ ثانٍ قد يحيد عن الأول. يقدم البرنامج الخدمي الوحدة من /__sibuna/wasm/sibuna-pow.wasm مع ترويسة تخزين مؤقت لمدة ساعة (Cache-Control: public, max-age=3600). وعند بنائها بملف تعريف البناء أعلاه، يبلغ حجم الوحدة 14,744 بايت مع المستويين كليهما (wasm_bytes في benchmarks/results/primitives-latest.json).

الدوال المصدَّرة

تصدِّر الوحدة ست دوال، ويستدعيها سكربت العامل بأسمائها:

الدالة المصدَّرة	الغرض
sibuna_get_buffer_ptr() / sibuna_get_buffer_len()	مخزن إدخال من 256 بايت يكتب فيه العامل سلسلة التحدي
sibuna_solve_step(ptr, len, bits, start, max_steps)	Hashcash: يبحث في max_steps قيمة مؤقتة بدءًا من start؛ ويعيد القيمة المؤقتة أو القيمة الحارسة التي كل بتاتها آحاد
sibuna_solve_sha256(ptr, len, bits)	Hashcash في دفعة واحدة بدءًا من القيمة المؤقتة 0 (خطوة u32::MAX مع sibuna_solve_step)
sibuna_posw_solve(len, depth, challenges)	العمل المتسلسل: يشغِّل المُبرهِن على المخزن ويعيد طول الإثبات، أو 0 عند الوسائط غير الصالحة
sibuna_posw_proof_ptr()	بداية بايتات الإثبات في مساحة عمل المُبرهِن

مساحة عمل المُبرهِن، أي المستويات العليا المحتفَظ بها ومكدس الأشقاء والإثبات، منطقة ثابتة من نحو 92 KiB؛ لا يُربَط أي مُخصِّص ذاكرة، وليس للوحدة أي استيرادات. Workspace::new دالة const fn، لذلك لا يكلف المتغير الثابت إلا ذاكرة مُصفَّرة، لا بيانات مُهيَّأة داخل الوحدة.

> **التنفيذ: sibuna_solve_step** الدالة المصدَّرة لـ Hashcash: تستعير التحدي من الذاكرة الخطية وتشغِّل دفعة واحدة من البحث المشترك عن القيمة المؤقتة. في cmd/sibuna-pow-wasm/src/lib.rs (الحزمة sibuna-pow-wasm).

```rust
#[unsafe(no_mangle)]
pub unsafe extern "C" fn sibuna_solve_step(
    challenge_ptr: *const u8,
    challenge_len: usize,
    difficulty_bits: u32,
    start_nonce: u64,
    max_steps: u32,
) -> u64 {
    let challenge: &[u8] = if challenge_len == 0 {
        &[]
    } else {
        // SAFETY: يضمن المستدعي وجود `challenge_len` بايتًا قابلة للقراءة عند
        // `challenge_ptr` لا تتغير طوال مدة الاستدعاء.
        unsafe { core::slice::from_raw_parts(challenge_ptr, challenge_len) }
    };
    Search::new(challenge)
        .find(difficulty_bits, start_nonce, u64::from(max_steps))
        .unwrap_or(EXHAUSTED)
}
```

الدوال المصدَّرة هي حد الدوال الخارجية للحزمة، والموضع الوحيد الذي توجد فيه شيفرة unsafe. مخزن الإدخال ومساحة عمل المُبرهِن متغيران ثابتان داخل ExportCell، وهو غلاف حول UnsafeCell يُعلَن Sync مع سبب مذكور: مثيل WebAssembly بلا ذاكرة مشتركة يشغِّل خيطًا واحدًا، وكل دالة مصدَّرة تنشئ مراجعها وتستخدمها وتُسقطها قبل أن تعود، فلا يكون حيًّا في أي لحظة أكثر من مرجع واحد إلى الخلية. على المضيف، حيث تستدعي الاختبارات الدوال المصدَّرة مباشرة، يُسلسِل قفل (mutex) خاص بالاختبارات تلك الاستدعاءات. يؤدي الذعر (panic) إلى توقف فوري (التعليمة unreachable) بدل تنسيق رسالة، وهذا يُبقي الوحدة خالية من شيفرة التنسيق ومن الاستيرادات؛ والذعر غير قابل للوصول بحكم البناء، لأن دوال sibuna-pow كلية ولا تذعر أبدًا.

تجزئة البادئة مسبقًا

تستوعب الحلقة الداخلية لـ Hashcash التحدي والنقطتين مرة واحدة، ثم تنسخ حالة SHA-256 لكل قيمة مؤقتة، فيكلف كل مرشح عملية ضغط واحدة بدل اثنتين. البحث هو sibuna_pow::hashcash::Search، وتشترك فيه وحدة المتصفح وبرنامج الحل المرجعي الأصلي:

```rust
pub fn new(challenge: &[u8]) -> Self {
    let mut prefix = Sha256::new();
    prefix.update(challenge);
    prefix.update(b":");
    Self { prefix }
}

pub fn digest(&self, nonce: u64) -> Digest {
    let mut buf = [0u8; NONCE_DIGITS_MAX];
    let mut hasher = self.prefix.clone(); // نسخة بالقيمة من البادئة المستوعَبة
    hasher.update(format_decimal(nonce, &mut buf));
    hasher.finalize()
}
```

```
pub fn find(&self, bits: u32, start: u64, steps: u64) -> Option<u64> {
    let mut nonce = start;
    for _ in 0..steps {
        if meets_bits(&self.digest(nonce), bits) {
            return Some(nonce);
        }
        nonce = nonce.wrapping_add(1);
    }
    None
}
```

حالة دالة التجزئة قيمة بسيطة من نحو 110 بايت، لذلك يكون نسخها رخيصًا. يستخدم مُبرهِن العمل المتسلسل الحيلة نفسها، إذ يستوعب العبارة χ مرة واحدة في base_hasher وينسخها لكل عقدة، وصار المتحقِّق يفعل ذلك أيضًا: يشتق كل ورقة مطلوبة في التحدي من حالة استوعبت العبارة مرة واحدة، فيوفر التحقق t عملية ضغط مع الملخصات نفسها.

على الخادم، تستخدم دالة الضغط امتدادات SHA في x86-64 أو تعليمات SHA-2 في ARMv8، ويُكشف عنها وقت التشغيل. أما في المتصفح فلا توجد تعليمات كهذه، ويقضي برنامج الحل معظم وقته تقريبًا في دالة الضغط المحمولة (crates/crypto/src/sha256/portable.rs). وقد كُتبت هذه الدالة من أجل WebAssembly: الجولات الـ 64 مفرودة، وجدول الرسالة يقيم في نافذة من 16 كلمة تكتب فيها كل جولة ابتداءً من الجولة 16 فوق الكلمة التي لم تعد تحتاجها، والمتغيرات العاملة الثمانية يُعاد تسميتها من جولة إلى أخرى بدل إزاحتها:

```
fn compress_block(state: &mut State, block: &Block) {
    let mut w = [0u32; 16];
    for (word, bytes) in w.iter_mut().zip(block.as_chunks::<4>().0) {
        *word = u32::from_be_bytes(*bytes);
    }
    let [mut a, mut b, mut c, mut d, mut e, mut f, mut g, mut h] = *state;
    eight_rounds!(w, 0, a, b, c, d, e, f, g, h);
    eight_rounds!(w, 8, a, b, c, d, e, f, g, h);
    // ... الجولات من 16 إلى 55
    eight_rounds!(w, 56, a, b, c, d, e, f, g, h);
    for (word, value) in state.iter_mut().zip([a, b, c, d, e, f, g, h]) {
        *word = word.wrapping_add(value);
    }
}

/// تعيد كلمة الجدول W(i)، وتحسبها في مكانها للجولات من 16 فصاعدًا.
#[inline(always)]
fn schedule(w: &mut [u32; 16], i: usize) -> u32 {
    if i >= 16 {
        let w15 = w[(i + 1) % 16];
        let w2 = w[(i + 14) % 16];
        let s0 = w15.rotate_right(7) ^ w15.rotate_right(18) ^ (w15 >> 3);
        let s1 = w2.rotate_right(17) ^ w2.rotate_right(19) ^ (w2 >> 10);
        w[i % 16] = w[i % 16]
            .wrapping_add(s0)
            .wrapping_add(w[(i + 9) % 16])
            .wrapping_add(s1);
    }
```

```
    w[i % 16]
}
```

كل `eight_rounds!` دورة كاملة لأسماء المتغيرات: يمررها الماكرو مُدارة بموضع واحد في كل جولة، فلا تكتب الجولة إلا اثنين منها ولا تحرك الستة الأخرى أبدًا. ومع فهرس جولة حرفي، يصبح كل موضع في الجدول ثابتًا، وتصير الكتلة كلها شيفرة خطية تبقى قيمها في المتغيرات المحلية لـ WebAssembly. يعمل هذا الترتيب أسرع بنحو 20% في V8 من جدول من 64 كلمة مع حلقة غير مفرودة، وأسرع بنحو 14% في التنفيذ الأصلي (قيس على تنفيذ Rust، أكتوبر 2026).

بروتوكول العامل

الأهداف نقرأ بروتوكول الرسائل، ونفهم تسلسل البدائل، ونرى خطأ القيمة الحارسة الذي سببه حد الأعداد الصحيحة ذات الإشارة بين WebAssembly وJavaScript.

ترسل الصفحة رسالة واحدة إلى `web/worker.js` وتتلقى رسائل التقدم والبديل والنتيجة:

```
// الصفحة -> العامل
{ challenge: spec.id, algorithm: spec.algorithm, difficulty: spec.difficulty,
    challenges: spec.challenges }

// العامل -> الصفحة
{ type: 'progress', iterations: 40000 }
{ type: 'fallback', message: 'Wasm unavailable; running JS prover' }
{ type: 'solved', challenge: spec.id, solution: { nonce: '90766' } }  // hashcash
{ type: 'solved', challenge: spec.id, solution: { proof: '<base64url>' } } // posw
{ type: 'error', message: '...' }
```

يجرب العامل WebAssembly أولًا، ويلجأ إلى JavaScript الخالصة عندما تكون `WebAssembly` غائبة، أو محجوبة بسياسة، أو يتعذر إنشاء مثيلها. وتحاكي مُبرهِنات JavaScript مُبرهِن Rust في `sibuna-pow` تسميةً بتسمية، وتنتج خرجًا متطابقًا بايتًا ببايت. وتثبِّت الاختبارات قيمًا مؤقتة وإثباتات محددة للمستويين، ويشترط SID 0010 أن ينجح برنامج الحل في متصفح بلا واجهة مع WebAssembly ومع بديل JavaScript كليهما. مُبرهِنات JavaScript أبطأ بكثير، لكنها تضمن أن كل متصفح يستطيع العبور.

القيمة الحارسة ذات الإشارة تعيد WebAssembly عددًا صحيحًا من 64 بت إلى JavaScript بوصفه `BigInt` ذا إشارة. تشير الدالة المصدَّرة لـ Hashcash إلى "batch exhausted" بقيمة كل بتاتها آحاد (`EXHAUSTED`، `u64::MAX`)، وتراها JavaScript على أنها `-1n`، لا `18446744073709551615n`. كان العامل الأصلي يقارن بالقيمة بلا إشارة، فتخرج الحلقة بعد الدفعة الأولى بقيمة مؤقتة زائفة. الإصلاح استدعاء واحد، `BigInt.asUintN(64, result)`، ويحتفظ به العامل:

```
// تصل نتائج i64 من WebAssembly أعدادًا من نوع BigInt ذات إشارة، لذا فالقيمة
// الحارسة "exhausted" لنوع u64 (كل بتاتها آحاد) تُقرأ على أنها 1n-.
const found = BigInt.asUintN(64,
    ex.sibuna_solve_step(ptr, encoded.length, bits, nonce, maxSteps));
if (found !== 18446744073709551615n) return found.toString();
```

يتعمم الدرس: كل قيمة u64 تعبر هذا الحد يجب أن تُطبَّع. لا تستطيع Rust فرض ذلك على جانب JavaScript؛ فنوع الدالة المصدَّرة هو u64، والتحويل يحدث داخل المحرك.

المعايرة

الأهداف نربط إعدادات الصعوبة بالزمن الفعلي في محرك متصفح وعلى معالج أصلي.

لا تختر صعوبة من متوسط زمن الحل على حاسوب محمول واحد. لـ Hashcash ذيل هندسي، وتشمل ساعات المتصفح الجدولة والحالة الحرارية وبدء التشغيل والترجمة. ينبغي أن تسجل معايرة النشر إصدار المتصفح والجهاز والخوارزمية والمعلمات وتوزيع حلول متكررة. افصل جلب الوحدة وترجمتها عن التجزئة في الحالة المستقرة.

خطة معايرة تطبيقية ثبِّت b، وسخِّن الوحدة، وحُلَّ 100 لغز صادرة بصورة مستقلة على الأقل لكل فئة من الأجهزة المستهدفة. سجِّل الوسيط والمدد عند المئينات العليا وحالات الفشل وأحجام الإثبات. كرِّر عند $b + 1$. يتضاعف العدد المتوقع للمحاولات؛ ولا يلزم أن تتضاعف كل عينة منفردة. اختر إعدادًا وفق ميزانية الجهاز البطيء، ثم اختبر التدفق الكامل من الجلب إلى الحل إلى التحقق.

ليست قياسات العمليات الأولية الأصلية في المستودع دليلًا على زمن حل موعود لهاتف أو متصفح. قد تشغِّل وحدة WASM الحساب نفسه مع اختلاف تكاليف بدء التشغيل والتنفيذ والذاكرة: ففي التنفيذ الأصلي يجزِّئ الخادم باستخدام SHA-NI، بينما يشغِّل المتصفح دالة الضغط المحمولة المذكورة أعلاه. تشكل رسائل العامل الحد الذي يمكن عنده قياس هذه التكاليف من دون إضافة مؤقِّت إلى معالج الطلبات في الخادم.

الصفحة الوسيطة

الصفحة (`web/challenge.html`، المضمَّنة في البرنامج الخدمي) ملف HTML واحد فيه بطاقة وشريط تقدم غير محدد وسطر حالة. تطلب التحدي لموقعها نفسه (`?path=`)، وتُطلق العامل، وترسل الحل بـ `fetch`، وتعيد التحميل عند النجاح؛ تُعاد استدعاءات `fetch` بتراجع أُسِّي مع تفاوت عشوائي، وتُعرض إجابة التحقق `4xx` بعنوانها التشخيصي. يسمح `Content-Security-Policy` للصفحة بسكربت واحد بالضبط، هو كتلة برنامج الحل، عبر تجزئة SHA-256 الخاصة به، التي يحسبها البرنامج الخدمي مرة واحدة من الصفحة المضمَّنة. ورابط المصيدة الموضوع خارج الشاشة

(`/__sibuna/honeypot`) غير مرئي للناس وللمتصفحات التي تشغّل السكربتات؛ أما الزواحف العمياء التي تتبع كل `href` فتحظر نفسها.

تمرين 7.1. لنفترض أن مُبرهِن WebAssembly يحل تحديًا عند الصعوبة الافتراضية في 20 ms على حاسوب محمول، وأن مُبرهِن العمل المتسلسل في JavaScript أبطأ منه 50 مرة. قدِّر زمن الحل بالبديل على ذلك الحاسوب وعلى هاتف أبطأ منه أربع مرات، واقترح سياسة للعملاء الذين يبلغون عن استخدام البديل (تلميح: يرسل العامل رسالة `fallback` يمكن للصفحة تمريرها).

اشرح اللامتغير. اشرح لماذا تُعد ترجمة وحدة المتصفح من مصادر المتحقِّق في الخادم ضمانًا أقوى من اختبار تنفيذين أحدهما مقابل الآخر.

الجزء الثامن

التقييم التجريبي

نعرض قياسات *Sibuna* التي سجلتها أدوات قياس مودَعة في المستودع: أزمنة العمليات الأولية، والمنتجات الكاملة تحت مولد حمل خارجي مع حساب وقت *CPU* والذاكرة، وزمن الاستجابة عند حمل معروض متساوٍ من مولد حلقة مفتوحة، ومقارنة مع *Anubis*، وهو بوابة إثبات عمل واسعة الانتشار. ويُكمل السجلَّ تشغيلٌ لوحدة التحكم بثماني لوحات معلومات حية. كل رقم في هذا الجزء مأخوذ من ملف نتائج مودَع في المستودع، يذكره كل جدول مع وقت إنشائه، أو من سجل تصميم. يجيب الاختبار الوظيفي وقياس الأداء عن سؤالين مختلفين، والقياس على مضيف واحد لا يصلح ترتيبًا على مضيف آخر.

المنهجية

الأهداف بنهاية هذا الفصل، ينبغي أن نستطيع إعادة إنتاج كل رقم في هذا الجزء، وأن نذكر ما تشمله كل أداة قياس وما تستبعده، وأن نشرح معنى أعمدة المجال، وأن نميز الرقم المقيس من الرقم المسجل أو المنشور.

أدوات القياس أوامر لأداة Rust واحدة، `tools/bench` (الملف التنفيذي `sibuna-bench`)، تُشغَّل عبر `cargo xtask bench`. وتسجل أداة أخرى تشغيل وحدة التحكم.

الجدول 11 أدوات القياس وملفات نتائجها، وكلها تحت `benchmarks/results/`

أداة القياس	ما تقيسه	ملف النتائج
`cargo xtask bench primitives`	ستة عشر عبء عمل أوليًا، وصفان إضافيان لمسار محدد المعدل بالتجزئة، على CPU واحدة: سبع دفعات، والوسيط لكل عملية مع مجال الأدنى-الأقصى	`primitives-latest.json`، `primitives-latest.md`
`cargo xtask bench product`	البرامج الخدمية الكاملة تحت `wrk`: الإنتاجية، ومئينات زمن الاستجابة، ووقت CPU لكل طلب، وذروة الذاكرة المقيمة لـ Gate وShield، في وضعي forward-auth والوكيل العكسي	`product-latest.json`، `product-latest.md`
`cargo xtask bench latency`	مئينات زمن استجابة الطلبات المقبولة عند معدلات معروضة ثابتة، من مولد حلقة مفتوحة	`latency-latest.json`، `latency-latest.md`

أداة القياس	ما تقيسه	ملف النتائج
`cargo xtask bench product --anubis`	أعباء المنتجات الكاملة مع Anubis منتجًا إضافيًا، خلف `caddy respond`	`comparison-latest.json`، `comparison-latest.md`
`cargo xtask bench admission --anubis`	عمليات القبول في الثانية لـ Sibuna وAnubis في وضع forward-auth، من عميلين يستخدمان keep-alive	`admission-latest.json`، `admission-latest.md`
`sibuna-browser dashboards` (`tools/browser`)	تشغيل وظيفي لوحدة التحكم: ثماني لوحات معلومات حية في Chromium بلا واجهة بينما يرسل مولد حركة مرور	`console-8-dashboards-2026-10-15.md`

ملفات `.md` ملخصات. أما ملفات `.json` فتحفظ كل تكرار، والجهاز، والمراجعة، ومسار كل ملف تنفيذي مقاس وحجمه وقيمة SHA-256 له. يحفظ Git التاريخ، لذا لا يحتفظ المجلد إلا بأحدث الملفات. سجلت نسخة أقدم من أداة القياس بعض الملفات، وتحفظ هذه الملفات أيضًا صفوف بناء ثانٍ للبرنامج الخدمي قيس في التشغيل نفسه؛ ويعرض هذا الجزء صفوف Sibuna.

العمليات الأولية

يعمل كل عبء في سبع دفعات موقوتة بعد إحماء غير موقوت من 1,000 عملية على الأكثر. يُعاد بناء الحالة القابلة للتغيير (مجموعة الحلول المُنفَقة، ومحدد المعدل) قبل كل دفعة، خارج المؤقت. تحيط المؤقتات بالدفعات الكاملة، لا بالعمليات المفردة أبدًا: فقراءة الساعة لكل عملية تكلف قدر العمل الذي تقيسه، وهو دون 100 ns. لذلك لا نعرض مئينات لكل عملية، ويكون المجال عبر الدفعات. نعرض الدفعة الوسيطة لكل عملية، ومعها أسرع دفعة وأبطؤها.
تعمل المجموعة في عملية فرعية (`sibuna-bench primitives-run`) مثبتة على CPU واحدة. على مضيف مشترك، يبطئ شقيق SMT مشغول أو تردد ساعة أدنى تشغيلًا كاملًا للمجموعة، لذا تشغل الأداة المجموعة ثلاث مرات (`--rounds`) وتحتفظ، لكل صف، بالجولة ذات الوسيط الأدنى. وداخل الجولة تكون الإحصاءات كما سبق.

المنتجات الكاملة

لكل وضع نشر (forward auth، والوكيل العكسي) ولكل وضع (Gate، وShield)، تبدأ أداة القياس البرنامج الخدمي في الذاكرة مع `--idle-timeout 5 --rate-limit 100000000 --difficulty 8 --algorithm hashcash` وأربعة عمال، عامل لكل CPU في مجموعتها. وتحصل على جلسة بحل تحدي Hashcash الخاص بالبرنامج الخدمي باستخدام `sibuna-pow`، كما يفعل المتصفح. ثم تشغّل أربعة أعباء عمل بـ `wrk` عبر اتصالات keep-alive: خيطان، و64 اتصالًا، و5 ثوانٍ، و3 تكرارات.

- `admitted`: الطلب `GET /private` مع ملف تعريف ارتباط جلسة صالح. في وضع الوكيل العكسي يصل الطلب إلى الخادم الأصلي وتُمرَّر إجابته؛ وفي وضع forward-auth يجيب البرنامج الخدمي بنفسه بـ `200`.

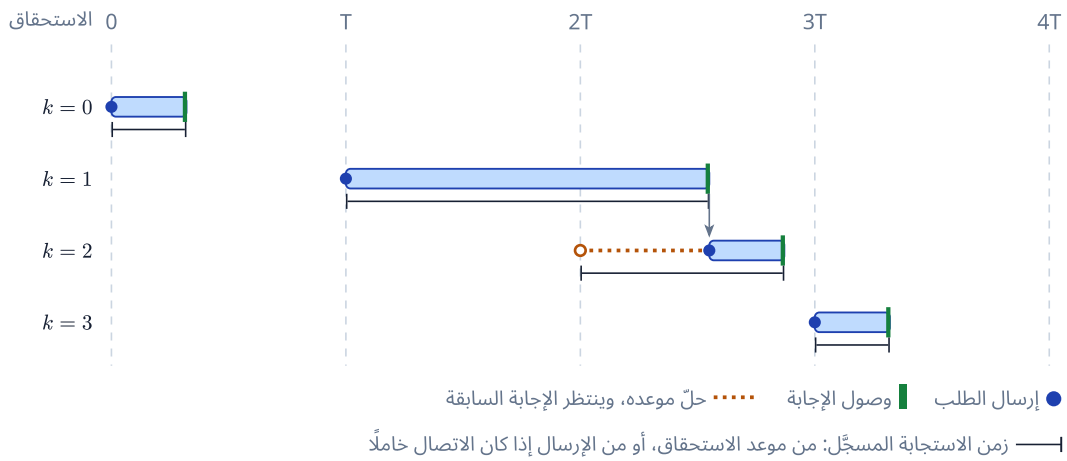
- `challenged`: الطلب نفسه بلا ملف تعريف ارتباط، من User-Agent متصفح. يقدم البرنامج الخدمي تحديه: `200` مع HTML في وضع الوكيل العكسي، و`401` في وضع forward-auth.
- `allowed_static`: الطلب `GET /robots.txt` من `curl/8.0`، ويقبله الوضعان كلاهما بلا تحدٍّ.
- `attack`: جلسة صالحة مع `q=' OR 1=1--`. يرفضه Shield وحده (`403`). أما Gate فيقبله، ويمرره في وضع الوكيل العكسي إلى الخادم الأصلي.

الخادم الأصلي بديل تجريبي (`sibuna-bench origin`): المكتبة القياسية فقط، وخيط لكل اتصال، وkeep-alive، وصفحة من 41 بايت. وقت CPU لكل طلب هو وقت المستخدم والنظام للبرنامج الخدمي من `/proc/<pid>/stat` طوال التشغيل (بنبضات 10 ms)، مقسومًا على الطلبات التي عدها `wrk`؛ ولا يدخل فيه وقت CPU للخادم الأصلي. ذروة الذاكرة هي `VmHWM` من `/proc/<pid>/status`، وتُعاد ضبطها عبر `/proc/<pid>/clear_refs` قبل كل تشغيل، حتى يعرض كل تشغيل ذروته الخاصة. كل رقم هو وسيط التكرارات، ويُحفظ الأدنى والأقصى بجانبه في ملف JSON.

`wrk` مولد حلقة مغلقة: يبقي 64 طلبًا قيد التنفيذ. لذلك يتلقى البرنامج الخدمي الأسرع طلبات أكثر، وتُسقط مئينات `wrk` الطلبات التي لم يرسلها أثناء انتظاره إجابة بطيئة (الإغفال المنسق). يبين p99 الخاص به سلوك البرنامج الخدمي تحت تشبعه هو، لا زمن استجابته عند حمل معين. وتجيب أداة قياس زمن الاستجابة عن ذلك السؤال.

زمن الاستجابة عند حمل متساوٍ

يشغل مولد الحلقة المفتوحة (`sibuna-bench generate`) خيطًا حاجبًا واحدًا لكل اتصال. يرسل كل اتصال من 64 اتصالًا الطلب `GET /private` مع جلسة وفق جدول ثابت: يستحق الطلب k من الاتصال في اللحظة $t_0 + o + kT$، حيث o إزاحة الاتصال وT فاصله، مستقلًا عن الإجابات السابقة. إذا وصلت إجابة بعد موعد استحقاق الطلب التالي، يُرسل ذلك الطلب فورًا ويُحسب زمن استجابته من موعد استحقاقه، كما في `wrk2`. وإذا كان الاتصال خاملًا عند موعد الاستحقاق، يُحسب زمن الاستجابة من الإرسال الفعلي، فلا يُحمَّل على البرنامج الخدمي تجاوزُ نوم المولد نفسه.



الشكل 10 اتصال واحد لمولد الحلقة المفتوحة. يُجاب الطلب 1 بعد موعد استحقاق الطلب 2، فيُرسَل الطلب 2 فورًا ويُحسب زمن استجابته من 2T؛ ويجد الطلب 3 الاتصال خاملًا فيُحسب من إرساله.

تشغل أداة القياس وضع Shield وعبء الطلبات المقبولة عند خمسة معدلات معروضة: 10,000 و25,000 و50,000 و100,000 و150,000 طلب في الثانية، لكل منها إحماء لمدة ثانيتين، و10 ثوانٍ مقاسة، و3 تشغيلات. تستطيع أداة القياس قياس بناءين للبرنامج الخدمي في تشغيل واحد: عندئذ تبدأ كليهما على مجموعة CPU نفسها وتتناوب بينهما عند كل معدل وتكرار، ولا يستهلك الخامل منهما وقت CPU، حتى يؤثر الانجراف البطيء للجهاز فيهما بالتساوي. وقد فعل التشغيل المسجل ذلك. المئينات بطريقة أقرب رتبة على كل طلب مقاس في تشغيل واحد؛ وكل رقم هو وسيط التشغيلات الثلاثة، مع مجال الأدنى-الأقصى لـ p99 بجانبه.

مجموعات CPU

تؤثر مجموعات CPU أكثر من أي إعداد آخر. تعمل البرامج الخدمية على وحدات CPU من 0 إلى 3 (`--cpus`؛ و`--workers` هو حجم تلك المجموعة). ويعمل `wrk` ومولد زمن الاستجابة والخادم الأصلي على وحدات CPU من 8 إلى 15 (`--load-cpus`). وتعمل مجموعة العمليات الأولية على CPU رقم 4 (`--cpu`). تُثبَّت كل عملية بـ `taskset -c` قبل بدئها، فيرث كل خيط تنشئه المجموعة. وتستخدم أداة قياس القبول وحدتي CPU، 0 و1، للمنتج، ووحدات CPU من 8 إلى 15 لعملائها.



يدل الرقم الصغير في كل نواة على شقيقها في SMT، وهو إحدى وحدات CPU من 16 إلى 31. لا يستخدمها أي تشغيل.

الشكل 11 مجموعات CPU على AMD Ryzen 9 5950X. وحدات CPU من 0 إلى 7 هي الأنوية الثماني في أول CCD، ومن 8 إلى 15 أنوية الثاني، ومن 16 إلى 31 أشقاؤها في SMT. لا يتشارك البرنامج الخدمي ومولد الحمل نواة أبدًا.

على الأجهزة الأخرى نختار مجموعات لا تتشارك الأنوية (يعرض `lscpu -e` البنية).

الجهاز والظروف

سُجلت الملفات على `paxos-zig`: معالج AMD Ryzen 9 5950X 16-Core Processor (32 وحدة CPU منطقية)، وLinux 7.0.0-28-generic x86_64، ومنظم التردد `powersave` مع تفعيل التعزيز، وrustc 1.99.0. سُجلت ملفات العمليات الأولية والمنتجات وزمن الاستجابة في 15 أكتوبر 2026، والمقارنة مع Anubis في 16 أكتوبر 2026. البرنامج الخدمي هو `cargo build --release -p sibuna`، أي البناء الافتراضي مع تخزين SQLite ووحدة التحكم مضمنين (SID 0010)؛ وتذكر ملفات المنتجات وزمن الاستجابة ملفًا تنفيذيًا من 49,113,856 بايت. يسجل كل ملف مراجعته (`efce14d` للعمليات الأولية، و`249d913` لتشغيلي المنتجات وزمن الاستجابة) مع `dirty: true`؛ وكانت شجرة العمل في الأخير تحوي بالفعل خيوط الاتصالات المعاد استخدامها التي يسجلها SID 0002، ويستشهد SID 0002 بـ `latency-latest.md` لبيان أثرها. تذكر ملفات المقارنة والقبول المراجعة `54e4ec5` مع `dirty: true`، لأن شجرة العمل كانت تحوي إضافات أداة القياس في `tools/bench`؛ وبرنامجها الخدمي بناءً بوضع الإصدار من تلك المراجعة (37,450,232 بايت). كان متوسط الحمل لدقيقة واحدة 3.53 قبل تشغيل المقارنة، و0.56 عند انتهاء تشغيل القبول.

تسجل الملفات متوسطات حمل لدقيقة واحدة قدرها 2.46 عند تشغيل العمليات الأولية، و1.30 قبل تشغيل زمن الاستجابة و52.31 بعده، و52.31 قبل تشغيل المنتجات و33.39 بعده. الأرقام المرتفعة سببها خيوط مولد زمن الاستجابة الـ 64، وقد تلاه تشغيل المنتجات مباشرة.

> **نطاق القياس** لا تُنتج أدوات القياس إلا قياسات محلية. يستخدم كل تشغيل مضيفًا واحدًا عبر واجهة loopback: بلا TLS، وبلا WAN، وبلا متصفحات حقيقية. تعمل البرامج الخدمية في الذاكرة، بلا تخزين. تحيط المؤقتات بالدفعات؛ وإعادة ضبط الحالة والإحماء غير موقوتة. لا تعد أداة القياس التخصيصات (تقول الملفات "not instrumented; API/source audit only")؛ وبدلًا من ذلك تتحقق اختبارات الحالة المستقرة في الجزء الخامس من أن مسار الطلب خالٍ من التخصيص. لا ترتبط أي أداة قياس بالبرنامج الخدمي ولا تضيف خطافًا إلى مسار الطلب: تشغل أداتا المنتجات وزمن الاستجابة الملف التنفيذي للإصدار دون تعديل، معرَّفًا بقيمة SHA-256 له. والترتيب على هذا المضيف لا يصلح ترتيبًا على مضيف آخر.

كيف نعيد إنتاج القياسات

نحتاج إلى Linux، و`taskset` (util-linux)، و`wrk` على `PATH`.

```
cargo xtask bench primitives
cargo xtask bench product
cargo xtask bench latency
cargo xtask bench help
```

يبني `cargo xtask bench` البرنامج الخدمي أولًا بـ `cargo build --release -p sibuna`، ثم يشغل `sibuna-bench` في وضع الإصدار. تجد الأداة البرنامج الخدمي بجانب ملفها التنفيذي؛ ويتجاوز `--rust <path>` ذلك. يسرد `cargo xtask bench help` الخيارات، ومنها `--cpus` و`--load-cpus` و`--cpu` و`--rounds` و

`--repetitions` و`--seconds` و`--rates` و`--warmup`. ويعطي القسم الخاص بـ Anubis أدناه الأوامر التي سجلت تلك المقارنة.

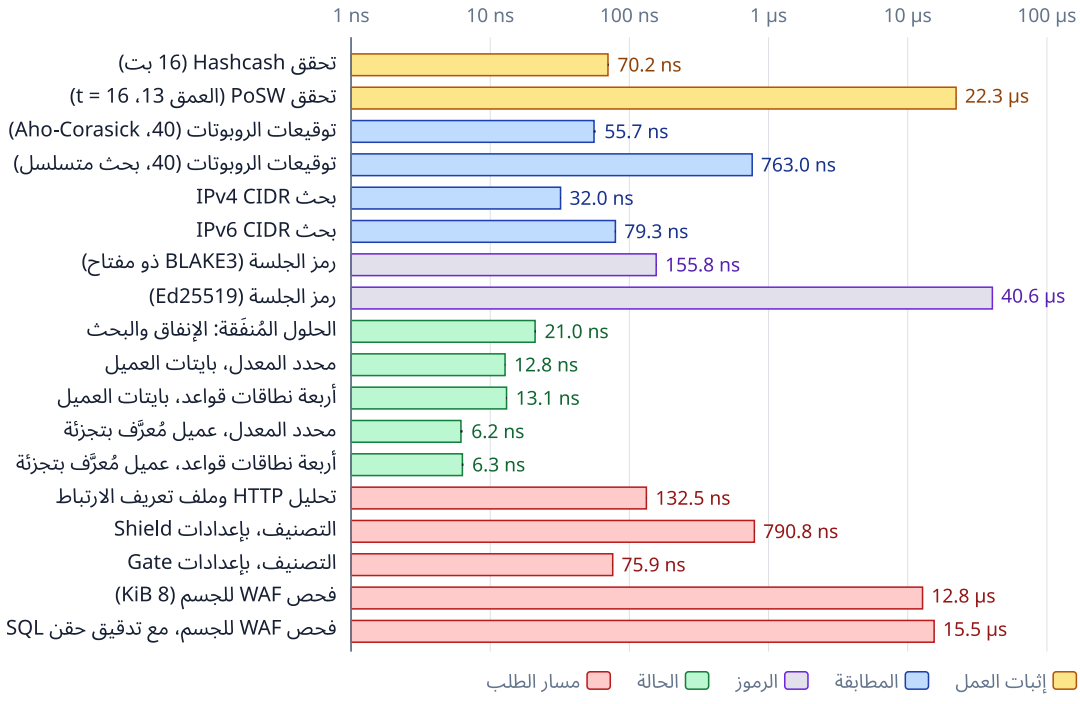
أزمنة العمليات الأولية

يعطي الجدول كل صف من تشغيل العمليات الأولية. يسمي ملف JSON كل صف بالنظام الفرعي وعبء العمل (`pow_verify` و`hashcash_16_bits`، مثلًا).

الجدول 12 توقيتات العمليات الأولية بالنانوثانية لكل عملية على نواة Ryzen 9 5950X واحدة (`benchmarks/results/primitives-latest.md`، أُنشئ في 2026-10-15T22:34:59Z)

عبء العمل	ns/op	الأدنى-الأقصى
تحقق Hashcash (16 بت)	70.2	69.9-71.0
تحقق PoSW (العمق 13، $t = 16$)	22272.4	22257.0-22316.4
توقيعات الروبوتات (40، Aho-Corasick)	55.7	55.3-57.2
توقيعات الروبوتات (40، بحث متسلسل)	763.0	762.2-763.5
بحث IPv4 CIDR	32.0	31.8-32.2
بحث IPv6 CIDR	79.3	78.5-79.6
رمز الجلسة (BLAKE3 ذو مفتاح)	155.8	155.3-156.4
رمز الجلسة (Ed25519)	40587.5	40544.6-40631.1
مجموعة الحلول المُنفَقة (Robin Hood)، الإنفاق والبحث	21.0	20.8-21.3
محدد المعدل (GCRA)، بايتات العميل	12.8	12.7-12.8
محدد المعدل، أربعة نطاقات قواعد، بايتات العميل	13.1	13.0-13.2
محدد المعدل (GCRA)، عميل مُعرَّف بتجزئة	6.2	6.1-6.3
محدد المعدل، أربعة نطاقات قواعد، عميل مُعرَّف بتجزئة	6.3	6.2-6.4
تحليل HTTP وملف تعريف الارتباط	132.5	132.3-133.2
التصنيف، بإعدادات Shield	790.8	787.2-793.1
التصنيف، بإعدادات Gate	75.9	75.6-76.1
فحص WAF للجسم (8 KiB)	12784.3	12771.5-12809.2
فحص WAF للجسم (8 KiB)، مع تدقيق حقن SQL	15485.4	15460.9-15530.1

المجالات ضيقة: لا تبعد أي دفعة في أي صف أكثر من 2.8% عن وسيط صفها. ويعرض الشكل الصفوف نفسها على مقياس لوغاريتمي، فتبقى الصفوف التي تستغرق 6 ns والتي تستغرق 40 us مقروءة جنبًا إلى جنب.



كل عمود هو وسيط 7 دفعات على نواة واحدة من AMD Ryzen 9 5950X 16-Core Processor، في 2026-10-15؛ ويمتد الخط الرفيع من أسرع دفعة إلى أبطئها. المقياس لوغاريتمي.

الشكل 12 أزمنة العمليات الأولية على مقياس لوغاريتمي، مرسومة من `benchmarks/results/primitives-latest.json`.

التحقق من إثبات العمل

يجزئ تحقق Hashcash تحديًا وفاصلًا وقيمة مؤقتة عشرية: 70.2 ns. ويفحص متحقق العمل المتسلسل عمليات الفتح في الرسم: 22,272.4 ns. تختلف استدعاءات التجزئة وأعداد ضغطات SHA-256 فيه عنها في Hashcash لأن أطوال المدخلات تختلف. كلا الصفين محدود بأداء SHA-256، وتستخدم `sibuna-crypto` تعليمات SHA-NI عندما تتوفر في CPU، وتكشف توفرها وقت التشغيل (SID 0010).

المطابقة

تستخدم آلة Aho-Corasick الكثيفة والبحث المتسلسل عن السلاسل الفرعية أنماطًا متطابقة ودلالة "أي تطابق" نفسها. الآلة أسرع من البحث المتسلسل 13.7 مرة على التوقيعات الـ 40 نفسها (55.7 مقابل 763.0 ns). تستخدم جداولها أصناف بايتات بدل 256 عمودًا لكل حالة (SID 0006، الجزء F)، فيبقى جدول توقيعات الروبوتات عند 42,244 بايت (جدول الحجم والذاكرة أدناه). وتستغرق مصنفات CIDR 32.0 ns (IPv4) و79.3 ns (IPv6).

الرموز

يُتحقق من رمز BLAKE3 MAC في 155.8 ns. تنفذ `sibuna-crypto` خوارزمية BLAKE3 تنفيذًا محمولًا، بلا مسارات SIMD؛ فهذه تتفوق على المدخلات الكبيرة، لكن Sibuna يجزئ بـ BLAKE3 مدخلات دون 100 بايت، حيث تكون الشفرة المحمولة سريعة. يُتحقق من رموز Ed25519 المدمجة في 40,587.5 ns: يستخدم المتحقق أرقام NAF بعرض 5 وعرض 8 ويحتفظ بالمفتاح العام بعد فك ترميزه. يكلف تحقق Ed25519 واحد قدر 260 تحققًا من MAC تقريبًا؛ وMAC هو مخطط الرموز الافتراضي في Sibuna.

الحالة

يُدرج صف مجموعة الحلول المُنفَقة وسمًا جديدًا ثم يبحث عنه مرة أخرى: 21.0 ns. للجدول في هذا الصف 16 شريحة في كل منها 4,096 موضعًا (65,536 وسمًا؛ والافتراضي في البرنامج الخدمي 1,048,576)، ويُفرَّغ في مكانه قبل كل دفعة، خارج المؤقت. لذلك تفحص كل دفعة ذاكرة مقيمة. كانت نسخة أقدم من المجموعة تبني لكل دفعة جدولًا جديدًا مُصفَّرًا بالحجم الافتراضي للبرنامج الخدمي؛ فكانت توقُّت النواة وهي تربط صفحات الجدول عند الكتابة الأولى، وهي تكلفة يدفعها البرنامج الخدمي أثناء عمله مرة واحدة، وليس هذا الصف ما نعرضه.

تحتاج صفوف محدد المعدل إلى عناية. الصفان الخاصان ببايتات العميل (`gcra_check` و `gcra_four_rule_scopes`) يستدعيان واجهة البايتات: يأخذ كل فحص عنوان العميل بايتاتٍ ويحسب أولًا قيمة SipHash-1-3 ذات مفتاح له، بسر مأخوذ من عشوائية نظام التشغيل. يمنع المفتاح السري أيًّا كان من حساب عناوين تشارك الضحيةَ نافذة السبر (SID 0010 ووثائق `sibuna-store`). تستغرق هذه الصفوف 12.8 و13.1 ns.

لا يستخدم البرنامج الخدمي واجهة البايتات على مسار الطلب. بل يجزئ العميل مرة واحدة لكل طلب (`ClientHasher`) ويمرر التجزئة إلى كل جدول يستشيره. يقيس صفا العميل المُعرَّف بتجزئة (`gcra_check_hashed` و`gcra_four_rule_scopes_hashed`) ذلك المسار، مع حساب التجزئات قبل الحلقة: 6.2 و6.3 ns لكل فحص. الفرق بين وسيطي صفوف البايتات وصفوف التجزئة، 6.6 ns، هو تقريبًا ثمن التجزئة ذات المفتاح على هذه المدخلات، ويدفعه البرنامج الخدمي مرة لكل طلب لا مرة لكل جدول.

محلل HTTP

يحلل صف المحلل طلب متصفح من 400 بايت ويبحث عن ملف تعريف ارتباط: 132.5 ns. ويتفق معه اختبار التوقيت الخاص بالحزمة (`crates/http`: نحو 135 إلى 180 ns على الجهاز نفسه). المحلل بلا نسخ؛ كل حقل مقطع من المخزن المؤقت للاتصال.

محرك السياسة

يستغرق التصنيف الكامل لطلب متصفح 790.8 ns، وبإعدادات Gate يستغرق 75.9 ns. تترك الصفوف عنوان العميل المحلل في الطلب فارغًا، فيحلل المحرك نص عنوان العميل داخل التقييم. أما البرنامج الخدمي فيحلل العنوان مرة لكل طلب، قبل التقييم، لذا تشمل هذه الصفوف تكلفة صغيرة يدفعها البرنامج الخدمي خارج المحرك. يبلغ مجموع جداول المحرك في الكومة 209,052 بايت (SID 0002).

فحص WAF

يستغرق الفحص الدلالي لجسم من 12,784.3 KiB 8 ns، أي 1.56 ≈ 12784.3/8192 ns لكل بايت. يفحص الصف الثاني الجسم نفسه مع تدقيق فئة حقن SQL وفرض الفئات الأخرى: 15,485.4 ns. عندما تكون فئة ما في وضع التدقيق لا الفرض، يُفحص كل حقل لكل فئة (inspection modes، "SID 0004")، يفحص المحرك الحقل بآلة التوقيعات مرة واحدة، لمجموعة الفئات التي تظهر توقيعاتها، ثم يبحث فقط في الفئات المطلوبة منها، لذا يكلف تدقيق فئة واحدة أكثر بنحو الخُمس من فرض الفئات كلها، لا فحصًا مستقلًا لكل فئة.

مقارنة المنتجات الكاملة

> **الأهداف** اقرأ إنتاجية البرنامج الخدمي وزمن استجابته ووقت CPU وذاكرته بوصفه عملية كاملة، وقارن أوضاعه وأوضاع نشره، وبيِّن في أي الصفوف يضع البرنامج الخدمي الحد وفي أيها يضعه مولد الحمل.

تعطي الجداول وسيطات ثلاثة تشغيلات لـ `wrk` مدة كل منها خمس ثوانٍ، لكل وضع وعبء عمل. الحالة بجانب كل عبء هي الإجابة التي توقعتها أداة القياس وتلقتها.

وضع Forward-Auth

يسأل وكيل الدخول البرنامج الخدمي هل يقبل الطلب؛ لا يوجد خادم أصلي ولا تمرير للجسم، لذا هذا أنقى قياس لتكلفة القبول.

الجدول 13 وضع forward-auth، Gate وShield؛ زمن الاستجابة بالميكروثانية (`benchmarks/results/product-latest.md`، أُنشئ في 2026-10-15T23:15:24Z)

الوضع	عبء العمل (الحالة)	req/s	p50	p90	p99	CPU us/req	ذروة RSS KiB
Gate	(200) `admitted`	211,345	151	175	305	11.84	15912
Gate	(401) `challenged`	198,315	160	180	326	13.15	16176
Gate	`allowed_static` (200)	211,053	152	175	305	11.73	16176
Gate	(200) `attack`	211,618	151	176	305	11.84	16176
Shield	(200) `admitted`	208,101	152	178	308	12.87	15928
Shield	(401) `challenged`	197,532	161	183	327	13.77	16192
Shield	`allowed_static` (200)	211,636	151	175	305	11.78	16192
Shield	(403) `attack`	210,549	152	175	307	12.32	16192

في وضع forward-auth لا يكون البرنامج الخدمي عنق الزجاجة. تبقى أنويته المشغولة (وقت CPU مقسومًا على وقت الساعة الفعلي، `cores_busy_median` في ملف JSON) بين 2.47 و2.72 من وحدات CPU الأربع للبرنامج الخدمي، ويقع كل صف بين 197,532 و211,636 طلبًا في الثانية. يشير ذلك إلى أن مولد الحمل هو السقف. وقرب ذلك السقف يتغير فرق بضعة في المئة بين الصفوف من تشغيل إلى آخر: فصف Gate `admitted` وحده يمتد من 205,151 إلى 211,513 طلبًا في الثانية عبر تشغيلاته الثلاثة. الرقم الأنفع في هذا الوضع هو وقت CPU لكل طلب: من 11.73 إلى 13.77 us، وهو أكثر قليلًا في Shield منه في Gate، وأكثر ما يكون في صفوف `challenged`، التي تنشئ إجابة التحدي.

وضع الوكيل العكسي

يقف البرنامج الخدمي أمام الخادم الأصلي التجريبي ويمرر الطلبات المقبولة والاستجابات. لذلك تشمل الصفوف الممررة تكلفة الخادم الأصلي نفسه وقفزة loopback إضافية.

الجدول 14 وضع الوكيل العكسي، Gate وShield؛ زمن الاستجابة بالميكروثانية (`benchmarks/results/product-latest.md`، أُنشئ في 2026-10-15T23:15:24Z)

الوضع	عبء العمل (الحالة)	req/s	p50	p90	p99	CPU us/req	ذروة RSS KiB
Gate	(200) `admitted`	142,498	427	548	768	27.68	15896
Gate	(200) `challenged`	199,465	160	181	324	12.98	16160
Gate	`allowed_static` (200)	147,303	411	541	759	26.88	16324
Gate	(200) `attack`	143,900	422	555	745	27.67	16348
Shield	(200) `admitted`	138,965	438	571	857	28.57	16064
Shield	(200) `challenged`	195,188	162	185	328	13.73	16328
Shield	`allowed_static` (200)	145,673	415	551	746	27.27	16328
Shield	(403) `attack`	209,507	151	177	306	12.17	16328

هنا تكون الصفوف الممررة (`admitted` و`allowed_static` و`attack` في Gate، الذي يمرره Gate) محدودة بوحدة CPU: الأنوية المشغولة من 3.96 إلى 3.98 من أربع. تخدم من 138,965 إلى 147,303 طلبًا في الثانية بوقت CPU من 26.88 إلى 28.57 us لكل طلب، أي نحو ضعف تكلفة إجابة يعطيها البرنامج الخدمي بنفسه. أما صفوف `challenged` وصف `attack` في Shield فيجيب عنها البرنامج الخدمي بنفسه، وتشبه صفوف forward-auth. وتحت التشبع يكون p99 في `wrk` ذيل المجدول عند معدل البرنامج الخدمي نفسه، لذا نقرأ ذيول الفصل التالي بدلًا منه.

في ثلاثة صفوف للوكيل العكسي عدَّ `wrk` استجابات خارج الفئتين 2xx و3xx حيث لم يُتوقع أي منها: 3 من 2,162,579 طلبًا (attack Gate)، و1 من 2,082,640 (`admitted` Shield)، و1 من 2,204,358 (`allowed_static` Shield)، مجموعة عبر التشغيلات الثلاثة. لا تذكر الملفات رموز حالة هذه الاستجابات. أظهرت إعادة

التشغيل مع مقاييس البرنامج الخدمي إجابات 503 معدودة في sibuna_overloaded_total: يفتح wrk اتصالاته الـ 64 من عنوان واحد، وهو بالضبط الافتراضي لـ --max-connections-per-ip، وعندما أُغلق اتصال بعد طلبه رقم 4,096، أعاد wrk الاتصال قبل أن يُخلي الاتصال القديم مكانه. صار البرنامج الخدمي الآن يُخلي المكان قبل أن يكتب استجابة الإغلاق ("bug fix"، SID 0002)؛ والجداول أقدم من ذلك الإصلاح.

الحجم والذاكرة

الجدول 15 أحجام البرنامج الخدمي وذاكرته المقيمة (benchmarks/results/product-latest.json، أُنشئ في 15T23:15:24Z-10-2026، وprimitives-latest.json، أُنشئ في 15T22:34:59Z-10-2026)

المقياس	القيمة
الملف التنفيذي للبرنامج الخدمي	49,113,856 بايت
الذاكرة المقيمة في الخمول، تشغيلات المنتجات	من 5,592 إلى 5,788 KiB
ذروة الذاكرة المقيمة أثناء عبء عمل	من 15,896 إلى 16,348 KiB
الذاكرة المقيمة بعد الأعباء	من 16,176 إلى 16,348 KiB
الذاكرة المقيمة في الخمول، تشغيل العمليات الأولية	6,052 KiB
جداول محرك السياسة	209,052 بايت (الكومة)
جدول توقيعات الروبوتات	42,244 بايت (الكومة)
جدول WAF	164,908 بايت (الكومة)
وحدة حل الألغاز في المتصفح (WebAssembly)	14,744 بايت

تمتد المجالات عبر تشغيلات المنتجات الأربعة (وضعا نشر، ووضعان)، وتمتد للذروة عبر أعبائها الستة عشر. أحجام الجداول أحجام كومة يعرضها المحرك. لا يفصِّل السجل حجم الملف التنفيذي.

زمن الاستجابة عند حمل معروض متساوٍ

> **الأهداف** اقرأ توزيع زمن الاستجابة عند حمل معروض ثابت، واقرأ مئين الذيل مع مجاله، وتتبع خللًا واحدًا في الذيل من قياسه إلى إصلاحه.

يأتي الجدولان كلاهما من تشغيل واحد لـ cargo xtask bench latency. يعد عمود الأخطاء أخطاء الاتصال والإدخال/الإخراج والتأطير في المولد.

الجدول 16 وضع Shield، forward-auth، طلبات مقبولة عند معدلات معروضة ثابتة؛ زمن الاستجابة بالميكروثانية (benchmarks/results/latency-latest.md، أُنشئ في 15T23:07:14Z-10-2026)

req/s المعروضة	req/s المحققة	p50	p90	p99 (الأدنى-الأقصى)	p99.9	الأقصى	الأخطاء	CPU us/req
10,000	10,000	30	32	37 (36-39)	58	355	0	16.92
25,000	25,000	28	31	34 (32-34)	45	1035	0	15.83
50,000	50,000	25	27	30 (30-30)	40	3029	0	13.83
100,000	99,998	24	26	41 (40-42)	62	2358	0	13.48
150,000	149,996	25	36	51 (51-52)	74	1753	0	14.16

الجدول 17 وضع الوكيل العكسي، Shield، طلبات مقبولة عند معدلات معروضة ثابتة؛ زمن الاستجابة بالميكروثانية (`benchmarks/results/latency-latest.md`، أُنشئ في 2026-10-15T23:07:14Z)

req/s المعروضة	req/s المحققة	p50	p90	p99 (الأدنى-الأقصى)	p99.9	الأقصى	الأخطاء	CPU us/req
10,000	10,000	56	61	65 (65-66)	76	555	0	32.25
25,000	25,000	51	54	58 (58-60)	72	1105	0	28.63
50,000	50,000	47	50	60 (58-66)	74	3066	0	27.45
100,000	99,996	55	75	100 (100-104)	150	2239	0	28.42
150,000	134,344	583934	544160	19062731 (1631234-2218760)	1967134	1989964	4	29.22

في وضع forward-auth يبقى p50 بين 24 و30 us عند كل معدل. ويبقى p99 عند 51 us أو دونها حتى 150,000 طلب في الثانية، وp99.9 عند 74 us أو دونها. ينفق البرنامج الخدمي من 13.48 إلى 16.92 us من وقت CPU لكل طلب.

في وضع الوكيل العكسي يكون p50 من 47 إلى 56 us حتى 100,000 طلب في الثانية، ويبلغ p99 قيمة 100 us وp99.9 قيمة 150 us عند 100,000. وعند 150,000 طلب في الثانية يتشبع البرنامج الخدمي: ينجز 134,344 طلبًا في الثانية، ويرتفع زمن الاستجابة إلى ثوانٍ، كما ينبغي أن يُظهر اختبار حلقة مفتوحة فوق السعة. هذا المعدل قريب من إنتاجية الوكيل في قياس الحلقة المغلقة في الفصل السابق (138,965 طلبًا في الثانية لصف Shield `admitted`). وقعت أخطاء ذلك الصف الأربعة أثناء الحمل الزائد.

مجالات p99 ضيقة دون التشبع؛ وأوسعها p99 للوكيل العكسي عند 50,000 طلب في الثانية، من 58 إلى 66 us. عمود الأقصى طلب واحد من بين ما يصل إلى 1.5 مليون في التشغيل. وعلى مضيف مشترك نقرؤه على أنه أسوأ لحظة مرّ بها المضيف، لا خاصيةً للبرنامج الخدمي.

ما الذي غيَّره القياس

غيَّر الحمل الخارجي تصميم Sibuna. يخدم البرنامج الخدمي كل اتصال على خيط خاص به (SID 0002، SID 0010)، ووجدت أولى تشغيلاته لقياس زمن الاستجابة ذيلًا لم يُظهره `wrk`. مع 64 اتصالًا، يغلق البرنامج الخدمي كلًّا منها بعد 4,096 طلبًا، كان المئين 99.9 عند 100,000 طلب مقبول في الثانية من 0.3 إلى 2 ms.

التشخيص، المسجل في SID 0002 وفي وثائق `cmd/sibuna/src/tasks.rs`، هو تبدُّل الخيوط. إنشاء خيط وإنهاؤه يربط مكدس إشاراته ويفك ربطه ويعيد صفحات مكدسه، وكانت تغييرات فضاء العناوين هذه توقف خيوط البرنامج الخدمي الأخرى. حسمت تجربة واحدة الأمر: بناءً لا يغلق الاتصالات أبدًا، فلا يُنشأ أي خيط أثناء التشغيل، أعطى من 54 إلى 60 us عند المعدل نفسه. يعيد الإصلاح استخدام خيوط الاتصالات: الخيط الذي ينهي اتصاله يتوقف منتظرًا ويأخذ الاتصال التالي، فلا ينشئ تدفق ثابت من الاتصالات الجديدة أي خيوط. والخيط المنتظر 10 ثوانٍ ينتهي ويُضم عند الإطلاق التالي أو عند انتهاء الخيط التالي من مهمة، فلا تترك الدفعة مكدساتها مقيمة. والخيط الذي يُصاب اتصاله بـ panic ينتهي بدل أن ينتظر. ويُخدم `--max-connections` اتصالًا على الأكثر في وقت واحد، كما من قبل.

الجداول أعلاه من التشغيل بعد الإصلاح: p99.9 قدره 62 us في وضع forward-auth و150 us في وضع الوكيل العكسي عند 100,000 طلب في الثانية. لم تُحفظ تشغيلات ما قبل الإصلاح ملفات نتائج، ولهذا لا تظهر أرقامها إلا في SID 0002 وفي هذا القسم.

مثال محلول: لماذا ظهر التوقف عند p99.9 عند 100,000 طلب في الثانية عبر 64 اتصالًا، يحمل كل اتصال $100000/64 = 1562.5$ طلبًا في الثانية. يغلق البرنامج الخدمي اتصال keep-alive بعد 4,096 طلبًا (`MAX_REQUESTS_PER_CONNECTION` في `cmd/sibuna/src/connection.rs`)، فيدوم الاتصال $4096/1562.5 \approx 2.62$ s، ويفتح المولد نحو $64/2.62 \approx 24.4$ اتصالًا جديدًا في الثانية. ومع خيط لكل اتصال، يعني ذلك نحو 24 بدء خيط و24 انتهاء خيط كل ثانية.

يقيس تشغيل مدته 10 ثوانٍ عند هذا المعدل 1,000,000 طلب. ويتجاوز قيمةَ p99.9 فيه 1,000 طلب منها، أي 100 في الثانية؛ ويتجاوز قيمةَ p99 فيه 1,000 طلب في الثانية. إذا أخَّر كل تبدل نحو $100/24.4 \approx 4$ طلبات قيد التنفيذ على خيوط أخرى بجزء من الميلي ثانية، فإن الطلبات المتأخرة وحدها تملأ أعلى 0.1%، بينما يظل أعلى 1% مكونًا من الاصطفاف العادي. التوقف النادر الدوري يحرك الذيل البعيد قبل أن يحرك p99 بوقت طويل. ولهذا يحتاج مثل هذا الخلل إلى p99.9 ومؤقت على كل طلب حتى يُرى.

Sibuna وAnubis

الأهداف اقرأ أرقام Sibuna بجانب أرقام منتج راسخ تحت الحمل نفسه، وبيِّن أي الفروق تأتي من المنتجين وأيها من الإعداد.

Anubis، من Techaro، بوابة إثبات عمل واسعة الانتشار تقف أمام مواقع الويب، كما يفعل Gate في Sibuna. قسنا الملف التنفيذي الرسمي لإصدار Anubis v1.27.0 (`anubis-1.27.0-linux-amd64`، 40,001,698 بايت) بجانب Sibuna على المضيف نفسه، وعلى وحدات CPU نفسها، وبالأعباء نفسها. سُجل ملفا النتائج بالأمرين:

```
cargo xtask bench product --anubis <path/to/anubis>
cargo xtask bench admission --anubis <path/to/anubis>
```

شُغِّل Anubis كما كانت تشغّله سكربتات القياس السابقة المكتوبة بـ Python في هذا المشروع: `--bind`، و`--metrics-bind`، و`--target=` (فارغ في وضع forward-auth، والخادم الأصلي في وضع الوكيل العكسي)، و`--difficulty 2`، و`--slog-level ERROR`، و`--cookie-secure=false`، وسياسة تتحدى كل User-Agent من نوع `Mozilla` مع تعطيل DNSBL والمصيدة. `GOMAXPROCS` يساوي عدد عمال Sibuna، وكل خيط من أي من المنتجين مثبت على مجموعة CPU نفسها. عمل Hashcash متكافئ: 8 بتات صفرية لـ Sibuna، ورقمان ست عشريان صفريان لـ Anubis، والمتوقع في الحالتين 256 تجزئة. تحصل أداة القياس على جلسة كل منتج كما يفعل المتصفح. ففي Anubis تجلب صفحة التحدي، وتقرأ التحدي من `<script id="anubis_challenge">`، وتجد القيمة المؤقتة التي يبدأ معها `SHA-256(randomData || nonce)` بـ `00`، وتمررها إلى `pass-challenge` مع ملف تعريف ارتباط التحقق الخاص بالصفحة. وفي وضع forward-auth يسأل عبئا `admitted` و`challenged` نقطة نهاية الفحص في Anubis (`/.within.website/x/cmd/anubis/api/check`).

أعباء المنتجات الكاملة

الأعباء الأربعة والحمل والإحصاءات هي نفسها في أداة قياس المنتجات الكاملة، مع تغيير واحد: الخادم الأصلي هو `caddy respond`، مثبتًا على وحدات CPU الخاصة بالحمل، كما كانت السكربتات السابقة تفعل. قبل كل تشغيل تنتظر أداة القياس حتى يبقى أقل من 3,000 مقبس في `TIME_WAIT`، لأن تشغيلات الوكيل العكسي لـ Anubis تترك عشرات الآلاف منها.

الجدول 18 وضع forward-auth: وسيطات ثلاثة تشغيلات لـ `wrk` مدة كل منها خمس ثوانٍ (`benchmarks/results/comparison-latest.md`، أُنشئ في 2026-10-16T00:55:19Z).

عبء العمل	المقياس	Sibuna Gate	Sibuna Shield	Anubis 1.27.0
`admitted`	req/s (الحالة)	(200) 210,682	(200) 208,635	(200) 36,509
	p99 us	305	308	7,477
	CPU us/req	12.09	13.21	106.30
	ذروة RSS KiB	16,228	16,316	32,624
`challenged`	req/s (الحالة)	(401) 198,251	(401) 193,087	(401) 113,170
	p99 us	326	342	3,781
	CPU us/req	13.49	15.02	32.96

عبء العمل	المقياس	Sibuna Gate	Sibuna Shield	Anubis 1.27.0
	ذروة RSS KiB	16,240	16,328	32,696
allowed_static	req/s (الحالة)	(200) 212,569	(200) 212,444	(200) 89,904
	p99 us	304	304	4,547
	CPU us/req	11.73	11.95	41.63
	ذروة RSS KiB	16,240	16,328	33,396
attack	req/s (الحالة)	(200) 212,086	(403) 211,347	(200) 36,853
	p99 us	305	306	7,283
	CPU us/req	12.07	12.56	104.88
	ذروة RSS KiB	16,240	16,328	33,764

الجدول 19 وضع الوكيل العكسي، خلف `caddy respond`: وسيطات ثلاثة تشغيلات لـ `wrk` مدة كل منها خمس ثوانٍ (`benchmarks/results/comparison-latest.md`، أُنشئ في 2026-10-16T00:55:19Z).

عبء العمل	المقياس	Sibuna Gate	Sibuna Shield	Anubis 1.27.0
admitted	req/s (الحالة)	(200) 106,376	(200) 103,298	(200) 19,398
	p99 us	3,331	3,321	11,062
	CPU us/req	27.02	27.92	190.40
	ذروة RSS KiB	16,528	16,444	39,980
challenged	req/s (الحالة)	(200) 198,402	(200) 197,327	(200) 28,842
	p99 us	325	328	9,546
	CPU us/req	13.05	13.78	137.73
	ذروة RSS KiB	16,544	16,464	516,872
allowed_static	req/s (الحالة)	(200) 105,427	(200) 106,819	(200) 28,862
	p99 us	3,320	3,316	30,124
	CPU us/req	26.24	26.52	135.41
	ذروة RSS KiB	16,544	16,464	616,548
attack	req/s (الحالة)	(200) 105,616	(403) 210,483	(200) 23,399
	p99 us	3,327	306	14,124

عبء العمل	المقياس	Sibuna Gate	Sibuna Shield	Anubis 1.27.0
	CPU us/req	26.98	12.28	169.04
	ذروة RSS KiB	16,544	16,464	599,516

في وضع forward-auth خدم Sibuna من 193,087 إلى 212,569 طلبًا في الثانية مع p99 من 304 إلى 342 us، وخدم Anubis من 36,509 (`admitted`) إلى 113,170 (`challenged`) مع p99 من 3,781 إلى 7,477 us. بقيت الأنوية المشغولة في Sibuna بين 2.49 و2.90 من أربع، فوضع مولد الحمل حده؛ وكانت في Anubis من 3.73 إلى 3.88، فكان Anubis محدودًا بوحدة CPU. الرقم الأوضح هو وقت CPU لكل طلب: كلف الطلب المقبول Sibuna Gate 12.09 us وAnubis 106.30 us. ومن الأجزاء المرجحة لهذا الفرق رمز الجلسة. يوقع Anubis الجلسات بـ Ed25519 افتراضيًا، ويفحص Sibuna رمز BLAKE3 MAC ذا مفتاح (155.8 ns في جدول العمليات الأولية)؛ وتدعم هذه القراءةَ قياساتُ عمليات القبول أدناه، حيث ترفع رموز HS512 فحوص الجلسات في Anubis بأكثر من النصف.

في وضع الوكيل العكسي خدمت صفوف Sibuna الممررة من 103,298 إلى 106,819 طلبًا في الثانية بأنوية مشغولة من 2.77 إلى 2.98. وهي دون الأرقام 138,965 إلى 147,303 للخادم الأصلي التجريبي في `product-latest.json`، والبرنامج الخدمي ليس محدودًا بوحدة CPU، لذا يقع الحد خارجه: الخادم الأصلي، `caddy respond`، يشارك `wrk` وحدات CPU الخاصة بالحمل. ويظهر ذيله في p90 من 2.1 إلى 2.4 ms. خدم Anubis من 19,398 إلى 28,862 طلبًا في الثانية بأنوية مشغولة من 3.68 إلى 3.97 ووقت CPU من 135 إلى 190 us لكل طلب. ارتفعت ذاكرته المقيمة من 39,980 KiB في عبء `admitted` إلى 516,872 KiB في `challenged` و616,548 KiB في `allowed_static`، واحتفظ بـ 589,276 KiB بعد الأعباء مقابل 21,552 KiB في الخمول. بدأ الارتفاع في `challenged`، حيث يتلقى كل طلب تحديًا جديدًا بمعرف خاص به؛ ويوحي ذلك بسجل يُحفظ لكل تحدٍّ صادر، لكن أداة القياس لم تؤكد السبب. وبقيت ذروة Sibuna بين 16,228 و16,544 KiB في كل صف من الوضعين.

وليست المقارنة متكافئة في صفوف عدة أيضًا:

- في وضع forward-auth يجيب Anubis عن `/robots.txt` وعن مسار `attack` بنفسه بـ `200`، بلا خادم أصلي؛ ويجيب Sibuna عنهما كما يجيب عن أي طلب في ذلك الوضع.
- لا يفحص Anubis الطلبات، لذا يقبل حقن SQL في عبء `attack`. يرفضه Shield بـ `403`، بعد عمل أكثر؛ ويقبله Gate، كما يفعل Anubis.
- يوقع المنتجان الجلسات بطريقتين مختلفتين (Ed25519 في Anubis افتراضيًا، وBLAKE3 MAC ذو مفتاح في Sibuna)، ويعرضان صفحتي تحدٍّ مختلفتين.

عمليات القبول

توقِّت أداة قياس القبول العمليات الأربع التي يتكون منها القبول، في وضع forward-auth، مع كل منتج على وحدتي CPU (عاملان في Sibuna، و`GOMAXPROCS=2`). يتشارك خيطا عميل على وحدات CPU الخاصة بالحمل، لكل منهما اتصال keep-alive واحد، كل دفعة: 200 عملية، أو 64 إثباتًا للتحقق، تُعد قبل الدفعة. نعرض وسيط سبع دفعات.

- فحص الجلسة: نقطة نهاية الفحص مع جلسة صالحة (200).
- فحص بلا مصادقة: الطلب نفسه بلا ملف تعريف ارتباط (401).
- تمهيد التحدي: ما يجلبه المتصفح لبدء تحدٍّ. في Sibuna هما طلبان، الصفحة الوسيطة وسجل التحدي؛ وفي Anubis طلب واحد، الصفحة مع التحدي المضمن.
- التحقق من الإثبات: إثبات جديد واحد لكل عملية. تصدر تحديات الإثبات في Sibuna بفاصل 25 ms، أي دون معدل 50 في الثانية الذي تبدأ عنده صعوبته التكيفية بإضافة بتات، فيبقى العمل متكافئًا. يعمل Sibuna مع `--challenge-rate-limit 100000000`، فيُفحص محدده لكنه لا يُستنفد أبدًا.

شُغِّل Anubis مرتين، مع رموز Ed25519 الافتراضية ومع `--hs512-secret`.

الجدول 20 عمليات القبول في الثانية، عميلان، والمنتج على وحدتي CPU (`benchmarks/results/admission-latest.md`، أُنشئ في 2026-10-16T00:55:44Z)

المنتج (الرمز)	فحص الجلسة	فحص بلا مصادقة	تمهيد التحدي	التحقق من الإثبات	RSS بعد القياس، KiB
Sibuna (BLAKE3 ذو مفتاح)	88,104	78,997	42,664	57,924	7,792
Anubis 1.27.0 (Ed25519)	10,951	28,619	7,893	12,668	33,400
Anubis 1.27.0 (HS512)	16,976	27,785	7,952	19,261	29,840

مجالات الدفعات السبع في الملف. تقع مجالات Sibuna ضمن 11% من وسيطاته؛ ومجالات Anubis أوسع (تمتد فحوصه بلا مصادقة مع رموز Ed25519 من 25,394 إلى 32,741)، والدفعة من 200 عملية لا تدوم إلا من 2 إلى 25 ms، لذا نقرأ هذه الأرقام بدقة العُشر تقريبًا. يعطي زمن العميل الأبطأ لكل عملية زمن الذهاب والإياب: 22.7 us لفحص جلسة في Sibuna، و182.5 us لفحص في Anubis مع رموز Ed25519، و117.7 us مع HS512.

لم يخدم Anubis في أيٍّ من الصفوف أعلاه طلباتٍ أكثر مما خدم Sibuna، ولم يستهلك وقت CPU أو ذاكرة أقل منه. هذا حكم على هذه الأعباء على هذا المضيف، لا أكثر.

> **قراءة صفوف Anubis** Anubis منتج متمكن وواسع الانتشار، وهذه الصفوف ليست ادعاءً بأنه بطيء. يعمل على بيئة تشغيل بجمع نفايات، ويوقع الجلسات بـ Ed25519 افتراضيًا حتى لا يحتاج التحقق إلى سر مشترك، ويوفر عدة أساليب تحدٍّ ولغة سياسات غنية، ويحفظ حالة التحديات في مخزن؛ وهذه خيارات تصميم لها فوائد لا تقيسها أداة القياس هذه. تبين الصفوف ما كلفه الطلب كل منتج على مضيف واحد تحت شكل حمل واحد، في أكتوبر 2026. Sibuna جديد، وهذه أولى قياساته بجانب منتج راسخ.

التكلفة الإضافية لوحدة التحكم

يضع SID 0007 عقد عزل وحدة التحكم. قد يكلف تفعيل وحدة التحكم 1% على الأكثر من إنتاجية مستوى البيانات و10% على الأكثر من زمن الاستجابة p99، مقيسًا على مصفوفة من الإعدادات (مستبعدة من البناء، ومضمنة فيه لكنها معطلة، وخاملة، وثماني لوحات معلومات نشطة) والأعباء، مع تشغيلات متكررة متداخلة وعدم يقين معلن. لا تثبت التشغيلات غير الحاسمة أو كثيرة الضجيج الامتثال، ولا تغني خطوط الأساس للعمليات الأولية وتشغيلات الفحص السريع عن معيار الاجتياز الرسمي.

يسجل SID 0010 قياسًا واحدًا: مع تضمين وحدة التحكم في بناء البرنامج الخدمي، سواء كانت معطلة، أو للتخزين فقط، أو مفعلة وخاملة، تبقى أعباء المنتجات الـ 16 ضمن نحو 1% من البرنامج الخدمي المبني من دونها، وهذا ضمن الضجيج. لا يذكر السجل ملف نتائج له. لم يُقس الإعداد ذو لوحات المعلومات النشطة الثماني تحت الحمل، لذا يبقى معيار الاجتياز الرسمي في SID 0007 مفتوحًا.

يفحص تشغيل لوحات المعلومات الثماني الوظيفة، لا التكلفة الإضافية. بدأ `sibuna-browser dashboards` برنامجًا خدميًا من بناء الإصدار على عقدة SQLite واحدة مع وحدة تحكمها، وأربعة عمال، وسياسة تسمح بـ `/static/*` و `/about` و`/missing/*` و`/broken/*` وتتحدى كل ما سواها؛ ويجيب خادم أصلي تجريبي عن `/missing/*` بـ 404 وعن `/broken/*` بـ 500. أرسل مولد 200 طلب في الثانية لمدة 30 ثانية، متناوبًا بين عشرة مسارات (أحدها حقن SQL)، وثلاث قيم User-Agent، و500 عنوان ممرر. عملت كل لوحة في Chromium خاص بها بجلستها الخاصة: اللوحات من 1 إلى 3 جلسات مسؤول، واللوحات من 4 إلى 8 شاشات حائط فُتحت برموز أصدرتها اللوحة الأولى، لأن وحدة التحكم تسمح بخمس عمليات تسجيل دخول في الدقيقة لكل عنوان ولكل حساب.

الجدول 21 ثماني لوحات معلومات Traffic على عقدة واحدة (`benchmarks/results/console-8-dashboards-2026-10-15.md`، سُجل في 15-10-2026، Chromium 155.0.8059.39). النتيجة: PASS.

المقياس	القيمة
طلبات مستوى البيانات	6,000 في 30.0 s (200 في الثانية، 4 خيوط، اتصال واحد لكل طلب)
المجاب عنها	6,000: 200 x 600، 404 x 600، 403 x 600، 401 x 3,000، 500 x 1,200
الأخطاء (بلا رد)	0
زمن الاستجابة p99 / p90 / p50 / الأقصى	246 / 311 / 373 / 552 us (بما في ذلك زمن الاتصال)
VmRSS للبرنامج الخدمي عند البدء	36,152 KiB
VmRSS للبرنامج الخدمي مع فتح 8 لوحات معلومات	38,028 KiB

المقياس	القيمة
VmRSS للبرنامج الخدمي بعد حركة المرور	41,496 KiB (VmHWM 41,496 KiB)

قُرئت البطاقات كل ثانيتين أثناء حركة المرور، وبعدها بـ 3 ثوانٍ. انتهت كل لوحة بالمجاميع نفسها (Requests 6,000، Admitted 2,400، Challenged 3,000، Policy denied 600، Origin 4xx 600، Origin 5xx 600)، وتلقت من 134 إلى 139 فرقًا من `stats`، وأبقت WebSocket مفتوحًا، ولم تبلغ عن أي استثناء أو خطأ في وحدة تحكم المتصفح أو انتهاك لـ Content-Security-Policy. ثمانية مشتركين من المتصفحات أقل بكثير من حد وحدة التحكم البالغ 64. البرنامج الخدمي في هذا التشغيل فيه تخزين SQLite ووحدة التحكم، بخلاف البرامج الخدمية العاملة في الذاكرة في أداة قياس المنتجات، لذا لا تُقارن ذاكرته المقيمة بذاكرتها.

ما لم يُقس

الجدول 22 ما لا يقيسه هذا الجزء، وما يقوم مقام القياس

الموضوع	لماذا لا يُقاس هنا	ما يقوم مقامه
الإصدار المجتمعي من SafeLine 9.x	لا يُوزع إلا حزمة Docker Compose؛ ولم يكن على المضيف برنامج Docker الخدمي، وصورة الكاشف ليست بناءً مفتوح المصدر يمكن بناؤه هناك	حقائق منشورة يعلنها المورد (فُحصت في سبتمبر 2026): معدل الكشف 71.65% (balance) و76.17% (strict)، والإيجابيات الكاذبة 0.07% (balance) و0.22% (strict)؛ والحد الأدنى للمضيف Linux x86_64 أو arm64 مع SSSE3، ونواة CPU واحدة، و1 GB من RAM، و5 GB من القرص، وDocker 20.10.14+ وCompose 2.0+
Cloudflare WAF	خدمة مستضافة؛ وقياس loopback سيقيس مسار الشبكة، لا WAF	حصص الخطط المنشورة (فُحصت في سبتمبر 2026): القواعد المخصصة Free 5، وPro 20، وBusiness 100، وEnterprise 1000؛ وقواعد حد المعدل Free 1، وPro 2، وBusiness 5، وEnterprise 100

الموضوع	لماذا لا يُقاس هنا	ما يقوم مقامه
عنقود تحت الحمل	لا توجد بعد أداة قياس تُحمِّل عدة عقد تشترك في قاعدة بيانات PostgreSQL واحدة	خصائص البروتوكول التي يحددها SID 0011 (بذرة مشتركة واحدة، وتحديات مرتبطة بمُصدرها) واختباره الوظيفي على ثلاث عقد (المعيار 2)؛ بلا قياس
التخزين على مسار الطلب	يعمل البرنامجان الخدميان كلاهما في الذاكرة في كل أداة قياس	SID 0011: خيط تخزين واحد يملك كل SQL، ولا ينتظره مسار الطلب أبدًا
TLS وWAN والمتصفحات الحقيقية	تستخدم أدوات القياس loopback وHTTP؛ وتُحل الجلسات مرة لكل حالة بـ `sibuna-pow`	لا شيء؛ يقيس كل نشر مساره الخاص
زمن الحل في برنامج حل الألغاز في المتصفح	تحل أداة القياس عند 8 بتات ببرنامج الحل الأصلي	حجم الوحدة أعلاه؛ ويشرح الجزء السابع لماذا لا تمثل التوقيتات الأصلية أزمنة الحل في المتصفح
أعداد التخصيصات	لا تُجهِّز أداة القياس المُخصِّص بأدوات رصد	اختبارات الحالة المستقرة في الجزء الخامس
عزل وحدة التحكم مع ثماني لوحات معلومات نشطة تحت الحمل	لم يُشغَّل معيار الاجتياز الرسمي في SID 0007	رقم SID 0010 (ضمن نحو 1%، ووحدة التحكم خاملة) والتشغيل الوظيفي أعلاه

لا يمثل أي صف في هذا الجدول ادعاءً بشأن أداء المنتج. تتعلق أرقام SafeLine نفسها بجودة الكشف، لا بالإنتاجية، وينشر Cloudflare حصص الخطط لا تكلفة كل طلب. يستطيع أي أحد لديه مضيف Docker تشغيل SafeLine خلف أعباء `wrk` نفسها.

تمرين 8.1. شغِّل `cargo xtask bench primitives` على جهاز آخر وقارن صفي تحقق Hashcash وتحقق الرمز بالمضيف المرجعي. أي الاثنين يعتمد أكثر على تعليمات التجزئة في العتاد، ولماذا؟

تمرين 8.2. في جدول forward-auth، احسب الأنوية المشغولة لصفٍّ: ميكروثواني CPU لكل طلب مضروبة في عدد طلباته في الثانية، مقسومة على مليون. هل كان البرنامج الخدمي محدودًا بوحدة CPU؟ ثم قارن ميكروثواني CPU لكل طلب بتكلفة التصنيف في الجدول الأول. لماذا تكون تكلفة CPU لكل طلب أعلى بكثير، وأي المكونات تفسر الفرق؟

تلميح: اسرد كل ما يقع بين `accept` وتفريغ الاستجابة (flush) مما لا توقِّته مجموعة العمليات الأولية.

اشرح اللامتغير. كل رقم في هذا الجزء وسيط عبر التكرارات، ومجاله بجانبه. اشرح الفرق بين رقم هو وسيط دفعات وبين $p99.9$ لكل طلب، ولماذا يعرض جدول العمليات الأولية الأول وجدول زمن الاستجابة الثاني، ولماذا لا يكشف إلا الثاني توقف تبدل الخيوط الموصوف أعلاه.

الجزء التاسع

التشغيل والنشر

نغطي الوضعين، وكل خيار من خيارات سطر الأوامر، ووصفات *forward auth*، وملفات السياسة، والتخزين الدائم في *SQLite* أو *PostgreSQL*، والعنقدة فوق قاعدة بيانات مشتركة واحدة، والتحزيم.

الأوضاع والإعداد

الأهداف بنهاية هذا الفصل، ينبغي أن نكون قادرين على تشغيل Sibuna وكيلًا عكسيًا أو خدمة تحقق في وضع forward auth، واختيار الوضع، وضبط مستوى إثبات العمل وصعوبته، وإدارة السر الرئيسي.

اختيار الوضع

- *Gate* (`--gate`، والاسم البديل `--no-waf`): القبول بإثبات العمل، والجلسات، والقواعد التصريحية، والسمعة، وحدود GCRA، والحظر.
- *Shield* (الافتراضي، `--shield`، والاسم البديل `--waf`): Gate مع الجدار الناري الدلالي.
- النشر الموزع (خيار متاح لكلا الوضعين): يحفظ `--data-dir` السياسات والسمعة وأدلة الحوادث في ملف SQLite محلي؛ ويربط `--database-url` مع `--cluster-node` عدة عقد بقاعدة بيانات PostgreSQL واحدة (SID 0011).

مرجع سطر الأوامر

يسرد `sibuna --help` الجدول كل خيارات. النطاقات المذكورة هي التي يفرضها المحلل.

الجدول 23 خيارات البرنامج الخدمي

الخيار	الافتراضي	المعنى
`--port, -p`	`8080`	منفذ الاستماع، من 1 إلى 65535
`--host, -h`	`0.0.0.0`	عنوان الاستماع، ويُكتب عنوان IP صريحًا
`--upstream-host`	`127.0.0.1`	مضيف الخادم الأصلي (وضع الوكيل العكسي)
`--upstream-port, -u`	`3000`	منفذ الخادم الأصلي

الخيار	الافتراضي	المعنى
`--mode, -m`	`reverse_proxy`	`reverse_proxy` أو `forward_auth` (وأيضًا `proxy` و `auth`)، ويُعطى مرة واحدة
`--workers, -w`	واحد لكل معالج	خيوط استقبال الاتصالات التي تتشارك مقبس الاستماع، من 0 (واحد لكل معالج) إلى 64؛ ثم يحصل كل اتصال مُستقبَل على خيط خاص به
`--max-connections`	`1024`	الاتصالات المخدومة في وقت واحد، من 1 إلى 8192؛ ويُجاب ما زاد عليها بـ `503`
`--max-connections-per-ip`	`64`	الاتصالات التي يمكن أن يحتفظ بها عنوان نظير واحد على المقبس، من 0 (بلا حد) إلى 8192؛ يُحتسب نظير IPv6 بشبكته ذات البادئة `--ipv6-prefix`، ولا دور للترويسات الممررة؛ ويُجاب ما زاد عليها بـ `503`
`--idle-timeout`	`15`	أطول صمت بالثواني، من 0 إلى 86400. يجب أن يصل رأس الطلب خلاله. تجدده استجابة الخادم الأصلي عند كل قراءة، فيبقى التدفق البطيء حيًا ما دامت البايتات تتدفق، ويُقطع الخادم الأصلي الصامت على المقبسين كليهما. يجب أن يوصل الرفع الممرر 16 KiB في كل مدة (حد أدنى للمعدل في وجه هجمات الجسم البطيء). ويحمل مقبس العميل مهلة كتابة بالطول نفسه، فالعميل الذي لا يقبل أي بايت طوال تلك المدة يفقد اتصاله
`--websocket-idle-timeout`	`300`	مهلة الخمول للاتصال المرقّى بالثواني، من 0 إلى 86400
`--trust-forwarded`	معطل؛ مفعل في forward auth	الأخذ بالترويستين `X-Forwarded-For` / `X-Real-IP` القادمتين من النظير
`--no-trust-forwarded`	معطل	عدم الأخذ بهما أبدًا، حتى في وضع forward auth
`--trusted-hops`	`1`	عندما تكون الهوية الممررة موثوقة، يكون العميل هو مدخل `X-Forwarded-For` الذي يبعد هذا العدد من المواضع من اليمين، من 1 إلى 8
`--ja4-header`	معطل	ترويسة الطلب التي يرسل فيها وكيل دخول موثوق بصمة TLS للعميل من نوع JA4، مثل `cf-ja4`؛ من 1 إلى 64

الخيار	الافتراضي	المعنى
		محرفًا من محارف token، وليست `X-Forwarded-*` أو `X-Real-IP` أو `X-Sibuna-*` أو `Forwarded`. لا تُقرأ إلا عندما تكون الهوية الممررة موثوقة (انظر "بصمات JA4 من وكيل الدخول")
`--ipv6-prefix`	`64`	طول بادئة IPv6، من 32 إلى 128، الذي تنطبق عليه حدود المعدل والحظر وحد الاتصالات لكل عميل
`--algorithm, -a`	`posw`	`posw` أو `hashcash`
`--difficulty, -d`	`16`	بتات العمل، من 1 إلى 64: البتات الصفرية في Hashcash، أو عمق PoSW مضافًا إليه ثلاثة
`--posw-challenges`	`16`	عدد عمليات الفتح في كل إثبات عمل متسلسل، من 1 إلى 32
`--token-scheme`	`mac`	`mac` (BLAKE3 ذو مفتاح) أو `ed25519`
`--token-ttl`	`86400`	عمر الجلسة بالثواني (من 1 إلى 31,536,000)
`--challenge-ttl`	`300`	عمر التحدي بالثواني (من 1 إلى 86,400)
`--secret-file, -s`	عشوائي	64 محرفًا سداسيًا عشريًا أو 32 بايتًا خامًا؛ ويُقبل أيضًا `SIBUNA_SECRET` (64 محرفًا سداسيًا عشريًا)
`--cookie-name`	`__sibuna_token`	اسم ملف تعريف الارتباط للجلسة، من 1 إلى 64 محرفًا من محارف token
`--secure-cookie`	معطل	إضافة السمة `Secure`
`--shield` / `--gate`	shield	اختيار الوضع
`--rate-limit`	`100`	الطلبات في كل نافذة لكل عميل (دفعة GCRA)
`--rate-window`	`10`	النافذة بالثواني، من 1 إلى 86400
`--challenge-rate-limit`	`30`	عمليات إصدار التحديات والتحقق منها في كل نافذة لكل عميل، منفصلة عن ميزانية الطلبات
`--ban-seconds`	`3600`	مدة حظر المصيدة، من 0 إلى 31,536,000
`--policy-file, -P`	لا شيء	سياسة JSON تصريحية

الخيار	الافتراضي	المعنى
`--rate-table-cells`	`262144`	الخلايا في كل جدول من جداول محدد المعدل، 16 بايتًا لكل منها، من 4,096 إلى 16,777,216، وتُقرّب صعودًا إلى قوة للعدد اثنين
`--ban-table-slots`	`65536`	مواضع جدول الحظر، 24 بايتًا لكل منها، من 1,024 إلى 16,777,216، وتُقرّب صعودًا إلى قوة للعدد اثنين
`--spent-tags`	`1048576`	التحديات المحلولة المحفوظة لمنع إعادة استخدامها، 24 بايتًا لكل منها، من 65,536 إلى 16,777,216
`--data-dir, -D`	لا شيء	عقدة واحدة: الحفظ في ملف SQLite `<path>/sibuna.db`
`--database-url`	لا شيء	عنقود: قاعدة بيانات PostgreSQL المشتركة (`postgres://` أو `postgresql://`)
`--database-url-file`	لا شيء	قراءة عنوان URL لقاعدة البيانات من ملف لا يقرؤه إلا مالكه
`--cluster-node`	لا شيء	معرف هذه العقدة؛ يرتبط بمفاتيح التحديات وبالبتات العليا من معرفات الحوادث؛ وهو مطلوب مع عنوان URL لقاعدة البيانات
`--storage-poll-ms`	`500`	فاصل نبضة التخزين، من 1 إلى 60,000 ms
`--verbose, -v`	معطل	تسجيل مفصل
`--version` / `--help`		طباعة المساعدة (خيارات وحدة التحكم أولًا، عندما تكون وحدة التحكم مضمنة في البناء) أو الإصدار، ثم الخروج

يستبعد `--data-dir` وعنوان URL لقاعدة البيانات أحدهما الآخر، ويستبعد `--database-url` و `--database-url-file` أحدهما الآخر. يُرفض عنوان URL لقاعدة البيانات من دون `--cluster-node`. أُزيلت خيارات عنقود Zaxonlite وهي `--cluster-listen` و`--cluster-peer` و`--cluster-secret-file` و `--cluster-tls-cert` و`--cluster-tls-key` و`--cluster-tls-ca`؛ وإعطاء أي منها يوقف بدء التشغيل بالخطأ `RemovedOption` مع تلميح يسمي SID 0011. تُسرد خيارات وحدة التحكم في الفصل التالي.

> **السر الرئيسي** من دون `--secret-file` أو `SIBUNA_SECRET` يسحب البرنامج الخدمي بذرة عشوائية، ويطبع رسالة البدء "random (set –secret-file)". عندئذ تموت الرموز والتحديات بموت العملية، وترفض

عقد العنقود التي تحمل بذورًا مختلفة ملفات تعريف الارتباط الصادرة بعضها عن بعض. ولّد سرًا بالأمر `head -c 32 /dev/urandom | xxd -p -c 64 > /etc/sibuna/secret` مع الصلاحيات 0600.

لمحة عن وحدة التحكم

تُضمَّن وحدة التحكم في البرنامج الخدمي عبر الميزة الافتراضية `console`، لكنها لا تبدأ إلا مع `--console`. وهي تحتاج إلى التخزين: يُرفض `--console` من دون `--data-dir` أو `--database-url` بالخطأ `CONSOLE001` (`StorageRequired`). استخدم `cargo build --release -p sibuna --no-default-features --features sqlite` لإخراجها من البناء؛ أما البناء بـ `--no-default-features` وحده فليس فيه تخزين ولا وحدة تحكم. هيّئها محليًا والبرنامج الخدمي متوقف:

```
sibuna init-admin admin --data-dir ./data
sibuna --data-dir ./data --console 127.0.0.1:19446
```

الأمر `sibuna console init-admin` هو الأمر نفسه. افتح `http://127.0.0.1:19446/console/` واستبدل كلمة المرور المؤقتة المولدة، التي يجب تغييرها عند أول تسجيل دخول وتنتهي صلاحيتها بعد ساعة واحدة. يوفر التنقل بعد تسجيل الدخول الكرة الأرضية المتحركة للبلدان، وإحصاءات الطلبات والتحديات، والتحقيق في الحوادث، ومحرري السياسة والفحص، والمستخدمين، ورموز API محددة النطاق، والتدقيق، وأدوات التحكم في العقدة الخادمة. لا تحمّل واجهة المصادقة أي هندسة للكرة الأرضية ولا أي قياس عن بعد. الواجهة وحدة WebAssembly يبنيها سكربت البناء الخاص بالبرنامج الخدمي من `cmd/console-ui` (الحزمة `sibuna-console-ui`)؛ وتُشحن هي وورقة الأنماط المودَعة في المستودع مع البناءات العادية.

الجدول 24 خيارات وحدة التحكم

الخيار	المعنى
`--console <ip:port>`	بدء وحدة التحكم على هذا المستمع (يُكتب IPv6 بين قوسين معقوفين، والمنفذ غير صفري)؛ ويتطلب `--data-dir` أو `--database-url`
`--console-key-file <path>`	64 محرفًا سداسيًا عشريًا، وأذونات للمالك وحده؛ يحمي أسرار العامل الثاني وأسرار webhook المخزنة
`--console-origin <https-origin>`	أصل HTTPS القياسي خلف وكيل
`--console-behind-proxy`	كل طلب يصل عبر وكيل ينهي TLS

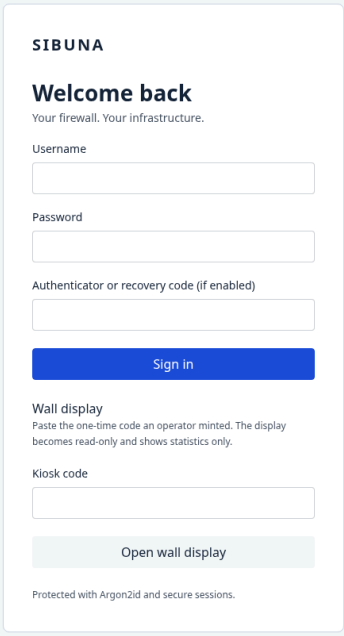
الخيار	المعنى
`--console-trusted-proxy <cidr>`	شبكة يجوز أن تحمل اتصالاتها حقولًا ممررة؛ قابل للتكرار، 16 على الأكثر
`--console-advertise <origin>`	الرابط الذي تعرضه وحدات التحكم الأخرى لهذه الوحدة
`--console-location <latitude,longitude>`	الموقع المعلن لهذه العقدة على الكرة الأرضية
`--console-probe <node-id>=<http://ip:port>`	مستمع مستوى البيانات لدى قرين تُفحص صحته؛ قابل للتكرار، ثمانية على الأكثر
`--console-peer <node-id>=<https://origin>`	قرين إدارة؛ قابل للتكرار، ثمانية على الأكثر
`--console-peer-key-file <path>`	مفتاح الأقران: للمالك وحده، 64 محرفًا سداسيًا عشريًا، مستقل عن مفتاح وحدة التحكم
`--console-peer-ca-file <path>`	مراسي ثقة PEM اختيارية للأقران، تحل محل جذور النظام
`--console-query-steps <n>`	خطوات SQLite لكل قراءة تجميعية، من 100,000 إلى 50,000,000، والافتراضي 4,000,000
`--console-capture-heads`	يخزن رؤوس الطلب ورؤوس استجابة الخادم الأصلي منقحةً لكل حادث؛ معطل افتراضيًا
`--console-capture-header <name>`	يحتفظ بقيمة هذه الترويسة في الرؤوس المخزنة؛ قابل للتكرار، 16 على الأكثر؛ يتطلب `--console-capture-heads`
`--console-cookie-secure`	يمنح ملفات تعريف الارتباط لوحدة التحكم السمة Secure من دون وكيل (يقبله المحلل، لكنه غير مدرج في `--help`)

خيار لوحدة التحكم من دون `--console`، أو خيار مكرر، أو قيمة معيبة الصيغة يوقف بدء التشغيل بالرمز `CONSOLE002` واسم خطأ الخيار. والإعداد الذي يخالف قاعدة من قواعد وحدة التحكم (مثل مستمع خارج الحلقة المحلية بلا وكيل موثوق) يوقفه بالرمز `CONSOLE001`.

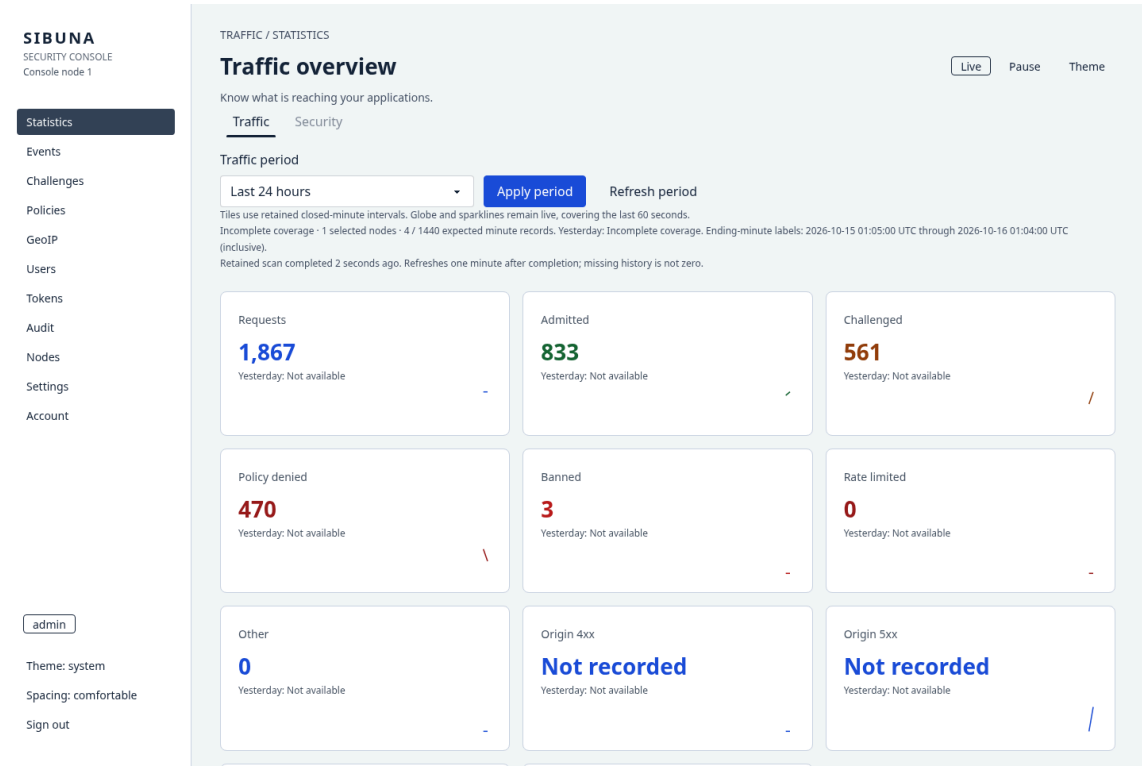
جولة في وحدة التحكم

تأتي لقطات الشاشة أدناه من بناء إصدار على عقدة واحدة تستمع على عنوان الحلقة المحلية مع تخزين SQLite وسياسة من أربع قواعد `ALLOW` فوق إجراء افتراضي هو التحدي، بعد نحو أربع دقائق من حركة مصطنعة: عناوين من نطاقات التوثيق، ومحاولات استكشافية لحقن SQL وللبرمجة النصية عبر المواقع، وثلاث إصابات للمصيدة، وسبعة تحديات حُلّت في Chromium بلا واجهة. لذلك فكل رقم هنا توضيحي. التقطناها بالأمر `cargo xtask browser console --viewport-only`، الذي يتجول في وحدة التحكم داخل Chromium بأبعاد

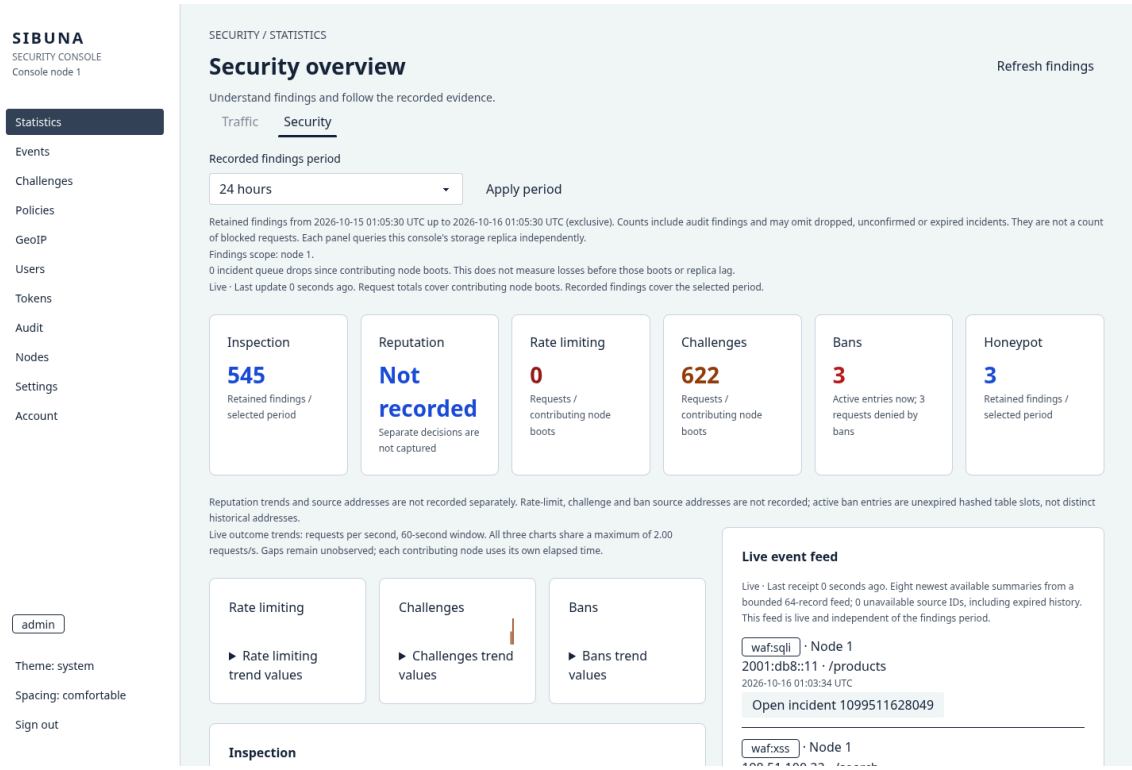
1440 في 960 بكسل ويُقيّم كل صفحة. تجيب كل صفحة عن سؤال واحد يذكره عنوانها، وتضع المنتج والعقدة الخادمة والصفحة وطريق العودة في الموضع نفسه.



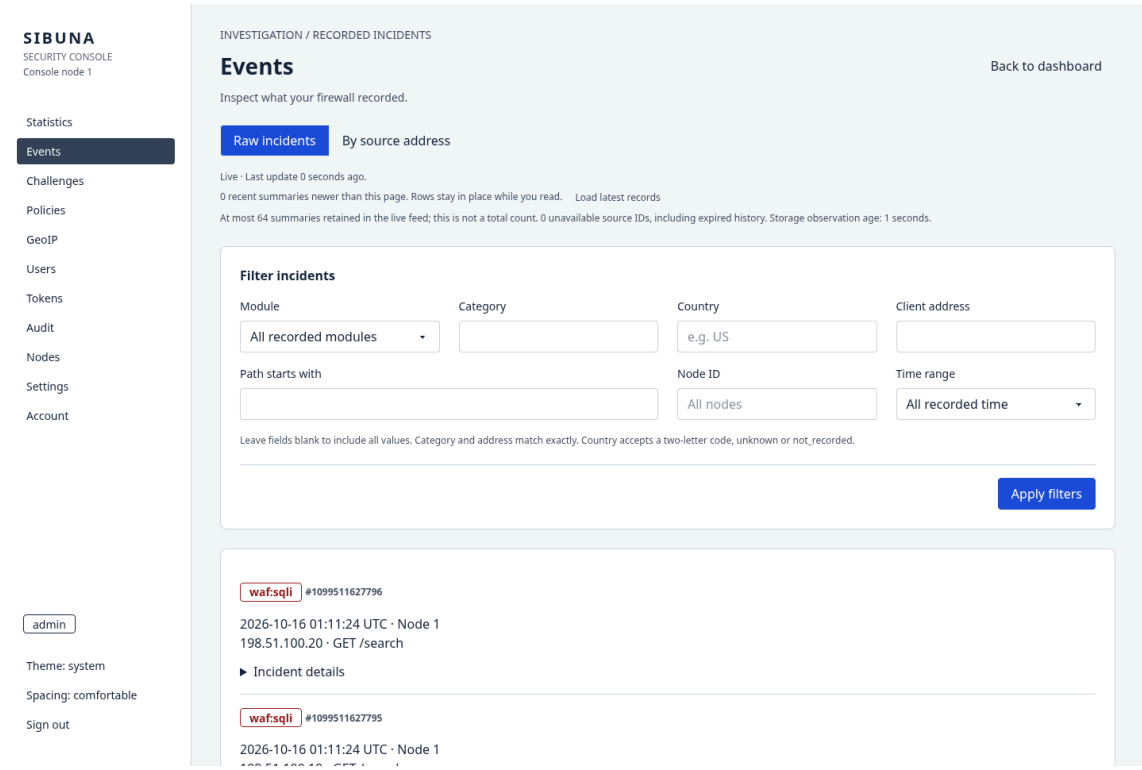
الشكل 13 تسجيل الدخول. لا تحمّل الواجهة العامة هندسة الكرة الأرضية ولا بيانات GeoIP ولا القياس عن بعد؛ ويستبدل النموذج الثاني رمز شاشة جدارية بجلسة إحصاءات للقراءة فقط.



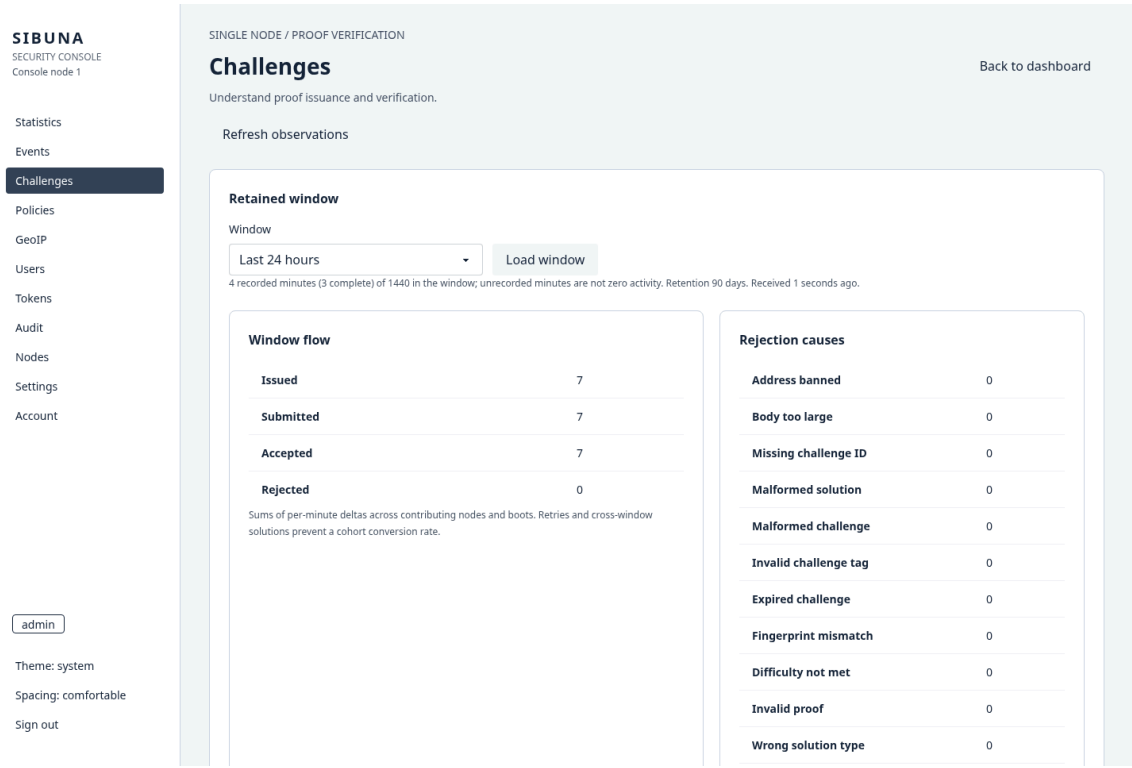
الشكل 14 Statistics، وTraffic، خلال آخر 24 ساعة. تذكر الملاحظة تحت ضابط الفترة التغطية: لا يوجد إلا أربعة من 1,440 سجل دقيقة متوقعًا، لذلك تظهر كل قيمة "Yesterday" بالعبارة "Not available". تعرض البطاقات أعداد الدقائق المغلقة المحتفظ بها مع منحنى صغير حي من 60 ثانية؛ وألوان القرار (المقبول أخضر، والخاضع للتحدي كهرماني، والمرفوض أحمر) واحدة في كل لوحة. وتظهر قيمتا 4xx و5xx للخادم الأصلي بالعبارة "Not recorded" لأي فترة محتفظ بها، لأن سجلات الدقائق لا تحفظ وضع الوكيل الذي يمنح عدد الخادم الأصلي معناه؛ وتعرضهما المجاميع الحية منذ الإقلاع. تحت البطاقات الظاهرة بطاقتا مستوى، هما مدخلات الحظر النشطة والعقد السليمة، ثم الخط الزمني، ولوحات عينات ترتب مسارات الطلبات والمضيفين المحيلين، وتعد أنظمة تشغيل العملاء ومتصفحاتهم وحالات الاستجابة.



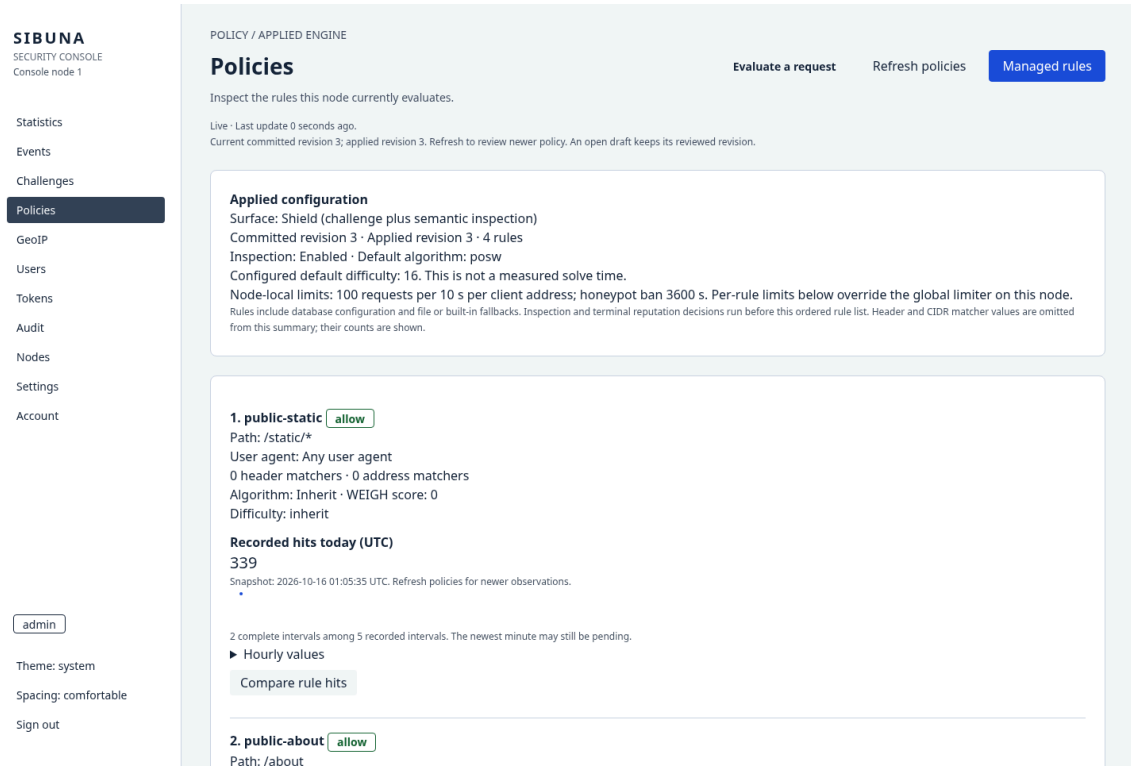
الشكل 15 Statistics، وSecurity. بطاقة لكل وحدة خلال الفترة المختارة، واتجاهات حية للنتائج، وموجز الأحداث الحي بأحدث نتائجه (هنا `waf:sqli` و`waf:xss`). والقيم غير المسجلة تصرّح بذلك بدل أن تعرض صفرًا: تظهر بطاقة Reputation بالعبارة "Not recorded". وفي أسفل الصفحة النتائج على ستين دلوًا متساويًا، وفئات الهجمات، والمسارات المهاجَمة.



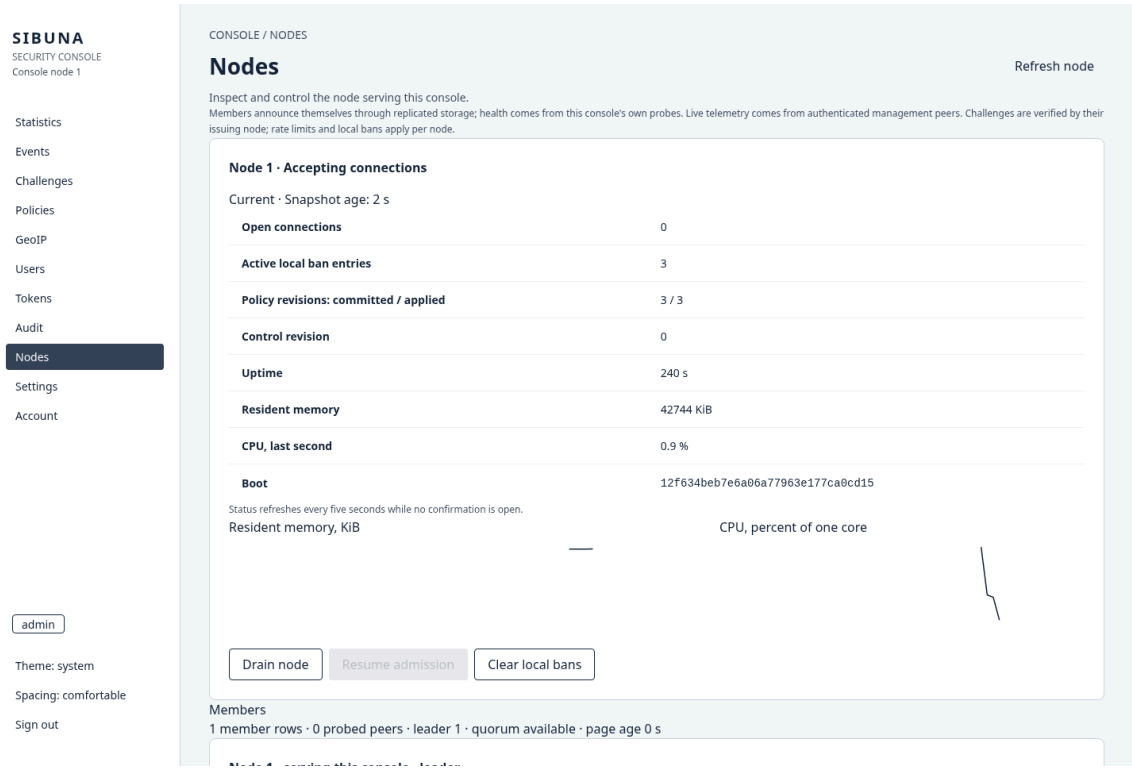
الشكل 16 Events. نموذج التصفية فوق الحوادث المسجلة، والأحدث أولًا. يُفتح كل حادث في مكانه مع أدلته: الاستجابة المحلية المختارة، وأطوال البايتات، ومرشح الحملة، ومدخلات "not recorded" صريحة للقاعدة المطابقة وحدود الدرجة وبصمة JA4. مع `--console-capture-heads` يُفتح رأس الطلب المنقح عند الطلب (ومعه رأس استجابة الخادم الأصلي لحالات القبول المدققة عبر الوكيل العكسي)، معروضًا بترميز UTF-8 أو Latin-1، مع ذكر حالة الاستجابة (captured، أو local، أو forward-auth unobserved، أو origin unavailable) وأمر للنسخ بصيغة cURL؛ ولا تُحفظ إلا قيم الترويسات المدرجة، ويضيف `--console-capture-header <name>` إلى تلك القائمة. وإلا فتظهر الرؤوس غير مسجلة. يفتح إجراءا الرفض والسماح نموذج مجموعات IP وقد أُدرج العنوان في مسودة.



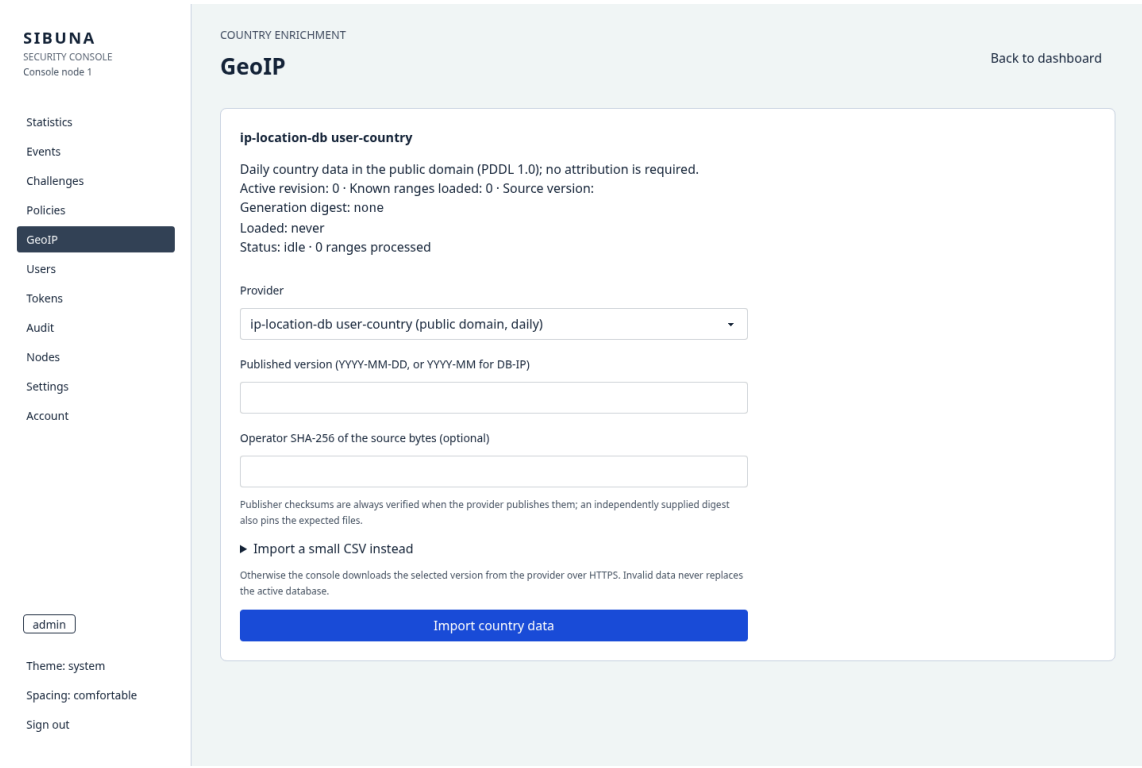
الشكل 17 Challenges. تجمع نافذة محتفظ بها (ساعة، أو يوم، أو سبعة أيام) سجلات الدقائق الدائمة، وتذكر عدد الدقائق المسجلة والمكتملة؛ هنا صدرت سبعة تحديات وقُدمت وقُبلت، وكل أسباب الرفض صفر. وفي الأسفل المجاميع الحية منذ هذا الإقلاع، والمعاملات المعدة والمعاملات الصادرة مؤخرًا، والمدرج التكراري لأزمنة الحل المقبولة مقسمًا حسب الخوارزمية وفئة المعاملات، وانتقالات الصعوبة التكيفية، وسجلات لكل عنوان مع اختصارات للرفض والسماح، محتفظًا بها سبعة أيام.



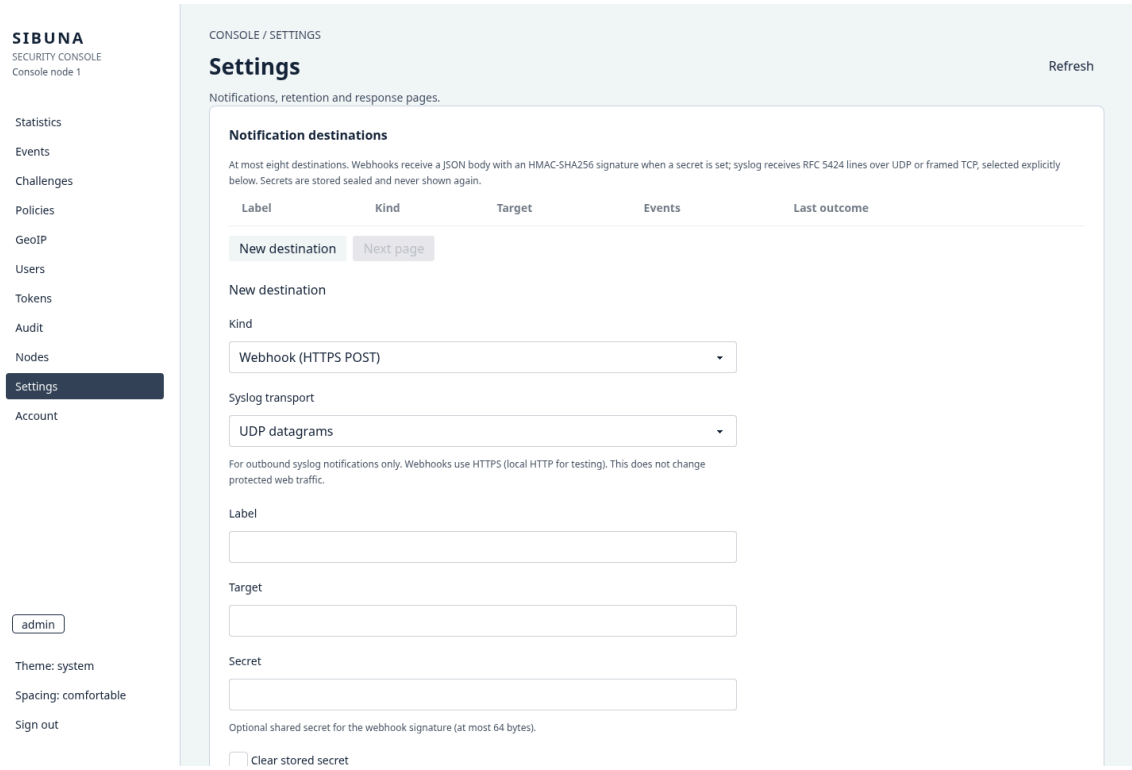
الشكل 18 Policies. يسمي المحرك المطبق وضعه ومراجعاته وخوارزميته الافتراضية وصعوبته وحدوده المحلية للعقدة، ثم يسرد القواعد بترتيب التقييم مع إصاباتها اليوم وتغطية تلك الأعداد. وفي الأسفل مصفوفة أوضاع الفحص، ومختبر الطلبات، ومجموعات IP. يفتح *Managed rules* المحرر؛ وحفظ تعديل يعرض أولًا مراجعة للحقول المتغيرة، والتأكيد يتحقق من المرشح كله وينشئ مراجعة جديدة.



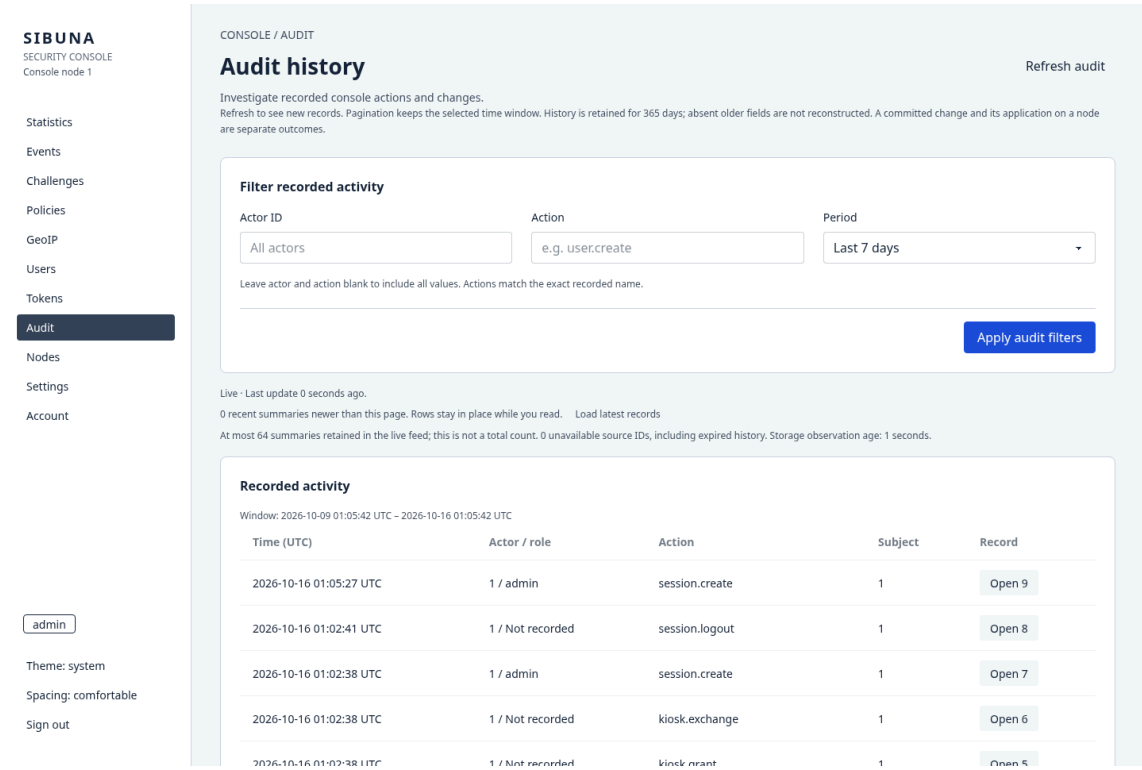
الشكل 19 Nodes. حالة العقدة الخادمة وأوامرها، مع الذاكرة المقيمة وCPU للثانية الأخيرة ومنحنياتهما الصغيرة من ستين ثانية؛ وتحتها كل عضو معلن مع مراجعة السياسة المطبقة لديه مقابل المراجعة المثبَّتة، وعنوان إدارته، ونتيجة مسباره (SID 0013، SID 0007).



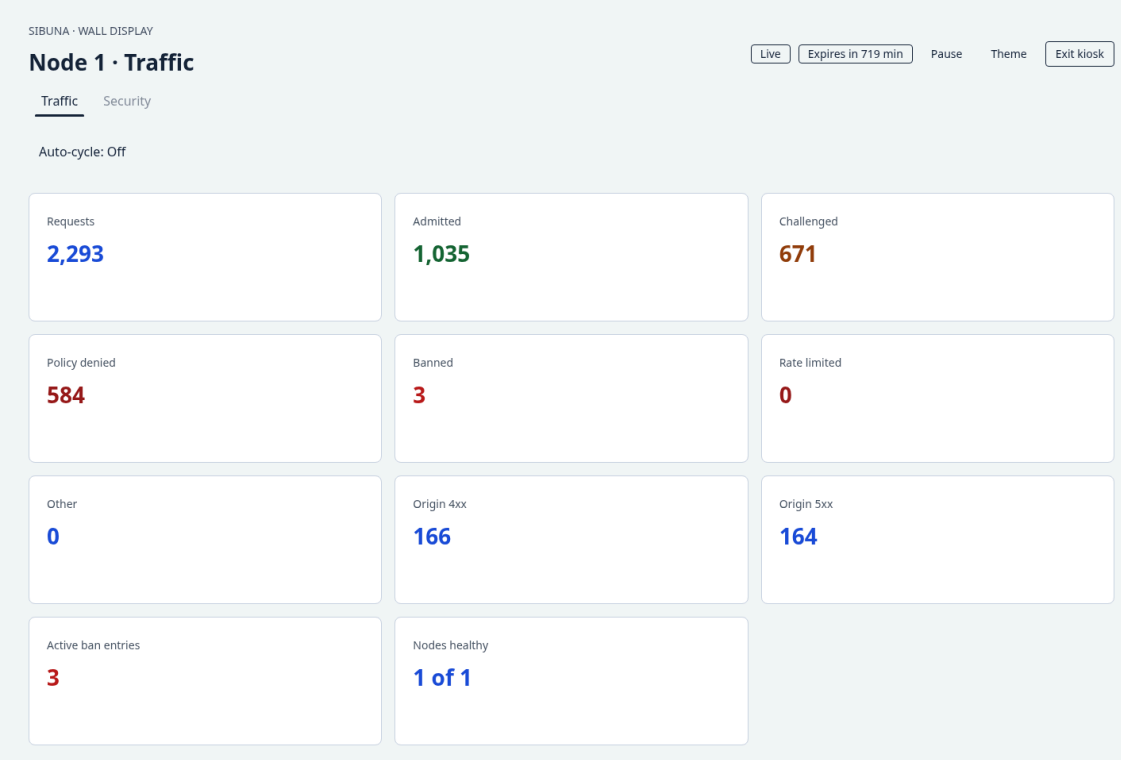
الشكل 20. GeoIP المزود، والترخيص، وملخص الجيل النشط، ووقت التحميل، ونموذج الاستيراد؛ هنا لم يُحمّل أي جيل بعد. والاستيراد الفاشل لا يحل أبدًا محل الجيل النشط.



الشكل 21 Settings، من دون وجهة إشعارات بعد، مع نموذج لوجهة جديدة. وفي أسفل الصفحة عتبات ارتفاع الرفض، والاحتفاظ مع إقرار صريح لكل نافذة، وقوالب صفحات الاستجابة، وAbout.



الشكل 22 Audit. سجل تاريخي لا يقبل إلا الإضافة يضم الفاعل ودوره، والإجراء، والموضوع، وعرضًا تفصيليًا بملخصات منقحة لما قبل التغيير وما بعده؛ وتُسجل محاولات تسجيل الدخول المرفوضة أيضًا. يسجل كل تغيير إداري وكل صف مصادقة عنوان العميل الذي قدم بيانات الاعتماد؛ والصفوف التي يكتبها النظام بنفسه تعرضه على أنه غير مسجل. يقدم سجل السياسة *Revert this change*، الذي يستعيد الوثيقة المسجلة قبل تلك المراجعة بوصفها مراجعة جديدة مدققة بعد تأكيد، ويُرفض إذا تقدمت مجموعة القواعد بعدها.



الشكل 23 شاشة جدارية. تعرض جلسة kiosk عرضي Traffic وSecurity فقط، بلا تنقل، وتذكر موعد انتهائها.

يعلن الخيار الاختياري `--console-location <latitude,longitude>` موقع هذه العقدة، مثل `1.3521,103.8198` لنشر في سنغافورة. تتمركز الكرة الأرضية هناك في البداية؛ ويعيد Center Sibuna إلى تلك النقطة بعد التدوير. يتبع نشاط البلدان أقواس دوائر عظمى متحركة نحو العقدة، مع قص نصف الكرة الخلفي وقص خط التاريخ في الخريطة المسطحة. مواقع البلدان مراكز تمثيلية، وتمثل الأسهم النافذة المرصودة من ستين ثانية لا اتصالات منفردة. ويبقى موقع الخادم غير المعد مجهولًا صراحة. تظل العلامة المعدة ظاهرة أثناء انقطاع القياس عن بعد؛ ولا تُحرَّك بيانات الحركة المتقادمة.

تبني معاينات السياسة مرشحًا خاصًا يشمل قواعد الملف المعدة. يقارن الحفظ المراجعة المتوقعة ويتطلب تأكيد مقارنة للحقول. ويقارن الاسترجاع التاريخي بالقاعدة الحالية وينشئ مراجعة جديدة. المراجعات المثبَّتة والمراجعات المطبقة محليًا أمران مختلفان. تحتفظ تفاصيل التدقيق بتغييرات محدودة في القرار وبالدور الفعلي للفاعل. تُنقح قيم المطابِقات، وتبقى السجلات القديمة التي ينقصها السياق غائبة صراحة. تتطلب أوامر drain وresume وclear-local-bans معاينة للأمر. والأمر الذي يصل بينما يجري أمر آخر على العقدة يُرفض بالرمزين `503` و`CONSOLENODE` بدل أن يوضع في طابور. افحص الإيصال الدائم بعد ضياع استجابة وقبل إعادة المحاولة؛ فالنية المثبَّتة وحدها لا تثبت أن الأثر في وقت التشغيل قد اكتمل.

للتشغيل عن بعد، استخدم وكيل HTTPS واضبط `--console-origin`، و`--console-behind-proxy`، ونطاقات CIDR صريحة عبر `--console-trusted-proxy`، و`--console-key-file`. يحتوي ملف المفتاح على 64 محرفًا سداسيًا عشريًا، وأذوناته للمالك وحده، ويحمي أسرار العامل الثاني المخزنة؛ احتفظ به عبر إعادة التشغيل منفصلًا

عن بذرة التحدي. أصل HTTPS المعد وقائمة السماح للوكلاء إلزاميان للوصول من خارج الحلقة المحلية. ولا ينهي Sibuna أي TLS وارد على مستمع وحدة التحكم كذلك؛ إذ لا تستخدم وحدة التحكم TLS إلا لاتصالاتها الصادرة، عبر rustls (SID 0012).

تصل واجهة CLI الأصلية إلى وحدة التحكم الجارية عبر عقود التفويض والتخزين نفسها. وتقرأ بيانات الاعتماد من ملفات خاصة، لا من قيم سطر الأوامر:

```
sibuna console geoip update --version 2026-09-09 \
    --origin http://127.0.0.1:19446 --username admin \
    --password-file ./admin-password
```

استخدم ملف كلمة مرور للمالك وحده، وأكمل إعداد كلمة المرور أولًا، وأضف `--factor-file` لرمز مصادق أو رمز استرداد عند الحاجة؛ ويحل `--token-file` محل اسم المستخدم وكلمة المرور في عمليات الحسابات وGeoIP. يتحدث العميل HTTP/1.1، ويتطلب HTTPS إلا مع عنوان حلقة محلية صريح، ولا يتبع إعادات التوجيه أبدًا، ويحد كل طلب بـ 20 ثانية وكل رد بـ 16 KiB. يقرأ `geoip status` الجيل النشط. تنزّل التحديثات الملفات المنشورة للمزود المختار عبر HTTPS (مجموعة البيانات `user-country` ذات الملكية العامة افتراضيًا، أو DB-IP Lite مع `--provider dbip --version YYYY-MM`)، وتتحقق من المجاميع الاختبارية للناشر (أو من المجموع المعطى مع `--checksum`)، وتتحقق من النطاقات المحدودة عبر `sibuna-geoip`، وتحفظ الجيل الجديد وتفعّله محليًا. تحافظ التحديثات الفاشلة على الجيل السابق. تُوقف مهلة CLI (`--timeout`، والافتراضي 1200 ثانية) الانتظار؛ لكنها لا تلغي عمل التخزين المُقدَّم. يتطلب DB-IP Lite إسنادًا وفق CC BY 4.0، لا يظهر إلا ما دامت بياناته نشطة. البناء مع `SIBUNA_GEOIP_DATA=<snapshot>` يضمّن لقطة متحققًا منها (تكتبها الأداة `geoip-snapshot`) حتى أول استيراد دائم؛ والبناء الذي لا يضم وحدة التحكم يرفضها بالرمز `GEOIP001`. تبقى العناوين المجهولة وفقد العينات والبيانات المتقادمة ظاهرة؛ واستيراد بيانات البلدان لا ينشئ حركة ولا يُثري عينات انتهت مدتها.

يعلن أعضاء العنقود عن أنفسهم عبر جدول العضوية في قاعدة البيانات المشتركة (`console_nodes`) عند البدء، وكل دقيقة، وخلال ثانيتين من تغير مراجعة السياسة أو مراجعة التحكم المطبقة، ويظهرون في صفحة Nodes في كل وحدة تحكم. لا يوجد انتخاب قائد: تبلغ عقدة SQLite عن الدور `single` وعن نفسها قائدًا، وتبلغ عقدة PostgreSQL عن الدور `unknown` ومن دون قائد، لأن تحويل PostgreSQL عند الفشل غير مرئي لـ Sibuna (SID 0007). اضبط `--console-probe <node-id>=<http://ip:port>` للأقران الذين يجب أن تفحص وحدة التحكم هذه صحة مستمعات مستوى البيانات لديهم، و`--console-advertise <origin>` للرابط الذي تعرضه وحدات التحكم الأخرى. يستخدم القياس الحي المباشر عن بعد `--console-peer <node-id>=<https://origin>` (قابل للتكرار، ثمانية على الأكثر) وملفًا مخصصًا عبر `--console-peer-key-file`. يتطلب الأقران `--console-behind-proxy`. يجب أن تسرد كل نقطة نهاية الأخرى، وأن تستخدم كلتاهما وكيل دخول HTTPS موثوقًا. مفتاح الأقران ملف للمالك وحده يحتوي على 64 محرفًا سداسيًا عشريًا، ويُجهز مستقلًا عن مفتاح وحدة التحكم وبذرة التحدي. يتواصل الأقران عبر WebSocket مصادق عليه فوق TLS، من خلال rustls مع المزود `ring` (SID 0012). يوفر `--console-peer-ca-file` اختياريًا مراسي ثقة PEM لبنية مفاتيح عامة (PKI) خاصة بالإدارة؛ وإلا فيستخدم العميل جذور النظام. يسري التحقق من اسم المضيف دائمًا ولا يمكن تعطيله: يجب أن تسمي شهادة القرين مضيف كل أصل إدارة، سواء كان اسم DNS أو عنوان IP صريحًا. والشهادة المرفوضة تعلّم القرين بالحالة `rejected` وتكتب

صف تدقيق واحدًا `peer.rejected`. يتطلب تدوير المفتاح وتغيير ملف CA إعادة تشغيل منسقة؛ أما تجديد شهادة القرين فلا يتطلبها. تبلغ واجهة Nodes API واشتراكها عن عمر الاستلام، وانحراف الساعة، وتغيرات الإقلاع، وفقد العينات. تبقى الأرصاد الغائبة غير متاحة، والقيم المنقطعة متقادمة؛ ولا تصبح الإحصاءات المستلمة أبدًا مساهمة خاصة بعقدة أخرى. يختار *Live traffic scope* في لوحة المعلومات عقدة واحدة أو يجمع العقد المعدة. ويحدد *Node coverage and locations* المصادر الغائبة والمتقادمة والمنحرفة الساعة، وأوقات الرصد، والوجهات المعدة. تعرض ترتيبات البلدان عدم اليقين الناتج عن صفوف المصادر المحذوفة. تحتفظ الأسهم بعقدتها المستقبلة، ولا تُختلق مواقع العقد الغائبة. يبقى المعدل المجمع غير مرصود ما دام أي مصدر مساهم يفتقد فترة متتالية. اختر عقدة واحدة لتفحص ثوانيها المحتفظ بها وترتيبات مسارات الدقيقة الحالية. تستخدم الاستعلامات البعيدة اتصال الأقران المصادق عليه؛ ويجب أن تتسع إجابة الاستعلام لرسالة WebSocket واحدة من 4 KiB، لذلك تنصّف وحدة التحكم الخادمة صفوف الصفحة التي لا تتسع. يبقى الأقران الغائبون غير متاحين، ولا تعبر مؤشرات التاريخ إعادة تشغيل عقدة. يحتفظ تاريخ الدقائق باختياره الخاص للعقد وبصفوف لكل عقدة. يحدد التنقل وترويسة الهاتف عقدة وحدة التحكم الخادمة مستقلًا عن مصدر الحركة المختار. وتبقى الأوامر المحلية خاصة بكل عقدة.

تُفتح بطاقات الحركة على آخر 24 ساعة من سجلات الدقائق المغلقة المحتفظ بها للعقد المختارة. ويقدم ضابط الفترة أيضًا ساعة، أو سبعة أيام أو تسعين يومًا، والمجاميع الحية منذ الإقلاع. تتطلب الفروق عن الأمس تغطية مطابقة كاملة من كل عقدة مختارة. وتظل الكرة الأرضية والمنحنيات الصغيرة تصف نافذتها الحية المنفصلة من 60 ثانية. يتجدد المسح المحتفظ به بعد دقيقة من اكتماله، مع إبقاء قيمه السابقة وعمرها ظاهرة حتى الاستبدال؛ وتغيير المصدر أو الفترة يُسقط النطاق القديم. التاريخ الغائب لا يصبح صفرًا أبدًا.

يعرض *Compare retained traffic* نافذتين مغلقتين من الدقائق جنبًا إلى جنب: النافذة نفسها أمس، أو الفترة السابقة، أو عقدة ثانية. تثبّت ضوابط المدة وإزاحة النهاية نطاقي UTC كليهما. يقرأ كل إجراء ست عشرة صفحة مضغوطة على الأكثر لكل جانب، مع 96 سجلًا مخزنًا في كل صفحة. يغطي هذا نافذة 24 ساعة بلا إعادات تشغيل متداخلة في دفعة واحدة؛ ويحتفظ Continue بالحدود نفسها للمسوح الأطول. تبقى التغطية الكاملة والتغطية الناقصة والتاريخ الغائب صريحة. تتطلب نسب المعدلات فترات كاملة غير متداخلة في نوافذ UTC متساوية، مطبّعة بالمللي ثانية المرصودة؛ والمرجع الصفري مع حركة جديدة يعرض "New". تبقى فترة Traffic الأساسية مستقلة. لا يسجل تنسيق الدقائق وضع الوكيل التاريخي، لذلك تبقى مقارنات استجابات الخادم الأصلي غير متاحة بدل تفسير أصفار forward-auth على أنها استجابات مرصودة.

يختار *Compare retained path rankings* نافذتين مغلقتين أو عقدتين. تشمل العقدة 0 كل العقد المحتفظ بها، بما فيها الأعضاء المتقاعدون. يقرأ الإجراء ستة عشر أرشيفًا ثابتًا على الأكثر، ويتحقق من هوياتها ومجاميعها الاختبارية، ويدمج كل عداد قبل اختيار صفوف العرض. تُختم الأرشيفات بعد نافذة العينات المتأخرة من 60 ثانية، لذلك قد تكون أحدث دقيقة مغلقة ما زالت معلقة. يحافظ Continue على النوافذ والمؤشر الأصليين. تصف الحدود بادئات مسارات مأخوذة بالعينات، لا مجاميع طلبات دقيقة. تعرض الصفحة المسوح الجزئية، والاقتطاع، وحدود الاحتفاظ، وفقد الطابور المبلغ عنه؛ والأرشيفات الغائبة لا تثبت حركة صفرية ولا تغطية كاملة.

تُفتح Security على آخر 24 ساعة، وتستخدم ستين دلوًا زمنيًا متساويًا على الفترة المختارة. تجري تجميعاتها على خيط التخزين تحت حد للعمل. في SQLite يكون الحد `--console-query-steps` خطوة من خطوات الآلة الافتراضية (من 100,000 إلى 50,000,000، والافتراضي أربعة ملايين). اضبطه مع `console--`؛ وتُرفض القيم

غير الصالحة أو المكررة. وفي PostgreSQL يكون الحد مهلة تعليمة قدرها 2,000 ms تُضبط في معاملة القراءة (SID 0013)؛ وتغيير `--console-query-steps` لا يغيرها. والاستعلامات المستنفدة تطلب فترة أضيق. تحد أعداد الخطوات عمل الاستعلام، لا الزمن المنقضي. تتشارك المخططات مقياسًا واحدًا؛ وسّع قيم Trend لترى بدايات فترات UTC والأعداد المجمعة الدقيقة. هذه نتائج محتفظ بها، لذلك لا يجوز تفسير غياب تغطية الحوادث على أنه غياب للهجمات.

تستخدم الشاشات الجدارية جلسات kiosk. تحت Account، يسمي المشغل الشاشة ويختار Create display code. تلصق الشاشة الرمز ذا الاستخدام الواحد في نموذج تسجيل الدخول خلال عشر دقائق؛ وجلستها للقراءة فقط، ومقصورة على الإحصاءات، وتنتهي خلال اثنتي عشرة ساعة. تمحو صفحة الحساب الرمز المعروض عند التنقل أو عند إخفائها. لا يلغي الإخفاء منحة غير مستخدمة؛ وتبقى مهلة التبادل سارية. لا مكان للرموز في عناوين URL أبدًا. تتشارك Traffic وSecurity الشاشة: تعرض Security اتجاهات الوحدات المجمعة ونتائج الطلبات بلا عناوين الحوادث ولا أدلة الحمولة. التناوب الآلي اختياري، ومعطل في البداية، ويُعلّق مع تقليل الحركة، أو البيانات المتقادمة، أو Pause.

وجهات الإشعارات (webhooks موقعة وsyslog لارتفاعات الرفض، والحظر، والأعضاء الذين لا يمكن الوصول إليهم، وتغيرات القائد) موجودة تحت Settings؛ وتحتاج أسرار webhook إلى `--console-key-file`. يبقى نوع تغير القائد معرفًا لكي تحتفظ الوجهات المخزنة بمعناها، لكن لا تطلقه أي عقدة، لأنه لا يوجد انتخاب قائد. لكل وجهة فترة تهدئة خاصة بها وميزانية إعادة محاولة من ثلاث محاولات. تعرض سجلات تدقيق التسليم النتائج؛ ويُحد تاريخ الطابور المكتمل بسبعة أيام و4,096 حدثًا. تتضمن webhooks مفتاحًا ثابتًا `Idempotency-Key` لكي يستطيع المستقبلون كبت الآثار المكررة بعد اكتمال غير مؤكد في الشبكة. تُرسل webhooks عبر HTTPS، أو عبر HTTP العادي إلى عنوان حلقة محلية صريح فقط، ويُقرأ جسم الرد حتى 64 KiB. لا يسري اختيار UDP/TCP في syslog إلا على الإشعارات الصادرة، مستقلًا عن حركة الويب المحمية. تسجل الاختبارات اليدوية النية والاكتمال في Audit وتجدد نتيجة الوجهة؛ ويُعرض اكتمال تدقيق غير مؤكد صراحة قبل أن يعيد المشغل المحاولة.

تتحكم Settings أيضًا في الاحتفاظ: من يوم واحد إلى 90 يومًا لتاريخ الدقائق، ومن يوم إلى سبعة للترتيبات، ومن يوم إلى 30 للحوادث، ومن يوم إلى 365 للتدقيق. القيم العليا هي الافتراضية؛ وتحتفظ الترتيبات بحصتها البالغة 512 MiB. يتطلب حفظ قيمة احتفاظ تأكيدًا لأن التنظيف يزيل السجلات الأقدم نهائيًا على دفعات محدودة. وزيادة القيمة لاحقًا لا تستعيد التاريخ المحذوف. تُرفض التعديلات المتقادمة، وتظهر التغييرات الناجحة في Audit.

صفحات الاستجابة (التحدي، والرفض، وتجاوز حد المعدل، والحظر، والتحميل الزائد) قابلة للتحرير تحت Settings بوصفها HTML محدودًا بعناصر نائبة ثابتة؛ وتُعاين المسودات في علامة تبويب معزولة، وتُقدم الصفحات المحفوظة ابتداءً من لقطة السياسة التالية.

مسارات عمل السياسة في صفحة Policies: أعد ترتيب القواعد المُدارة، وأعد تشغيل مسودة على نتائج الفحص المحتفظ بها، وأدر مجموعات IP وحظر البلدان المثبت على جيل GeoIP النشط، وصدّر المجموعة المُدارة أو استوردها ذريًا (وكذلك عبر `sibuna console policies export` و `sibuna console policies import --file <set.json>`). الاستيراد طلب واحد يحمل من 1 إلى 128 وثيقة، لا تتجاوز كل منها 4 KiB ولا يتجاوز مجموعها 256 KiB؛ وتسمي ملاحظة الصفحة الاستيراد من سطر الأوامر للمجموعات التي تتجاوز نحو 96 KiB. تعرض القواعد المطبقة الإصابات المسجلة اليوم، ومنحنيات صغيرة لكل ساعة،

وجدولًا سهل الوصول. الإصابة تقييم ناجح لمطابِق تصريحي: تُعد قواعد WEIGH المطابقة وأول مطابقة نهائية؛ وقد يمنع قرار سابق من الفحص أو السمعة التقييم. لا تزيد الاختبارات الخاصة هذه العدادات. يثبّت *Compare rule hits* فترتين مغلقتين لعقدة مسجلة واحدة. ويقدم تاريخ المراجعات *Compare hits around this edit*، مستثنيًا دقيقة التعديل ومستخدمًا فترتين متاحتين متساويتين حتى المدة المختارة. تُبقي مرشحات المراجعة المطبقة الاختيارية الأجيال غير ذات الصلة خارج المقارنة. يترك بدء التشغيل والتحويل والكتابات الغائبة والاحتفاظ فجوات تغطية ظاهرة؛ وتتطلب النسب تغطية كاملة في الجانبين. تتبع أرصاد القواعد والملخصات الساعية واليومية إعداد الاحتفاظ بالدقائق. والتغير المرصود حول تعديل مقارنةٌ، لا دليل على أن التعديل سبب تغير الحركة.

استيراد جيل GeoIP أحدث لا يجدد تلقائيًا صفوف السمعة الموجودة المشتقة من البلدان. عاين إجراء البلد لتقارن البادئات المضافة والمحتفظ بها والمزالة؛ وتصفح الفرق المراجع صفحة صفحة قبل تطبيقه. يزيل الاستبدال الصفوف المتقادمة التي يملكها ذلك البلد، ويحافظ على البادئات المُدارة مستقلًا. يتطلب تغير الجيل أو مراجعة السياسة معاينة جديدة، وتُرفض التعديلات المستقلة المتداخلة. تحتوي صفحة Events على نتائج WAF المحتفظ بها وحوادث المصيدة؛ وليست سجل وصول كاملًا. عند تحميل GeoIP، يسجل خيط التخزين بلد كل حادث وجيله في معاملة دفعة الحوادث. هذا إسناد عند الحفظ، لا موقع مُعاد بناؤه وقت الطلب. تترك عمليات الاستيراد اللاحقة الربط المسجل دون تغيير. يقبل مرشح البلد رمزًا من حرفين كبيرين، أو `unknown` لعنوان غائب عن الجيل المحمّل، أو `not_recorded` لحادث بلا بيانات ربط. تبلغ مجموعات المصادر عن البلدان المختلطة أو التغطية صراحة. يحتفظ إجراء *View events* في الكرة الأرضية بالعقدة المختارة ويفتح حوادث البلد المحتفظ بها لآخر ساعة. يجمع عرض *Security* في صفحة Statistics معدلات حد المعدل والتحديات والحظر الحية مع نتائج الفحص والمصيدة المحتفظ بها على مدى ساعة، أو يوم، أو سبعة أيام، أو ثلاثين يومًا. تفتح روابط الفئة والمصدر والمسار مسار عمل الحوادث المعتاد بالعقدة نفسها وحد زمني ثابت. يبدأ تطبيق مرشحات الحوادث فترة جديدة. تشمل النتائج سجلات التدقيق، وهي مختلفة عن مجاميع الطلبات المحظورة؛ ويُعرض غياب إسناد السمعة وإصابات القواعد على أنه غير مسجل. تبقي مرشحات الأحداث والتدقيق التسميات مع ضوابطها، وتحاذي إجراء Apply منفصلًا، وتكيّف أعمدتها مع عرض المحتوى المتاح. يبقى WebSocket مصادق عليه واحد عبر التنقل، ويحمل الإحصاءات، وملخصات الحوادث، وحالة العقد، ومراجعات السياسة، والتحديات، وملخصات التدقيق. تنتظر سجلات الحوادث والتدقيق الجديدة خلف *Load latest records* لكي يبقى الجدول في مكانه أثناء قراءته. تعرض تحديثات السياسة المراجعات المثبَّتة والمطبقة الحالية من دون استبدال مسودة مفتوحة. تتحدث عدادات التدفق حيًا؛ أما قسم توقيت تحدٍّ غير افتراضي مختار فجدّده صراحة. تبقى الاستعلامات التاريخية وقراءات التفاصيل والتغييرات طلبات HTTP.

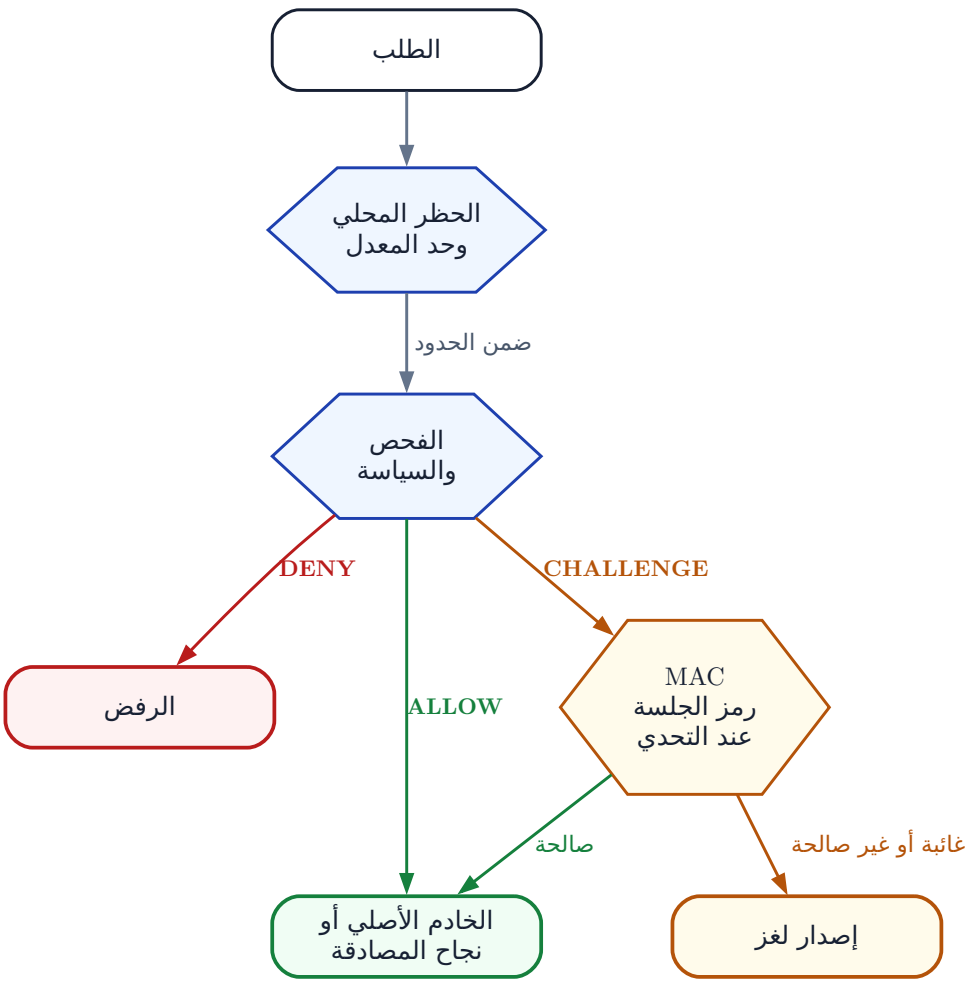
يمكن وضع إشارات مرجعية على أجزاء الصفحات؛ ويعيد Back وForward فتح الصفحات التي تتطلب المصادقة. ويتذكر الشريط الجانبي أيضًا اختيارات السمة والتباعد في هذا المتصفح، والسمة الافتراضية هي System. حالة SID 0007 هي Committed. تجتاز وحدة التحكم اختباراتها المرجعية الذهبية ومسارات عملها في المتصفح داخل Chromium بلا واجهة (`cargo xtask browser console`، المعيار 5 من SID 0010). ومجموعة اختباراتها من طرف إلى طرف هي `cmd/sibuna/tests/console_*.rs`، وتجري سيناريوهات عنقود PostgreSQL فيها عند ضبط `SIBUNA_TEST_DATABASE_URL`. يسجل SID 0010 أن أعباء العمل الـ 16 للمنتج تبقى في حدود نحو 1% من البرنامج الخدمي المبني بدون وحدة التحكم، حين تكون مضمّنة في البناء، سواء أكانت معطلة، أم للتخزين فقط، أم مفعلة وخاملة، وهذا ضمن هامش الضجيج. أما معيار الاجتياز الرسمي للعزل في SID 0007، مع ثماني لوحات

معلومات نشطة تحت الحمل، فلم يُقس بعد (الجزء الثامن). لتطبيق WebAssembly الكامل سقف قدره 768 KiB غير مضغوط، يفرضه سكربت بناء البرنامج الخدمي بإفشال البناء، وعتبة مراجعة قدرها 640 KiB. في الإصدار beta.3-1.0.0 يبلغ حجم الوحدة 724,330 بايت (272,568 بعد gzip)، وهو فوق عتبة المراجعة، ويسجله SID 0007 للمراجعة. تترك حدود المشروع هذه متسعًا لمسارات عمل وحدة التحكم؛ لكنها لا تغني عن قياس التحميل والاستجابة في المتصفح. وتنمو الذاكرة الخطية للوحدة عند الطلب حتى حد أقصى معلن قدره 16 MiB.

طوبولوجيات النشر

الأهداف انشر الوكيل العكسي المستقل وخدمة التحقق forward auth خلف Nginx أو Caddy.

الوكيل العكسي



الشكل 24 توجيه الطلبات في وضع Shield

```
sibuna --port 80 --upstream-host 127.0.0.1 --upstream-port 3000 \
       --secret-file /etc/sibuna/secret --difficulty 16 --algorithm posw
```

تصل الطلبات المقبولة إلى الخادم الأصلي مع الترويسات X-Forwarded-For وX-Forwarded-Proto وX-Real-IP وX-Sibuna-Status وX-Sibuna-Rule. تُحذف ترويسات hop-by-hop، والحقول التي يكتبها Sibuna بنفسه، وحقول التمرير البديلة (Forwarded وX-Forwarded-Host وX-Forwarded-Port)، حتى عندما يكتب العميل أسماءها بشرطات سفلية (X_Forwarded_For). X-Real-IP هو العميل الذي حُكم على الطلب بناءً عليه. و X-Forwarded-For هو نظير المقبس وحده، وليس أبدًا مدخلًا كتبه العميل؛ وعندما تكون الهوية الممررة موثوقة، يكون هو السلسلة المستلمة متبوعة بنظير المقبس. لذلك يجب أن تقرأ الخوادم الأصلية العميل من X-Real-IP أو من يمين X-Forwarded-For، وليس أبدًا من مدخله الأيسر. المسار الذي يستلمه الخادم الأصلي هو الصيغة المعيارية التي حكمت عليها السياسة (SID 0009)؛ والمسار الذي لا صيغة معيارية له يُجاب بالرمز 400.

تمر ترقيات WebSocket في HTTP/1.1 بفحوص القبول والسياسة قبل قبول مصافحة الخادم الأصلي. ويبقى التفاوض على البروتوكولات الفرعية والامتدادات بين الطرفين. يرحّل خيطان البايتات، خيط لكل اتجاه، ولكل منهما مخزن مؤقت ثابت واحد، بما في ذلك البايتات المجلوبة مسبقًا والإغلاقات الجزئية؛ لذلك يشغل الاتصال المرقّى خيطين، وقد يبلغ عدد خيوط الاتصالات ضعف --max-connections عندما يُرقّى كل اتصال. تبقى المقابس المرقاة ضمن حصة الاتصالات المعتادة وسجل الإغلاق، لكنها تستخدم --websocket-idle-timeout (300 ثانية افتراضيًا) مستقلًا عن مهلة خمول HTTP. تجدد الحركة في أي من الاتجاهين ذلك الحد، لكن الطرف الذي لا يقبل أي بايت طوال الفترة كلها ينهي الترحيل، مهما كان الاتجاه الآخر مشغولًا. الإطارات بعد المصافحة ليست مدخلات لفحص WAF. يستخدم HTTPS/WSS وكيل دخول ينهي TLS أمام مستمع HTTP/1.1 الخاص بـ Sibuna؛ ولا ينهي Sibuna نفسه TLS المتصفح.

يستطيع وكيل الدخول التفاوض على HTTP/2 مع المتصفحات وتمرير HTTP/1.1 إلى Sibuna. أُرجئ دعم HTTP/2 الأصلي في Sibuna إلى تحديث لاحق؛ ولا يوفر هذا الترتيب دلالات HTTP/2 من طرف إلى طرف لتطبيقات مثل gRPC الأصلي.

تتدفق أجسام الطلبات ذات Content-Length عبر مخازن مؤقتة ثابتة، مع الحفاظ على البايتات وترويسات MIME. تحتفظ عمليات الرفع متعددة الأجزاء بالحدود الفاصلة، وأسماء الحقول المكررة، وأسماء الملفات، وأنواع الأجزاء. لا يفحص WAF إلا أول 8 KiB من الجسم على الأكثر: تبقى البيانات الوصفية متعددة الأجزاء والحقول غير الملفية مدخلات للفحص النصي، بينما تكون حمولات الملفات معتمة. والجزء الذي يكون filename فيه فارغًا حقل عادي. الجسم من نوع MIME ثنائي معروف على المستوى الأعلى (الصور عدا SVG، والصوت، والفيديو، وPDF، وZIP، وgzip، و7z، وprotobuf، وoctet-stream) لا يكون معتمًا إلا إذا بدت بادئته ثنائية أيضًا، أي إذا كان بايت واحد على الأقل من كل ستة عشر بايت تحكم غير الجدولة وتغذية السطر والجدولة الرأسية وتغذية الصفحة والعودة إلى بداية السطر؛ وإلا فيُفحص نصًا (SID 0004). تحتفظ JSON وXML وSVG والنماذج المرمزة بصيغة URL والأنواع المجهولة بالفحص النصي. وتعود البيانات الوصفية لـ MIME الملتبسة أو المعيبة إلى الفحص النصي. يُحد

تحليل الأجزاء المتعددة بـ 32 جزءًا و2 KiB من الترويسات لكل جزء ضمن البادئة نفسها، ويقبل حشو النقل الذي يجيزه RFC 2046 بعد الفاصل.

هذا توافق مع عمليات الرفع، وليس تحققًا من الملفات ولا فحصًا للبرمجيات الخبيثة. يجب أن يفرض الخادم الخلفي أنواع الوسائط المقبولة بدل الوثوق بتصريح Content-Type من العميل. الحقول الواقعة بعد بادئة الفحص، بما فيها الحقول الواقعة بعد ملف كبير مرفوع، لا تُفحص. ولا تُفك الحمولات المضغوطة لفحصها. تمر عمليات الرفع مع ذلك بفحوص القبول والمسار والاستعلام والترويسات، وتبقى خاضعة لمهل الاتصال. تُفك أجسام الطلبات المجزأة (chunked) داخل مخزن الاتصال المؤقت قبل الفحص (SID 0009). الجسم الذي ينتهي داخل المخزن يصل إلى الخادم الخلفي مع Content-Length. أما الأطول فيعيد Sibuna تجزئته، جزءًا لكل قراءة، فلا يرى الخادم الخلفي أبدًا أحجام أجزاء العميل ولا امتداداته ولا ذيوله. سطر الجزء الذي يحتوي CR أو LF منفردًا، أو مسافات بيضاء حول الحجم، أو امتدادًا معيبًا، أو أكثر من 4 KiB يُرفض بالرمز 400. وكذلك قسم الذيول الذي يتجاوز 16 KiB. ويتلقى `Transfer-Encoding` بترميز آخر الرمز 501. والخوادم الخلفية التي لا تستطيع تحليل الطلبات المجزأة تتلقى مع ذلك Content-Length للأجسام الأصغر من نحو 44 KiB. ويُغلق اتصال الخادم الأصلي الذي حمل جسم طلب بعد تبادله، ولا يعود أبدًا إلى المجمّع. يتلقى العملاء الذين ينتظرون `100-continue` هذا الرد محليًا قبل إرسال الجسم؛ وتتلقى التوقعات غير المدعومة الرمز 417. وتُستهلك ترويسة Expect قبل التمرير إلى الخادم الخلفي.

Forward Auth خلف وكيل دخول

في `--mode forward_auth` يجيب البرنامج الخدمي الطلب الفرعي لوكيل الدخول بالرمز `200` (مع ترويسات التدقيق `X-Sibuna-Status` و`X-Sibuna-Rule` و`X-Sibuna-Rule-Hash`)، أو `401` للتحدي، أو `403` للرفض، أو `429` عند تجاوز حد المعدل. يجب أن يمرر وكيل الدخول عنوان العميل وعنوان URL الأصلي؛ ووضع forward-auth يثق بهما افتراضيًا. اربط هذا المستمع بعنوان خاص لكي لا يتصل به إلا وكيل الدخول. تستعيد `X-Forwarded-Uri` (Caddy) أو `X-Original-URI` (Nginx)، ومعهما `X-Forwarded-Method`، طلب التطبيق لتقييم السياسة؛ ويوضع المسار الأصلي في الصيغة المعيارية، كما يوضع مسار سطر الطلب. تُرفض حقول URL الأصلي المكررة أو المتعارضة، وURI الأصلي الذي ليس بصيغة origin-form، بالرمز `400`. تستخدم المسارات الداخلية للبرنامج الخدمي دائمًا URI الطلب الفعلي. ولا يفحص Forward-auth جسمًا يحذفه وكيل الدخول.

```
map $http_upgrade $sibuna_connection_upgrade {
    default upgrade;
    '' close;
}
server {
    listen 443 ssl;
    location / {
        auth_request /__sibuna_auth;
        auth_request_set $sibuna_auth_status $upstream_status;
        auth_request_set $sibuna_retry_after $upstream_http_retry_after;
        auth_request_set $sibuna_status $upstream_http_x_sibuna_status;
        auth_request_set $sibuna_rule $upstream_http_x_sibuna_rule;
        auth_request_set $sibuna_rule_hash $upstream_http_x_sibuna_rule_hash;
        error_page 401 = @sibuna_challenge;
        error_page 500 = @sibuna_auth_error;
```

```
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection $sibuna_connection_upgrade;
        proxy_set_header Host $host;
        proxy_set_header X-Sibuna-Status $sibuna_status;
        proxy_set_header X-Sibuna-Rule $sibuna_rule;
        proxy_set_header X-Sibuna-Rule-Hash $sibuna_rule_hash;
        proxy_read_timeout 300s;
        proxy_pass http://127.0.0.1:3000;
    }
    location = /__sibuna_auth {
        internal;
        proxy_pass http://127.0.0.1:8080/;
        proxy_pass_request_body off;
        proxy_set_header Content-Length "";
        proxy_set_header X-Original-URI $request_uri;
        proxy_set_header X-Forwarded-Method $request_method;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-Forwarded-Uri "";
        proxy_set_header X-Forwarded-For $remote_addr;
        proxy_set_header User-Agent $http_user_agent;
        proxy_set_header Cookie $http_cookie;
    }
    location @sibuna_challenge {
        rewrite ^ /__sibuna/challenge break;
        proxy_pass http://127.0.0.1:8080;
        proxy_pass_request_body off;
        proxy_set_header Content-Length "";
        proxy_set_header X-Original-URI $request_uri;
        proxy_set_header X-Forwarded-Method $request_method;
        proxy_set_header X-Forwarded-For $remote_addr;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
    location @sibuna_auth_error {
        add_header Retry-After $sibuna_retry_after always;
        if ($sibuna_auth_status = 429) { return 429; }
        return 503;
    }
    location /__sibuna/ {
        proxy_pass http://127.0.0.1:8080;
        proxy_set_header X-Forwarded-For $remote_addr;
    }
}
```

```
example.com {
    handle /__sibuna/* {
        reverse_proxy localhost:8080
    }
    handle {
        forward_auth localhost:8080 {
            uri /
            header_up X-Forwarded-For {remote_host}
            header_up -X-Original-URI
            copy_headers X-Sibuna-Status X-Sibuna-Rule X-Sibuna-Rule-Hash
```

```
        }
        reverse_proxy localhost:3000
    }
}
```

يجب أن يصل فضاء الأسماء */sibuna__/ (الصفحة الوسيطة، والتحدي، والتحقق، وموارد برنامج الحل) إلى البرنامج الخدمي مباشرة في الوصفتين كلتيهما. تمرر صفحة الخطأ في nginx الـ URI والطريقة الأصليين لكي تحمل الصفحة الوسيطة المتطلب الذي قرره التفويض؛ ومن دونهما تظل الصفحة تعمل، ويعود الإصدار إلى تقييم عنوان URL الذي يبلغ عنه المتصفح. يوفر Caddy البيانات الوصفية للـ URI والطريقة الأصليين بنفسه. تضمن كتل handle الحصرية ألا تدخل مسارات التحدي والتحقق الفحص المسبق لـ forward-auth. يتطلب Nginx ترويستي Upgrade/Connection الصريحتين المبينتين أعلاه لـ WebSockets التطبيق. يعرض معالج الأخطاء في Nginx مسار التحدي الداخلي من دون اتخاذ قرار تفويض ثانٍ ومن دون استهلاك جسم الرفع. لا تقبل وحدة المصادقة فيه مباشرة إلا 2xx و401 و403، لذلك يحوّل المثال خطأ مصادقة ناتجًا عن حد المعدل إلى 429 مع Retry-After؛ وتبقى إخفاقات التفويض الأخرى مغلقة بالرمز 503. يوفر Nginx جسم 403 الخاص به، بينما يمرر Caddy صفحة الرفض من Sibuna. تستبدل الوصفتان كلتاهما ترويسات تدقيق Sibuna الواردة باستجابة التفويض الفعلية. اضبط شهادات TLS المعتادة وحدود الرفع والمهل الخاصة بالتطبيق عند وكيل الدخول؛ فالأمثلة تبين منطق التوجيه. لا يختبر المستودع هذه الوصفات بعدُ مع وكلاء دخول حقيقيين؛ أما قواعد البيانات الوصفية الممررة التي تعتمد عليها فتغطيها اختبارات الوحدات في sibuna-http.

بصمات JA4 من وكيل الدخول

تلخص بصمة JA4 الطريقة التي يفتح بها العميل اتصال TLS. وتُحسب من أول رسالة مصافحة يرسلها العميل، وهي ClientHello. تثبّت مكتبة TLS لدى العميل هذه الرسالة قبل إرسال أي بايت من HTTP، لذلك لا يستطيع العميل تغيير بصمته من داخل طلبات HTTP التي يرسلها. وتختلف فيها المتصفحات والمكتبات والروبوتات. وهذا يجعل JA4 إشارة أقوى من قيمة User-Agent، التي يستطيع أي عميل أن يكتبها كما يشاء (SID 0014). عرّفت FoxIO بصمة JA4. للبصمة 36 محرفًا في ثلاثة أجزاء يصل بينها _:

```
t13d1516h2_8daaf6152771_02713d6af862
^^^^^^^^^^ ^^^^^^^^^^^^ ^^^^^^^^^^^^

a          b            c

a  البروتوكول (t: TCP، q: QUIC، d: DTLS)، وإصدار TLS (13 تعني TLS 1.3)،
   ثم d مع SNI أو i بدونه، وعدد مجموعات التشفير (رقمان)،
   وعدد الامتدادات (رقمان)، ورمز ALPN h2) تعني HTTP/2)
b  تجزئة من 12 محرفًا ست عشريًا بأحرف صغيرة لمجموعات التشفير بعد فرزها
c  تجزئة من 12 محرفًا ست عشريًا بأحرف صغيرة للامتدادات
   ولخوارزميات التوقيع، بعد فرزها
```

لا يرى ClientHello إلا المكون الذي ينهي TLS، لذلك لا يستطيع حساب JA4 غيره. لا ينهي Sibuna اتصالات TLS: فهو يخدم HTTP/1.1 العادي خلف وكيل دخول، ووكيل الدخول هو الذي يتحدث TLS وHTTP/2

وHTTP/3 مع المتصفحات. ويحسب عدد من وكلاء الدخول JA4 بالفعل. لذلك ندع وكيل الدخول يحسب البصمة ويرسلها في ترويسة طلب، ويقرؤها Sibuna من هناك.

يسمي `--ja4-header <name>` تلك الترويسة. وهو معطل افتراضيًا وليس له اسم افتراضي؛ فنحن نسمي دائمًا الترويسة التي يستخدمها وكيل الدخول. يخزن Sibuna الاسم بأحرف صغيرة ويطابقه دون تمييز حالة الأحرف. ويرفض أسماء الحقول التي يكتبها أو يحذفها بنفسه (`X-Forwarded-*` و`X-Real-IP` و`X-Sibuna-*` و`Forwarded`، وكذلك مع _ بدل -). عندما يُضبط الخيار، تطبع رسالة البدء سطرًا مثل `JA4: from cf-ja4 (trusted)`.

لا تُقرأ الترويسة إلا عندما تكون الحقول الممررة موثوقة. وهو القرار نفسه الذي يحكم `X-Forwarded-For`: `--trust-forwarded`، أو وضع forward auth من دون `--no-trust-forwarded`. وتُتجاهل الترويسة إذا جاءت من نظير غير موثوق. ومع الثقة، يحمل الطلب بصمة عندما يوجد حقل واحد بالضبط بذلك الاسم وتكون قيمته على صيغة JA4. أما القيمة المعيبة، أو الحقل المكرر، فلا يعطي الطلب أي بصمة. يرصد ذلك عدادان: `sibuna_ja4_present_total` الطلبات التي تحمل بصمة، ويعد `sibuna_ja4_invalid_total` الطلبات التي كانت ترويستها معيبة أو مكررة. ولا يزيد الطلب الآتي من نظير غير موثوق أيًا من العدادين.

في وضع الوكيل العكسي، لا يرى الخادم الأصلي الترويسة إلا عندما يثق بها Sibuna. فمن دون ثقة، يحذف Sibuna كل حقل بذلك الاسم من الطلب الذي يرسله إلى الخادم الأصلي، حتى عندما يكتب العميل الاسم بـ _ بدل -. لذلك لا يستطيع العميل أن يسلم الخادم الأصلي بصمة مزورة تحت ذلك الاسم. ومع الثقة، يمر الحقل دون تغيير بكتابته الدقيقة، ويظل الاسم البديل الذي يستخدم _ محذوفًا. ومن دون `--ja4-header`، يكون الحقل الذي يحمل ذلك الاسم حقلًا عاديًا.

> **يجب أن يملك وكيل الدخول الترويسة** لا تكون البصمة جديرة بالثقة إلا بقدر وكيل الدخول الذي كتبها. لا يستطيع Sibuna أن يميز قيمة حسبها وكيل الدخول من قيمة أرسلها العميل. لذلك يجب على وكيل الدخول أن يستبدل، أو يحذف، أي نسخة من الترويسة يقدمها العميل، على كل مسار يصل إلى Sibuna. ويجب ألا يقبل مستمع Sibuna الاتصالات إلا من وكيل الدخول وحده، كما في `X-Forwarded-For`.

Cloudflare. يضيف التحويل المُدار "Add bot protection headers" الحقل `cf-ja4` إلى الطلب المرسل إلى الخادم الأصلي، بجانب `cf-bot-score` و`cf-verified-bot` و`cf-ja3-hash`. ويتطلب خطة Enterprise مع Bot Management. ويجب على أي وكيل دخول بين Cloudflare وSibuna أن يمرر الحقل دون تغيير. نشغّل Sibuna بالأمر:

```
sibuna --ja4-header cf-ja4 --trust-forwarded -s /etc/sibuna/secret
```

يجب ألا يقبل وكيل الدخول الذي أمام Sibuna الاتصالات إلا من عناوين Cloudflare وحدها؛ وإلا استطاع العميل أن يتجاوز Cloudflare ويرسل `cf-ja4` الخاص به.

nginx. تحسب الوحدة `ja4-nginx-module` من FoxIO بصمة JA4 داخل nginx وتعرضها باسم `$http_ssl_ja4`. وتحتاج إلى nginx 1.26 أو أحدث مع رقعة الوحدة لالتقاط ClientHello، مبنيًا مع

`--with-http_ssl_module`. يستبدل `proxy_set_header` حقل العميل الذي يحمل الاسم نفسه، لذلك تضبط كل كتلة location تصل إلى Sibuna الترويسة بنفسها:

```
# الوكيل العكسي: يعمل Sibuna مع --ja4-header x-ja4 --trust-forwarded
location / {
    proxy_set_header X-JA4 $http_ssl_ja4;
    proxy_set_header X-Forwarded-For $remote_addr;
    proxy_pass http://127.0.0.1:8080;
}

# Forward auth: يعمل Sibuna مع --mode forward_auth --ja4-header x-ja4
location = /__sibuna_auth {
    internal;
    proxy_pass http://127.0.0.1:8080/;
    proxy_set_header X-JA4 $http_ssl_ja4;
    # ... بقية حقول وصفة forward auth أعلاه
}
```

في وصفة forward auth، ينتمي السطر نفسه `proxy_set_header X-JA4 $http_ssl_ja4;` إلى `@sibuna_challenge` وإلى `location /__sibuna/` أيضًا. فكتلة location التي تُغفله تمرر `X-JA4` الخاص بالعميل إلى Sibuna، الذي يثق به.

Apache Traffic Server. يُفعَّل المكون الإضافي `ja4_fingerprint` بالسطر `ja4_fingerprint.so` في `plugin.config`. وليس له خيارات. ويضيف إلى طلبات كل اتصال TLS حقل `ja4` يحمل البصمة، وحقل `x-ja4-via` يحمل اسم الوكيل. نشغّل Sibuna بالخيارات `--trust-forwarded --ja4-header ja4`. ولا تذكر وثائق المكون الإضافي هل يستبدل حقل `ja4` الذي أرسله العميل. لذلك يجب أن نتحقق من ذلك في الإصدار الذي نستخدمه، أو أن نحذف حقل العميل قبل أن يضيف المكون الإضافي حقله، قبل أن نثق بالقيمة.

تطابق القاعدة البصمة بالحقل `ja4` (الجزء العاشر، مخطط السياسة). وتكون القيمة إما بصمة كاملة يجب أن تساويها البصمة، وإما بادئة يتبعها * يجب أن تبدأ بها البصمة. والقيمة "*" وحدها تطابق أي بصمة. ولا يطابق الطلب الذي لا يحمل بصمة أي قاعدة `ja4` أبدًا. ويرفض المحمّل النمط الذي لا يمكن أن تطابقه أي بصمة، مثل `T13D*` بأحرف كبيرة، مع `INVALID POLICY JA4`. وتقارن المطابقة البايتات ولا تخصص أي ذاكرة.

```
{ "rules": [
    { "name": "deny-known-scanner", "action": "DENY",
      "ja4": "t13i010400_0f2cb44170f4_5c4c70b73fa0" },
    { "name": "weigh-no-sni", "ja4": "t13i*", "action": "WEIGH", "weight": 15 },
    { "name": "weigh-headless", "user_agent": "Headless", "action": "WEIGH",
      "weight": 30 },
    { "name": "challenge-tls12-checkout", "path": "/checkout/*", "ja4": "t12*",
      "action": "CHALLENGE" }
] }
```

ترفض القاعدة الأولى بصمة واحدة بعينها؛ والقيمة مثال، لا أداة مسح معروفة. وتضيف القاعدتان التاليتان وزنًا إلى درجة الطلب بدل أن تحسما القرار: عميل TLS 1.3 لم يرسل اسم خادم، ومتصفح بلا واجهة يُعرف من `User-Agent` الخاص به. وما لم تحسم قاعدة لاحقة القرار أولًا، فإن الطلب الذي تظهر فيه الإشارتان معًا ينال الدرجة 45 ويبلغ

عتبة الرفض الافتراضية 40، بينما يتلقى الطلب الذي تظهر فيه إحداهما تحديًا. وتتحدى القاعدة الأخيرة، بالبادئة، كل عميل تفاوض على TLS 1.2 عبر TCP في مسار الدفع. وهي تتحدى ولا ترفض، لأن كثيرًا من العملاء الأقدم المشروعين ما زالوا يستخدمون TLS 1.2. وينبغي أن تجمع القواعد بين JA4 وإشارات أخرى على هذا النحو. فقيمة JA4 تسمي مكتبة TLS وإعدادها، لا شخصًا ولا جهازًا. ويتشارك كثير من العملاء مكتبة واحدة، ويمكن لمكتبات متخصصة أن تقلد ClientHello الخاص بمتصفح. يرفع JA4 تكلفة الأتمتة؛ لكنه لا يثبت الهوية. وللسبب نفسه يبقي Sibuna الحظر والسمعة مرتبطين بالعنوان، لا بالبصمة أبدًا.

تسجل الحوادث (حالات الرفض من WAF، وإصابات المصيدة، ونتائج التدقيق) بصمة طلبها عندما يحمل بصمة، في العمود `security_incidents.ja4`. وتعرضها وحدة التحكم في أدلة الحادث باسم "JA4 fingerprint"، مع القيمة أو "Not recorded". وتقبل عوامل التصفية في Events بصمة أو بادئة. ولمحرر القواعد المُدارة حقل JA4 يُخزن في `policies.ja4_pattern`، ويقبل نموذج اختبار السياسة بصمة.

السياسة التصريحية

> **الأهداف** اكتب ملف سياسة يضم القواعد، والأوزان، والعتبات، وقواعد IP، ومفتاح WAF.

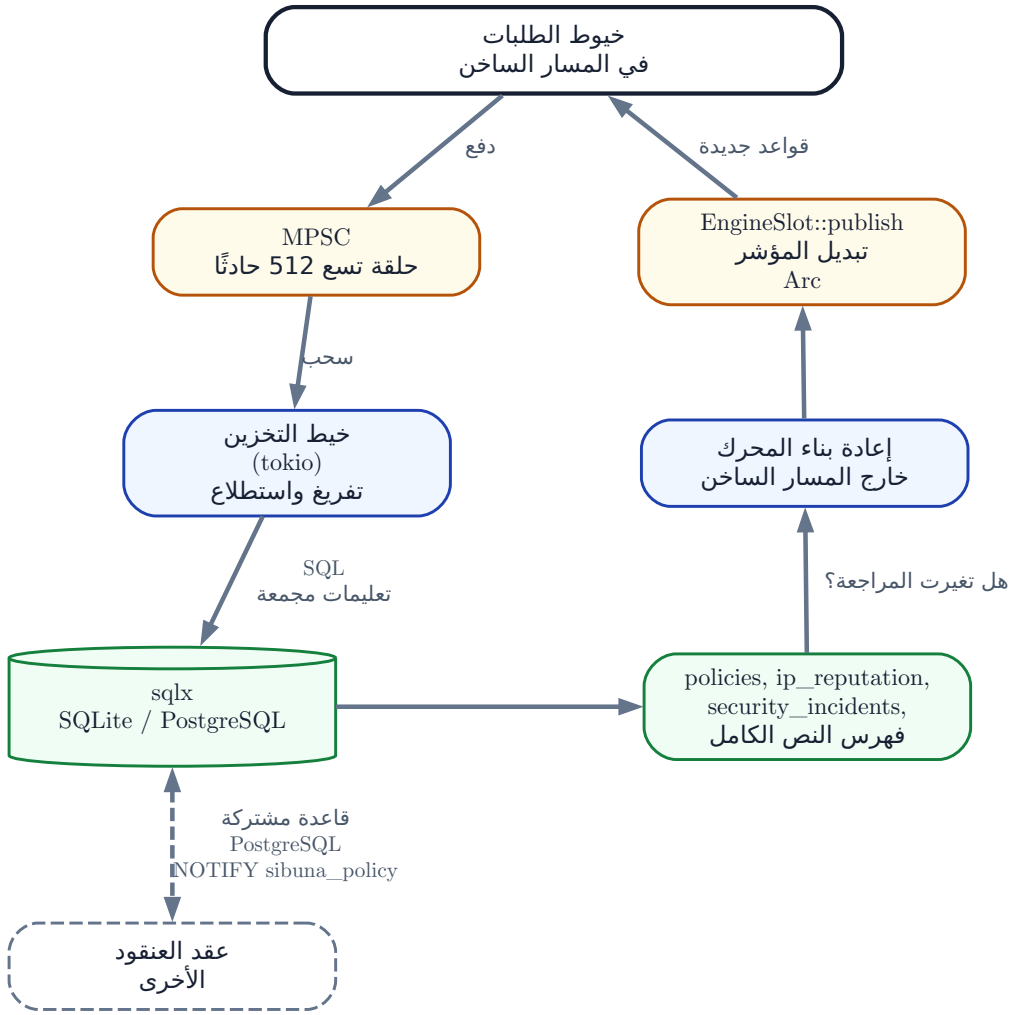
```
{
  "default_action": "CHALLENGE",
  "waf": true,
  "thresholds": { "challenge_at": 10, "deny_at": 40, "bits_step": 5 },
  "ip_rules": { "10.0.0.0/8": "ALLOW", "192.0.2.0/24": "DENY", "2001:db8::/32":
"DENY" },
  "rules": [
    { "name": "deny-cf-workers", "headers": { "CF-Worker": ".*" }, "action":
"DENY" },
    { "name": "deny-amazonbot", "user_agent": "Amazonbot", "action": "DENY" },
    { "name": "api-with-key", "path": "/api/*", "headers": { "X-Api-Key": ".*" },
    "action": "ALLOW" },
    { "name": "protect-checkout", "path": "/checkout/*", "action": "CHALLENGE",
      "challenge": { "difficulty": 20, "algorithm": "posw" } },
    { "name": "headless", "user_agent": "Headless", "action": "WEIGH", "weight":
30 },
    { "name": "internal-vpc", "remote_addresses": ["10.0.0.0/8", "fd00::/8"],
    "action": "ALLOW" }
  ]
}
```

يفشل المحمّل مغلقًا: مفتاح مجهول، أو إجراء مكتوب خطأً، أو عنوان معيب، أو رقم خارج النطاق يرفض الملف كله، ويسمي التشخيص القاعدة والحقل، ولا يبدأ البرنامج الخدمي. ويُرفض سطر الأوامر الذي فيه خيار مجهول أو قيمة خارج النطاق بالطريقة نفسها. يحل `rules` محل الجدول المدمج عند وجوده؛ ويغذي `ip_rules` شجرة السمعة، التي تتسع لآلاف البادئات؛ ويختار `waf: false` وضع Gate من الملف. ومفتاحان آخران جزء من المخطط (الجزء العاشر): يضبط `inspection` كل فئة من فئات WAF على `disabled` أو `audit` أو `enforce`، ويمنح كائن `limits` في القاعدة حصة خاصة بها. ويطابق الحقل `ja4` في القاعدة بصمة JA4 التي مررها وكيل دخول موثوق، تطابقًا تامًا

أو ببادئة يتبعها *؛ ويحتاج إلى `--ja4-header` (انظر "بصمات JA4 من وكيل الدخول"). تطابق أنماط المسارات مسار الطلب في صيغته المعيارية (SID 0003)، لذلك يجب أن يُكتب النمط بتلك الصيغة.

التخزين الدائم والعنقدة

الأهداف فعّل تخزين SQLite أو PostgreSQL، وأضف سياسة ديناميكية وحظرًا باستخدام SQL، واستعلم عن الأدلة الجنائية للحوادث، وافهم تجميع الحملات، وشغّل عدة عقد على قاعدة بيانات PostgreSQL واحدة.



الشكل 25 معمارية التخزين: لا يتوقف العمال أبدًا انتظارًا لقاعدة البيانات

يفتح `--data-dir /var/lib/sibuna` ملف SQLite `/var/lib/sibuna/sibuna.db` (وضع WAL، و`synchronous=NORMAL`، ومهلة انشغال) ويبدأ خيط التخزين. ويتصل `--database-url` (أو

`--database-url-file`) بقاعدة بيانات PostgreSQL بدلًا من ذلك. يملك خيط التخزين بيئة تشغيل tokio أحادية الخيط ومجمّع اتصالات صغيرًا؛ وهو المكان الوحيد الذي ينفذ SQL. عند البدء، وكلما تغيرت مراجعة السياسة (كتابة في `policies` أو `ip_reputation` أو `policy_inspection`)، وعند انتهاء مدخل سمعة، يعيد الخيط بناء المحرك من ملف السياسة وقاعدة البيانات وينشره؛ ولا تنتظر الطلبات SQL أبدًا. تقرأ إعادة البناء المراجعة قبل التحميل وبعده، وتُسقط مرشحها إذا ثبّت كاتبٌ معاملةً بينهما.

المخطط

الجدول 25 الجداول الأساسية

الجدول	الغرض والمفتاح
`policy_revision`	مراجعة السياسة، صف واحد؛ وتقدّمها المشغلات (triggers) على جداول السياسة الثلاثة.
`policies`	قواعد ديناميكية مرتبة، مفتاحها `id`. ويحمل `ja4_pattern` القابل للقيمة NULL معيار JA4 للقاعدة، بطول 37 بايت على الأكثر.
`policy_inspection`	وضع WAF المخزن لكل فئة، صف واحد.
`ip_reputation`	الدرجات وأعداد الإصابات والانتهاء، مفتاحها العنوان أو البادئة.
`incident_receipts`	إيصال الحوادث الدائم لكل عقدة (التسلسل التالي بعد آخر دفعة مثبَّتة لها)، مفتاحه العقدة.
`security_incidents`	سجل حدث محدود واحد مع تضمينه ذي 64 مكونًا (256 بايت)، مفتاحه `sequence | node_id << 40`. ويحمل `ja4` القابل للقيمة NULL بصمة JA4 للطلب، بطول 36 بايت بالضبط.
`incidents_fts`	في SQLite: جدول FTS5 على الوحدات المعجمية (tokens) المعيارية للمسارات والحمولات. وفي PostgreSQL تكون الوحدات نفسها هي العمود `search` من النوع `tsvector` في `security_incidents` مع فهرس GIN.
`sibuna_schema`	كل ترحيل مطبق، مفتاحه المكون والإصدار.

المخطط الأساسي هو `crates/storage/src/schema.rs`، مكتوبًا مرة واحدة لكل لهجة SQL. ومع وحدة التحكم يضيف المكون `console` 33 جدولًا تبدأ أسماؤها بـ `console_` (SID 0013؛ `crates/console-storage/src/schema/`). تُرقّم الترحيلات لكل مكون وتُترجم داخل الملف التنفيذي. ويضيف ترحيل النواة رقم 2 عمودي JA4؛ وتُرحَّل قاعدة البيانات التي أُنشئت قبله عند البدء. وتجري كل الترحيلات المعلقة في معاملة واحدة، وترفض العقدة قاعدة بيانات يكون فيها إصدار مكون أحدث مما تعرفه. والمخطط المطبوع المختصر ليس سكربت ترقية. جداول SQLite من نوع `STRICT`، وقيود `CHECK` تعكس أنواع Rust، لذلك ترفض

قاعدة البيانات صفًا لا يستطيع البرنامج الخدمي تمثيله. تُخزن القيم المعدّدة بأحرف صغيرة: الكتابة المباشرة لـ 'DENY' تُرفض.

تشغيله

استخدم صدفة sqlite3 على sibuna.db (يتيح وضع WAL القراءة أثناء عمل البرنامج الخدمي)، أو psql على قاعدة بيانات العنقود:

```
-- أضف قاعدة ديناميكية؛ تعيد كل عقدة بناء محركها في نبضتها التالية.
INSERT INTO policies (id, name, priority, path_pattern, action, difficulty,
algorithm,
  header_matchers, cidr_matchers, weight, enabled, created_at, updated_at)
VALUES ('p-checkout', 'protect-checkout', 10, '/checkout/*', 'challenge', 20,
'posw',
  '{"X-Api": "v2"}', '["10.0.0.0/8"]', 0, 1, unixepoch(), unixepoch());

-- احظر عنوانًا على مستوى العنقود كله يومًا كاملًا.
INSERT INTO ip_reputation (ip_or_cidr, reputation_score, banned_until,
  trigger_rule, hits, last_seen)
VALUES ('198.51.100.7', -100, unixepoch() + 86400, 'analyst', 1, unixepoch());

-- الأدلة الجنائية: بحث في النص الكامل للحمولات المسجلة.
SELECT s.id, s.client_ip, s.violation_category, s.path, s.campaign_id
FROM incidents_fts f JOIN security_incidents s ON s.id = f.rowid
WHERE incidents_fts MATCH 'union' ORDER BY rank LIMIT 20;
```

التعليمات مكتوبة لـ SQLite. في PostgreSQL، تكون enabled هي TRUE، ويكون الوقت CAST(extract(epoch FROM now()) AS BIGINT) بدل unixepoch()، ويقرأ البحث WHERE search @@ CAST('union' AS tsquery) على security_incidents. يحمل path و offending_payload بايتات الطلب الخام (BLOB أو BYTEA). الكتابة المباشرة التي تخالف قيدًا ترفضها قاعدة البيانات، ولا تصل أبدًا إلى عقدة. وفي PostgreSQL يصدر مشغل المراجعة أيضًا NOTIFY sibuna_policy، فتتقدم العقد الأخرى فورًا؛ ويبقى الاستطلاع هو الضمان.

تصبح الدرجات التي تساوي 50− أو تقل عنها بادئات deny في الشجرة، وتصبح الدرجات التي تساوي 50 أو تزيد عليها allow، ويحد banned_until مدة الحظر. تُدرج إصابات المصيدة سجلًا بقيمة 100− تلقائيًا، لذلك مع قاعدة بيانات PostgreSQL مشتركة، يحظر الفخ الذي يُستثار على عقدة واحدة العنوانَ على العقد كلها.

إيصال المعاملة يجعل إعادة المحاولة آمنة

يجمع خيط التخزين 32 سجلًا معلقًا على الأكثر ويبني معاملة SQL واحدة. تُدرج المعاملة الحوادث، وتحدّث فهرس النص الكامل، وتطبق تغييرات السمعة الناتجة عن المصيدة. لكل مُصدِر (عقدة) إيصال يتزايد رتيبًا، هو صفه في incident_receipts. كل تعليمة تغير البيانات محروسة بذلك الإيصال؛ وتقدّمه المعاملة في النهاية، ولا ترجعه إلى الوراء أبدًا.

مثال محلول: الإقرار يختفي لنفترض أن المُصدِر 2 يجهز التسلسلات من 101 إلى 132، منتهيًا عند الإيصال 133. تثبّت قاعدة البيانات كل السجلات والإيصال 133، ثم يفشل الاتصال قبل وصول الإقرار. يحتفظ خيط التخزين بالدفعة المجهزة (المعرفات، والتضمينات، والحملات) ويعيد محاولتها. الإيصال يساوي 133 بالفعل، لذلك لا تفعل الكتابات المحروسة شيئًا: لا تتضاعف الحوادث ولا أعداد إصابات المصيدة. وإذا تراجعت المعاملة الأصلية، يبقى الإيصال القديم وتطبق إعادة المحاولة كل الكتابات مرة واحدة.

التثبيت الفاشل يحتفظ بالدفعة المعلقة في الذاكرة ويزيد `incident_write_failures`. ومع ذلك تحاول النبضة استطلاع السياسة. والطابور الممتلئ بسعة 512 سجلًا يعد `incidents_dropped`؛ ولا يعطّل أبدًا استجابة HTTP انتظارًا للقرص. وتضيف الدفعة المعلقة مساحة لـ 32 سجلًا. قد يؤدي موت العملية قبل التثبيت إلى فقد البيانات المصفوفة في الطابور: هذا مسار جنائي غير متزامن محدود، وليس طابور رسائل دائمًا لحظة الإدراج.

تمرين 9.1. انقل تحديث الإيصال إلى خارج معاملة البيانات. صمّم تسلسلًا للانهيارات يضاعف أثر سمعة، وتسلسلًا آخر يُضيّع حادثًا.

تجميع الحملات

تُمثَّل كل حمولة حادث بمتجه وحدة من 64 بعدًا عبر حيلة التجزئة على ثلاثيات البايتات (مع طي الأرقام وطي حالة الأحرف)، وتُخزن في 256 بايت في `security_incidents.embedding`. يحتفظ خيط التخزين بأحدث 16,384 تضمينًا من كل العقد في الذاكرة (4 MiB)، ويجددها من قاعدة البيانات في كل نبضة. قبل الإدراج يجد أقربها، بما في ذلك السجلات السابقة في الدفعة نفسها؛ فإذا كانت مسافة جيب التمام 0.35 على الأكثر انضم الحادث الجديد إلى `campaign_id` لذلك الحادث، وإلا بدأ حملة تحمل اسم معرفه هو. لا تُقارن الحوادث الأقدم من النافذة. التشابه استدلالي: تجمع أمثلة الانحدار حمولات SQL المترابطة، لكن ذلك لا يضمن أن كل زوج من الهجمات يشترك في حملة. لا يعمل أي نموذج ولا يُدرَّب شيء.

العنقدة

العنقود مجموعة من العقد تتشارك قاعدة بيانات PostgreSQL واحدة. ابنِ مع الميزة `postgres-tls` (`cargo build --release -p sibuna --features postgres-tls`؛ وهي تتضمن `postgres`) وابدأ كل عضو بقاعدة البيانات ومعرف عقدة خاص به:

```
sibuna --database-url-file /etc/sibuna/database-url --cluster-node 1 \
       --secret-file /etc/sibuna/secret
```

يجب أن يستخدم كل عضو السر الرئيسي نفسه وقيمة `--cluster-node` فريدة من 1 إلى 8,388,607. يحمل ملف URL عنوان `postgres://` واحدًا ويجب ألا يقرأه إلا مالكه؛ ولا تطبع رسالة البدء عنوان URL أبدًا. تذهب الكتابات إلى خادم PostgreSQL الأساسي. يرى خيط التخزين في كل عقدة تغير المراجعة المثبَّتة (عبر `NOTIFY` أو بالاستطلاع) ويعيد بناء محركه.

تصل العقد إلى قاعدة البيانات المشتركة عبر الشبكة، لذلك نشفّر ذلك الاتصال ونتحقق منه. نمنح الخادم شهادة موقعة من سلطة الشهادات الخاصة بنا، ونسمي تلك السلطة في عنوان URL: `postgres://sibuna:...@db.internal:5432/sibuna?sslmode=verify-full&sslrootcert=/etc/sibuna/db-ca.pem`.

وعلى الخادم لا نقبل TCP إلا عبر أسطر `hostssl` في `pg_hba.conf`.

عند بدء التشغيل يطبع البرنامج الخدمي سطرًا واحدًا عن اتصال قاعدة البيانات. مع `verify-full` يقول إن الشهادة واسم المضيف متحققٌ منهما. ومع أي وضع أضعف يطبع تحذيرًا وتلميحًا. ومن دون `sslmode` يكون الوضع `prefer`: لا يُشفَّر الاتصال إلا إذا عرض الخادم TLS، ولا يُتحقق من الخادم. والبرنامج الخدمي المبني بلا `postgres-tls` يرفض `require` و`verify-ca` و`verify-full` ويسمي الميزة الناقصة؛ ولا يعود أبدًا إلى النص الصريح. والشهادة المرفوضة توقف بدء التشغيل فورًا؛ ويطلب منا التلميح أن نتحقق من `sslrootcert` ومن أن مضيف عنوان URL اسم وارد في شهادة الخادم. يضيف `sslrootcert` سلطتنا إلى مخزن الثقة في النظام؛ ولا يزيل سلطات النظام (SID 0011، SID 0012).

لا ينسخ Sibuna قاعدة بيانات خاصة به ولا ينتخب قائدًا. النسخ المتماثل، والتحويل عند الفشل، والنسخ الاحتياطي من مسؤولية PostgreSQL (النسخ المتدفق، أو Patroni، أو خدمة مُدارة). لا قائد في عرض العقد في وحدة التحكم: يعرض الدور `single` لعقدة SQLite و`unknown` لعقدة PostgreSQL، ولا يعني `quorum` إلا أن آخر رحلة ذهاب وإياب للعقدة إلى قاعدة البيانات نجحت. إذا تعذر الوصول إلى قاعدة البيانات، يواصل آخر محرك منشور الخدمة، وتُصف الحوادث في الحلقة المحدودة وتُسقط مع عداد حين تمتلئ، ويعيد خيط التخزين المحاولة مع تراجع تدريجي؛ وتحتفظ العقدة بآخر سياسة لديها حتى تعيد الاتصال. لا يُقرأ مجلد بيانات Zaxonlite (SID 0005)؛ تنتقل السياسات عبر `sibuna console policies export` و`sibuna console policies import`، ولا يُرحَّل التاريخ الجنائي.

توجيه التحديات في العنقود

مفاتيح التحدي مربوطة بـ `--cluster-node`؛ لذا أبقِ إصدار التحدي والتحقق منه على العضو نفسه. تستخدم الرموز البذرة المشتركة وتعمل على الأعضاء الآخرين. لا تصبح حصص المعدل المحلية حصصًا عامة، وتمحو إعادة تشغيل العملية مجموعات الحلول المُنفَقة. مع ثلاث عقد على قاعدة بيانات PostgreSQL محلية واحدة، يصل صف سياسة مكتوب مباشرة إلى العقد الثلاث خلال نبضة واحدة، ويصل حظر المصيدة على عقدة إلى الأخريات، ولا تتصادم معرفات الحوادث أبدًا (`crates/storage/tests/multi_node.rs`، المعيار 2 من SID 0011؛ ويجري عند ضبط `SIBUNA_TEST_DATABASE_URL`).

التحزيم

> **الأهداف** نثبّت حزمة إصدار ونتحقق منها، ونشغّل Sibuna خدمةً، أو في حاوية، أو تحت Kubernetes، ونبني الملف التنفيذي بأنفسنا.

يحمل كل إصدار على GitHub الملف التنفيذي نفسه لكل منصة، مبنيًا من وسم الإصدار بالميزات الافتراضية: تخزين SQLite ووحدة التحكم. أما عناقيد PostgreSQL فتحتاج إلى بناء خاص بنا (انظر أدناه).

المنصة	ملفات الإصدار
Linux x86-64، ARM64	يحمل `sibuna-linux-amd64.tar.gz` و`sibuna-linux-arm64.tar.gz` ملفًا تنفيذيًا ثابت الربط مع musl يعمل على أي توزيعة. وتثبّته `sibuna_<version>-1_amd64.deb` و`sibuna_<version>-1_arm64.deb` و `sibuna-<version>-1.x86_64.rpm` و`sibuna-<version>-1.aarch64.rpm` مع خدمة systemd؛ وتفعل `sibuna-bin-<version>-1-x86_64.pkg.tar.zst` ذلك على Arch Linux.
macOS arm64، x86-64	`sibuna-macos-arm64.tar.gz` و`sibuna-macos-amd64.tar.gz`، لنظام macOS 12 أو أحدث.
Windows x86-64	يحمل `sibuna-windows-amd64.zip` الملف `sibuna.exe`، لنظام Windows 10 أو Server 2016 أو أحدث.
FreeBSD 15.1، OpenBSD 7.9 (amd64)	`sibuna-<version>-freebsd-15.1-amd64.pkg` و`sibuna-<version>.tgz`.
Homebrew وKubernetes والحاويات	`sibuna.rb`، و`helm-sibuna-<version>.tgz`، والصورة `ghcr.io/insanai/sibuna:<version>` لمعماريتي amd64 وarm64، وخلاصتها في `IMAGE-DIGEST.txt`.
المصدر	`sibuna-<version>-source.tar.gz`.

يحمل كل أرشيف وكل حزمة `LICENSE` و`NOTICE` و`LICENSES/` (مع `THIRD-PARTY-RUST.txt`، أي حزم Rust (crates) المضمّنة في الملف التنفيذي) و`README.md` و`SECURITY.md` و`sibuna.build.json`. يسجل البيان الإصدار، والإيداع، وهدف Rust، وإصدار المترجم، والميزات، وقيمة SHA-256 للملف التنفيذي ولوحدتي WebAssembly الخاصتين به. ويشغّل سير عمل الإصدار كل ملف تنفيذي على منصته قبل أن ينشر أي شيء، ويثبّت الحزم ويعيد تثبيتها ويزيلها على Debian وFedora وArch Linux وFreeBSD وOpenBSD.

ملفات الإصدار غير موقعة. نتحقق من الملف المنزَّل مقابل `SHA256SUMS` من الإصدار نفسه قبل أن نثبته:

```
sha256sum --ignore-missing -c SHA256SUMS          # Linux
shasum -a 256 --ignore-missing -c SHA256SUMS      # macOS, FreeBSD
sha256 -C SHA256SUMS sibuna-1.0.0-beta.3.tgz      # OpenBSD
```

على macOS، يحمل الملف المنزَّل عبر المتصفح سمة حجر صحي توقف أي ملف تنفيذي غير موقع؛ ويزيلها `xattr -d com.apple.quarantine sibuna` بعد التحقق.

خدمة systemd

تثبّت حزم Debian وRPM وArch الملف /usr/bin/sibuna، والوحدة sibuna.service، والحساب sibuna (عبر systemd-sysusers)، والأداة المساعدة /usr/lib/sibuna/seed.sh، وملف الإعدادات /etc/sibuna/service.env. ولا يبدأ التثبيت أي شيء. نضبط الخادم الأصلي، بعنوان IP صريح، في ملف الإعدادات ثم نبدأ الخدمة:

```
sudoedit /etc/sibuna/service.env      # SIBUNA_UPSTREAM_HOST, SIBUNA_UPSTREAM_PORT
sudo systemctl enable --now sibuna
curl -s http://127.0.0.1:8080/__sibuna/health
```

تستمع الخدمة على 127.0.0.1:8080 ما لم يحدد SIBUNA_HOST وSIBUNA_PORT غير ذلك. ويمرر SIBUNA_OPTIONS خيارات إضافية مفصولة بمسافات، مثل --mode forward_auth أو --console 127.0.0.1:19446. قبل كل بدء، ينشئ seed.sh بذرة القبول /var/lib/sibuna/admission.seed (64 رقمًا ست عشريًا، بالصلاحيات 0600، ومالكها sibuna) إن لم تكن موجودة؛ ولا يستبدل بذرة موجودة أبدًا، لأن البذرة الجديدة تنهي كل جلسة. ويحفظ البرنامج الخدمي قاعدة بيانات SQLite الخاصة به في /var/lib/sibuna/، وهو دليل بالصلاحيات 0700. وتعمل الوحدة دون قدرات (capabilities)، مع نظام ملفات للقراءة فقط (ProtectSystem=strict)، و/tmp وأجهزة خاصة، ودون امتيازات جديدة.

تحتفظ الترقية بملف الإعدادات والبذرة والحالة، ولا تعيد تشغيل الخدمة؛ ونعيد تشغيلها نحن بعد أخذ نسخة احتياطية. أما الإزالة فتوقف الخدمة وتعطلها، لكنها تحتفظ بالحساب والبذرة والإعدادات والحالة، ونحذفها بأنفسنا عندما نُخرج المضيف من الخدمة.

لا تثبّت حزم FreeBSD وOpenBSD إلا /usr/local/bin/sibuna ووثائقه (pkg add، أو pkg_add -D unsigned لحزمة من خارج مرايا OpenBSD). ولا تنشئ خدمة ولا حسابًا ولا حالة: نشغّل البرنامج الخدمي بمستخدم بلا امتيازات مع بذرة ودليل بيانات خاصين.

Homebrew

يثبّت sibuna.rb أرشيف الإصدار المناسب للجهاز: macOS أو Linux، وarm64 أو x86-64. ولا يبني من المصدر، لأن Rust في Homebrew لا تملك مكتبة قياسية لـ wasm32-unknown-unknown، وهي ما يحتاج إليه بناء وحدتي WebAssembly المدمجتين؛ والأرشيفات تحمل الملفات التنفيذية التي سبق أن فحصها سير عمل الإصدار. لا يثبّت Homebrew الصيغ إلا من المستودعات (taps)، لذا نضع الصيغة في مستودع محلي:

```
brew tap-new "$USER/local"
cp sibuna.rb "$(brew --repository)/Library/Taps/$USER/homebrew-local/Formula/"
brew install "$USER/local/sibuna"
```

لا تبدأ الصيغة أي خدمة. تبين ملاحظاتها كيف ننشئ بذرة؛ ويشغّل اختبارها البرنامج الخدمي مرتين على دليل البيانات نفسه.

الحاويات وKubernetes

تحمل الصورة `ghcr.io/insanai/sibuna:<version>` الملف التنفيذي الثابت على قاعدة distroless: شهادات CA، والمستخدم 65532، ولا صدفة ولا مدير حزم. ونقطة الدخول هي `sibuna`، لذا فالوسائط خيارات للبرنامج الخدمي. ويشغّلها سير عمل الإصدار للقراءة فقط، دون قدرات، مع بذرة مركّبة للقراءة فقط ووحدة تخزين للحالة، ثم يعيد تشغيلها مرة واحدة:

```
docker run -d --read-only --cap-drop=ALL --security-opt=no-new-privileges \
  -v "$PWD/admission.seed:/run/admission.seed:ro" -v sibuna-state:/var/lib/sibuna
\
  -p 8080:8080 ghcr.io/insanai/sibuna:1.0.0-beta.3 \
  --host 0.0.0.0 --secret-file /run/admission.seed --data-dir /var/lib/sibuna/data
\
  --upstream-host 10.0.0.5 --upstream-port 3000
```

يشغّل مخطط Helm `helm-sibuna-<version>.tgz` وحدة Pod واحدة من تلك الصورة. ويتطلب Secret ننشئه نحن (`secret.existingSecret`، بالمفتاح `admission.seed`)، فلا تظهر أي بذرة أبدًا في القيم أو القوالب. تعمل وحدة Pod بالمعرّف UID 65532 مع نظام ملفات جذري للقراءة فقط وفق معيار أمان Pod المسمى Restricted. ونسخة واحدة مع ترقيات `Recreate` تبقي كاتبًا واحدًا على وحدة التخزين ReadWriteOnce، حيث يحفظ Sibuna حالة SQLite في `/data`؛ وتبقى المطالبة بوحدة التخزين بعد إزالة التثبيت. وتستدعي الفحوص `/__sibuna/health`. والخدمة Service من نوع ClusterIP، ولا ينشئ المخطط ingress ولا وحدة تحكم، وتمنع NetworkPolicy الخاصة به كل حركة المرور حتى يسمح `networkPolicy.ingressFrom` و `networkPolicy.egressTo` بالعملاء والخادم الأصلي. يجب أن يكون `upstream.host` عنوان IP صريحًا، ويمرر `extraArgs` خيارات إضافية للبرنامج الخدمي. ويثبّت سير العمل المخطط ويرقّيه ويزيله في عنقود kind يفرض معيار Restricted.

```
kubectl -n sibuna create secret generic sibuna-admission --from-
file=admission.seed
helm install gateway ./helm-sibuna-1.0.0-beta.3.tgz --namespace sibuna \
  --set secret.existingSecret=sibuna-admission \
  --set upstream.host=10.0.0.5 --set upstream.port=3000
```

البناء من المصدر

نبني ونختبر باستخدام Rust 1.100؛ ويبقى Rust 1.94 أدنى إصدار مدعوم.

نثبت	سلسلة	أدوات	الإصدار	بالأمر

```
rustup toolchain install 1.100.0 --profile minimal --target wasm32-unknown-unknown
```

ونختارها بالأمر `cargo +1.100.0` في الأوامر التالية.

- ينتج `cargo build --release -p sibuna` الملف `target/release/sibuna` بالميزات الافتراضية: تخزين SQLite ووحدة التحكم. تضم sqlx مكتبة SQLite، ويحتاج `ring` (rustls) إلى مترجم C وقت البناء؛ ويربط الملف التنفيذي بيئة تشغيل C للنظام ديناميكيًا. يبني سكربت البناء أيضًا برنامج الحل WebAssembly

وواجهة وحدة التحكم، وهذا يحتاج إلى `rustup target add wasm32-unknown-unknown`؛ ويضمّن `SIBUNA_POW_WASM=<path>` و`SIBUNA_CONSOLE_WASM=<path>` وحدات مبنية مسبقًا بدلًا من ذلك.

- ينتج `cargo build --release -p sibuna --no-default-features` الجدار الناري في الذاكرة من دون أي حزمة خارجية على الإطلاق؛ وعندها يُرفض `--data-dir` و`--database-url` عند البدء بالخطأ `StorageDisabled`، ويُرفض `sibuna console` بالخطأ `CONSOLEBUILD`.
- يضيف `cargo build --release -p sibuna --features postgres-tls` عناقيد PostgreSQL مع TLS متحقق منه إلى قاعدة البيانات. ولا يحتاج إلى OpenSSL.
- تأتي الملفات التنفيذية الثابتة لنظام Linux في الإصدارات من `cargo build --release --locked -p sibuna --target x86_64-unknown-linux-musl` (أو `aarch64-unknown-linux-musl`) مع `musl-gcc` مترجمًا لـ C (`CC_x86_64_unknown_linux_musl=musl-gcc`).

يستخدم ملف تعريف الإصدار fat LTO ووحدة codegen واحدة. يبني سير العمل وحدتي WebAssembly مرة واحدة ويمررهما إلى كل منصة، فتضمّن كل الملفات التنفيذية لإصدار ما وحدات متطابقة بايتًا ببايت. ويعيد `cargo xtask licenses` توليد `LICENSES/THIRD-PARTY-RUST.txt` بعد تغير أي اعتمادية.

على Windows، يتوقف البرنامج الخدمي عند Ctrl+C وCtrl+Break وأحداث إغلاق الطرفية؛ ولا يعمل خدمةً من خدمات Windows. ولا تملك ملفات Windows بتات صلاحيات، لذا لا يستطيع Sibuna رفض ملف مفتاح أو بيانات اعتماد أو عنوان URL لقاعدة بيانات قد يقرؤه آخرون: نقيّد هذه الملفات بقوائم ACL. وتُقرأ مقاييس الذاكرة وCPU في وحدة التحكم من `/proc`، ولا تتوفر على الأنظمة الأخرى.

تمرين 9.2. اكتب ملف السياسة وصفوف `ip_reputation` اللازمة لكي تتجاوز شبكة تجهيز (`10.20.0.0/16`) التحديات، ويطلب `/admin/*` 20 بتًا من العمل المتسلسل، وتُقبل أداة كشط تابعة لشريك تُعرف بالترويسة `X-Partner-Key` بمعدل 30 طلبًا لكل 10 ثوانٍ.

تلميح: تضبط خيارات البرنامج الخدمي الحدَّ الخاص بكل عميل، وهو أول حد يمر به كل طلب؛ ويمكن أن تحمل القاعدة حصة خاصة بها في `limits`.

اشرح اللامتغير. اشرح لمشغل لماذا تسري إضافة صف إلى `policies` من دون إعادة تشغيل ومن دون أن ينتظر أي طلب أبدًا، مستعينًا بخيط التخزين وموضع المحرك.

الجزء العاشر

المرجع التفصيلي والتشخيص

نقاط النهاية، والمقاييس، وفهرس الأخطاء، ومخطط السياسة، وجداول التخزين في مكان واحد.

نقاط النهاية

الأهداف معرفة كل مسار تحت `/__sibuna__/` وما يعيده كل منها.

الجدول 27 المسارات المحجوزة

نقطة النهاية	الطريقة	الوظيفة
`/__sibuna/challenge`	GET	الصفحة الوسيطة
`/__sibuna/challenge.json?need=&path=`	GET	يصدر تحديًا عديم الحالة: تحدده تذكرة المتطلبات الواردة في استجابة التحدي، وإلا يُقيِّم عنوان URL المُبلَّغ عنه في صيغته المُطبَّعة (414 إذا تجاوز 8 KiB، و400 لمسار ليست له صيغة مُطبَّعة). يعيد `{id, algorithm, difficulty, challenges, expires_at}`
`/__sibuna/verify`	POST	يقبل `{"challenge_id", "nonce"}` أو `{"challenge_id", "proof"}`؛ ويعيد 200 مع `Set-Cookie`، أو 400 مع تشخيص، أو 413 لجسم أكبر من مخزن الاتصال المؤقت
`/__sibuna/wasm/sibuna-pow.wasm`	GET	وحدة برنامج الحل، بحجم 14,744 بايت (`wasm_bytes` في `benchmarks/results/primitives-latest.json`)، قابلة للتخزين المؤقت ساعة واحدة
`/__sibuna/worker.js`	GET	Web Worker يضم مُبرهِنَي WASM وJavaScript، قابل للتخزين المؤقت ساعة واحدة
`/__sibuna/honeypot`	GET	يحظر المستدعي مدة `--ban-seconds` ويسجل حادثًا؛ 403. لا يحظر إلا طلب المستوى الأعلى (بلا `Sec-Fetch-Dest` أو

نقطة النهاية	الطريقة	الوظيفة
		بقيمة document، وبلا Sec-Fetch-Site عابر للمواقع)؛ ويُجاب أي طلب آخر بـ 404 ويُحتسب
/__sibuna/health	GET	حالة الحيوية بصيغة JSON مع اسم المحرك، والإصدار، ووضع الوكيل، وخوارزمية الإثبات
/__sibuna/metrics	GET	تنسيق نص Prometheus (الإصدار 0.0.4)

يُجاب أي مسار آخر تحت /__sibuna/ بـ 404. يستهلك الإصدار والتحقق ميزانية --challenge-rate-limit المستقلة، ويُرفض العنوان المحظور بـ 403 على كل مسار. تحمل استجابات التحدي X-Sibuna-Status: CHALLENGE؛ وتحمل الطلبات المقبولة X-Sibuna-Status: PASS وX-Sibuna-Rule نحو الخادم الأصلي (وX-Sibuna-Rule-Hash، بالنظام الست عشري، في ردود forward-auth). يتلقى العميل الذي لا يقبل HTML الحالة 401 مع {"error":"challenge_required","challenge":"/__sibuna/challenge.json?need=<ticket>"}.

يقدم مستمع الإدارة الاختياري المستقل /console/. هيّئ حساب مسؤوله محليًا بالأمر sibuna init-admin، ثم استبدل كلمة المرور المؤقتة عند أول تسجيل دخول. يتطلب /console/api/stats و/console/ws جلسة موثقة (بما فيها جلسة الكشك). يدمج اتصال WebSocket الإحصاءات والأحداث والعقد والسياسة والتحديات والتدقيق في قناة واحدة، مع لقطات محدودة وفروق واستعادة للفجوات. يتصل نظراء الإدارة بـ /console/peer، وتقع كل مسارات API تحت /console/api/. لا تُرصد عدادات أخطاء الخادم الأصلي في وضع forward-auth. يوثق الجزء التاسع مسارات عمل وحدة التحكم، ونشرها عبر HTTPS، والحدود الحالية لـ SID 0007.

المقاييس

عدادات معروضة بالصيغة sibuna_<name>_total بتنسيق نص Prometheus، بهذا الترتيب:

الجدول 28 العدادات

العداد	يزداد عندما
incidents_persisted، incident_batches	السجلات والمعاملات التي أكد خيط التخزين تثبيتها
incidents_dropped، incident_write_failures	إضافات إلى الطابور رُفضت لامتلاء الحلقة، ومحاولات تثبيت فاشلة
requests	وصل طلب مُحلَّل إلى مرحلة التوجيه (بما في ذلك المسارات المحجوزة)
allowed، denied، challenged	اتُخذ قرار سياسة (قبول، أو رفض، أو إصدار تحدٍّ أو إعادة إصداره)

العداد	يزداد عندما
`challenges_issued`	أنشأ `/__sibuna/challenge.json` سجل تحدٍّ
`solutions_accepted`، `solutions_rejected`	قبل `/__sibuna/verify` إثباتًا أو رفضه
`rate_limited`	رفض GCRA طلبًا (429): حد الطلبات، أو ميزانية التحديات، أو حصة قاعدة
`banned`	رُفض عنوان محظور، أو حظرت المصيدة عنوانًا
`bans_issued`	سُجّل حظر (من المصيدة، أو حظرٌ تفرضه حصة قاعدة)
`proxied`، `upstream_errors`	مُرر طلب إلى الخادم الأصلي، أو فشل الخادم الأصلي (502)
`parse_errors`	رُفض رأس مشوه، أو `Host` مكرر، أو هدف ليست له صيغة مُطبَّعة (400)، أو ترميز نقل غير مدعوم (501)
`overloaded`	أُجيب اتصال يتجاوز `--max-connections` أو `--max-connections-per-ip` بـ 503
`bans_refused`	لم يُسجل حظر لأن كل موضع يمكن أن يأخذه مفتاحه كان يحمل حظرًا ساريًا
`forwarded_invalid`	تعذر تحليل عنوان عميل ممرر موثوق؛ فحُكم على الطلب بعنوان نظير المقبس
`honeypot_refused`	لم يحظر طلبُ المصيدة أحدًا، لأن الذي طلبها صفحة لا شخص
`rate_evictions`	أزاح وصولٌ إلى محدد المعدل خليةَ عميل نشط آخر
`ja4_present`	مع `--ja4-header` والتمرير الموثوق، حمل الطلب ترويسة واحدة بالضبط بذلك الاسم تحتوي على بصمة JA4
`ja4_invalid`	مع `--ja4-header` والتمرير الموثوق، كانت الترويسة مشوهة أو مكررة؛ فحُكم على الطلب دون بصمة

قد تزيد إعادة المحاولة `incident_write_failures` دون فقد سجلات، لأن الدفعة المعلقة تُستبقى. راقب الزيادات خلال فترة؛ فالمجاميع وحدها ليست عمق الطابور.

رموز الحالة

الجدول 29 رموز حالة مستوى البيانات

الرمز	متى
200	مقبول (forward-auth)، والصفحة الوسيطة (تصفّح HTML يحتاج إلى تحدٍّ)، والمسارات الداخلية
302	لا يستخدمه البروتوكول الحالي؛ يجيب التحقق بـ `200` وتعيد الصفحة التحميل
400	طلب مشوه، أو حقل `Host` مكرر، أو هدف ليست له صيغة مُطبَّعة، أو بيانات تمرير وصفية غير صالحة، أو جسم مشوه، أو حل مرفوض مع تشخيص بأسلوب Elm
401	مطلوب تحدٍّ لعميل لا يقبل HTML، أو في وضع forward-auth
403	رفض من السياسة أو WAF، أو عنوان محظور، أو المصيدة
404	مسار غير معروف تحت `/__sibuna__/`، أو طلب مصيدة ليس طلبًا من المستوى الأعلى
413	جسم الحل أكبر من مخزن الاتصال المؤقت البالغ 64 KB
414	عنوان URL المُبلَّغ عنه للتحدي أطول من 8 KiB
417	توقُّع (Expect) غير مدعوم في الطلب؛ ويُعالَج `100-continue` محليًا
429	تجاوز حد GCRA؛ `Retry-After` بالثواني
431	رأس الطلب أكبر من 16 KB
501	`Transfer-Encoding` بترميز غير chunked
502	الخادم الأصلي غير قابل للوصول، أو كان رأس استجابته مشوهًا أو أكبر من 16 KB
503	بلوغ حد الاتصالات (`--max-connections` أو `--max-connections-per-ip`)، أو العقدة قيد التصريف؛ ويُغلق المقبس بعد الرد

فهرس الأخطاء

يوقف `INVALID COMMAND LINE` بدء التشغيل عند أي خيار لا يعرفه البرنامج الخدمي، وأي قيمة خارج مداها الموثق، وأي اختيار وضع غير صالح أو غائب أو مكرر. تذكر كتلة الخطأ الخيار، والقيمة المعطاة، والمدى المتوقع، والخطأ (`UnknownOption`، و`MissingValue`، و`InvalidValue`، و`InvalidMode`، و`MissingMode`، و`DuplicateMode`، و `RemovedOption` لخيار عنقود Zaxonlite، و`ConflictingOptions` لخياري تخزين يستبعد أحدهما الآخر، و `MissingOption` لعنوان URL لقاعدة بيانات بلا `--cluster-node`)، وتنتهي بتلميح. يُعرض عنوان URL لقاعدة البيانات بالصيغة `[redacted]`. حدّد `--mode reverse_proxy` أو `--mode forward_auth` مرة واحدة؛ والخيار القصير المكافئ هو `-m`. يطبع ملف السياسة المرفوض كتلة عنوانها اسم خطئه (مثل `INVALID POLICY ACTION`) مع الملف، واسم الخطأ، والقاعدة، والحقل، والقيمة. ونمط `ja4` الذي لا يمكن أن تطابقه أي بصمة يطبع

INVALID POLICY JA4 (InvalidJa4)؛ ويُرفض مستند إدارة وحدة التحكم الذي يحوي مثل هذا النمط بالخطأ نفسه. تُفحص خيارات وحدة التحكم على حدة: يسمي CONSOLE002 خيار وحدة تحكم مشوهًا أو مكررًا، ويسمي CONSOLE001 إعداد وحدة تحكم يخالف قاعدة (StorageRequired بلا تخزين).

> **التنفيذ: ErrorKind::explain** يربط كل خطأ من أخطاء المجال بسطر عنوان فاصل، وشرح، و:Hint. في crates/core/src/errors.rs (الحزمة sibuna-core).

الجدول 30 أخطاء التحدي والجلسة

الخطأ	السبب	تلميح
MalformedChallenge	المعرف ليس سجل تحدٍّ سليم البنية	اجلب تحديًا جديدًا وقدمه دون تغيير
InvalidChallengeTag	الوسم لا يجتاز المصادقة؛ لم يصدره هذا العنقود، أو عُدّل	لا يمكن تزوير التحديات؛ اطلب تحديًا جديدًا
ChallengeExpired	أقدم من TTL التحدي، أو يحمل وقت إصدار في المستقبل	اطلب تحديًا جديدًا
FingerprintMismatch	قُدم من عنوان أو User-Agent مختلف	قدمه من العميل الذي جلبه
DifficultyNotMet	لا تحقق قيمة Hashcash المؤقتة البتات الصفرية المطلوبة	واصل البحث عن قيم مؤقتة
InvalidProof	لا يفتح إثبات العمل المتسلسل التسميات المُلتزَم بها	شغّل المُبرهِن حتى النهاية بالعمق وعدد عمليات الفتح المحددين في التحدي
WrongSolutionType	قيمة مؤقتة لتحدي PoSW أو إثبات لتحدي Hashcash	طابق حقل الحل مع الخوارزمية
DoubleSpendAttempt	أُنفق التحدي من قبل	التحديات للاستخدام مرة واحدة
StoreFull	امتلأت مجموعة الحلول المُنفَقة	اخفض TTL التحدي أو ارفع --spent-tags
InvalidTokenSignature	فشل التحقق من وسم ملف تعريف الارتباط أو توقيعه	أعد المصادقة عبر الصفحة الوسيطة
TokenExpired	تجاوز ملف تعريف الارتباط مدة صلاحيته	أعد المصادقة

الخطأ	السبب	تلميح
TokenBoundAddressMismatch	قُدم ملف تعريف الارتباط من هوية عميل مختلفة	لا يمكن مشاركة ملفات تعريف الارتباط

تجيب واجهة API لوحدة التحكم عن الفشل بـ {"error":"<CODE>","hint":"<sentence>"}؛ والتلميح هو النص الذي تعرضه الواجهة (crates/console/src/http/reply.rs).

الجدول 31 رموز فشل وحدة التحكم التي تهم المشغلين

الرمز	الحالة	المعنى
CONSOLE002	400	الطلب مشوه، أو كبير جدًا، أو غير مسموح بصيغته
CONSOLE003	503 / 429	طابور مهام التخزين ممتلئ (32 مهمة)، أو متحقق كلمات المرور مشغول
CONSOLEQUORUM	503	لم تجب قاعدة البيانات خلال مدة انتظار الطلب البالغة 10 ثوانٍ؛ النتيجة غير معروفة، ويواصل مستوى البيانات العمل بآخر سياسة مطبقة
CONSOLE004	503	لا تستطيع الخدمة الإجابة: غير متاحة في هذا البناء، أو في طور التوقف، أو فشل داخلي
CONSOLENODE	400 / 409 / 503	رُفض أمر عقدة: يسمي عقدة أخرى (400)، أو مراجعة إقلاع أو تحكم قديمة (409)، أو وصل أثناء تنفيذ أمر آخر أو تعذر تسجيل اكتماله (503)؛ افحص إيصال العملية قبل إعادة المحاولة
CONSOLEMUTATION	429	أكثر من ستين تعديلًا إداريًا في دقيقة واحدة من جلسة واحدة

مخطط السياسة

```
{
  "default_action": "ALLOW" | "DENY" | "CHALLENGE",
  "waf": true | false,
  "inspection": { "path_traversal" | "sqli" | "xss" | "rce": "disabled" | "audit"
| "enforce" },
  "thresholds": { "challenge_at": int, "deny_at": int, "bits_step": int },
  "ip_rules": { "<cidr>": "ALLOW" | "DENY" | "CHALLENGE", ... },
  "rules": [                                                    // حتى 128
    {
      "name": "<kebab-case>",                                   // من 1 إلى 128
بايتًا قابلة للطباعة
      "path" | "path_regex": "<pattern>",                       // حتى 512 بايتًا
      "user_agent" | "user_agent_regex": "<pattern>",
```

```
      "headers" | "headers_regex": { "<Header>": "<pattern>" }, // حتى 4
      "remote_addresses" | "cidrs": ["<cidr>", ...],           // IPv4 حتى 8، من
IPv6 أو
      "action": "ALLOW" | "DENY" | "CHALLENGE" | "WEIGH",
      "weight": int,                                           // لـ WEIGH فقط،
-1000..1000
      "challenge": { "difficulty": <work bits>, "algorithm": "hashcash" |
"posw" },
      "limits": { "rate": int, "window_seconds": int, "ban_seconds": int },
      "ja4": "<fingerprint>" | "<prefix>*"                     // 36 بايتًا، أو
بادئة يتبعها *
    }
  ]
}
```

تقع challenge_at وdeny_at بين -1000 و1000، على أن تكون deny_at مساوية لـ challenge_at أو أعلى منها، وتقع bits_step بين 1 و1000. لا تنتمي كتلة challenge إلا إلى قواعد CHALLENGE، ببتات عمل من 1 إلى 64. تمنح limits القاعدة حصتها الخاصة: rate من 1 إلى 1,000,000 وwindow_seconds من 1 إلى 86,400 مطلوبتان، وban_seconds (86,400 على الأكثر) اختيارية، ولا يمكن لقواعد WEIGH حمل حدود. أما ja4 فهو بصمة JA4 كاملة يجب أن تساويها البصمة، أو بادئة يتبعها * يجب أن تبدأ بها البصمة؛ والقيمة * وحدها تطابق أي بصمة، والطلب الذي لا يحمل بصمة لا يطابق أي قاعدة ja4. ويُرفض النمط الذي لا يمكن أن تطابقه أي بصمة بالخطأ InvalidJa4 (الجزء التاسع، بصمات JA4 من وكيل الدخول). تحمل شجرة السمعة في ip_rules 8,192 عقدة على الأكثر. يمكن لكل قاعدة استخدام صيغة واحدة لكل معيار.

صياغة الأنماط: يطابق *. أو * أي شيء؛ ويقيّد $...^ المطابقة بمسار تام؛ وتجعل * أو */ أو *. في النهاية النمطَ بادئة؛ والنمط الذي يبدأ بـ / مسار تام؛ وأي شيء آخر سلسلة فرعية دون تمييز حالة الأحرف. تطابق أنماط المسار المسارَ في صيغته المُطبَّعة (شرطة مائلة واحدة بين المقاطع، بلا مقاطع نقطية، والمحارف غير المحجوزة غير مهرّبة، وسائر الترميزات المهرّبة بأرقام ست عشرية كبيرة)، لذا لا يطابق النمط الذي يحمل ; أو %2F أو %5C أو \ أي شيء، ومطابقة المسار حساسة لحالة الأحرف (SID 0003).

جداول التخزين

policy_revision، وpolicies (مع المعيار ja4_pattern القابل للقيمة NULL بطول 37 بايت على الأكثر)، وpolicy_inspection، وip_reputation، وincident_receipts، وsecurity_incidents (مع التضمين ذي 256 بايت وبصمة ja4 القابلة للقيمة NULL بطول 36 بايت)، وفهرس النص الكامل (incidents_fts، وهو FTS5 على الوحدات المعجمية المُطبَّعة في SQLite؛ والعمود search من نوع tsvector مع فهرس GIN في PostgreSQL)، وsibuna_schema؛ وتضيف وحدة التحكم 33 جدولًا من نوع *_console. تُخزن أسماء الإجراءات والخوارزميات والأوضاع بأحرف صغيرة. معرفات الحوادث هي sequence | node_id << 40، فريدة عبر العنقود دون تنسيق؛ وتمتد معرفات العقد من 1 إلى 8,388,607.

أهداف البناء

الجدول 32 أوامر البناء والمستودع

الأمر	النتيجة
`cargo build --release -p sibuna`	البرنامج الخدمي مع تخزين SQLite ووحدة التحكم (الميزات الافتراضية)
`cargo build --release -p sibuna --no-default-features`	برنامج خدمي في الذاكرة بلا تخزين ولا وحدة تحكم، وبلا أي حزمة خارجية
`cargo build --release -p sibuna --features postgres-tls`	برنامج خدمي يتصل أيضًا بقاعدة بيانات PostgreSQL مشتركة عبر TLS مُتحقَّق منه (الميزة `postgres` وحدها تتصل دون TLS)
`cargo test`	اختبارات الوحدات، واختبارات التكامل على برنامج خدمي قيد التشغيل، واختبارات التخزين ووحدة التحكم (واختبارات PostgreSQL عند ضبط `SIBUNA_TEST_DATABASE_URL`)
`cargo clippy --all-targets`، `cargo fmt --check`	فحوص lint والتنسيق (`max_width = 99`)
`cargo xtask check`	حدود الدالة ذات 70 سطرًا، والملف ذي 1,408 سطرًا، وطول السطر 99/108 عمودًا؛ وبلا محارف جدولة
`cargo xtask browser`	فحوص في Chromium بلا واجهة (`sibuna-browser`)
`cargo xtask bench <primitives|product|latency>`	قياسات أداء في وضع الإصدار (release)؛ والنتائج في `benchmarks/results/<command>-latest.json`
`cargo build -p sibuna-pow-wasm --target wasm32-unknown-unknown --profile wasm`	وحدة المتصفح فقط (`-p sibuna-console-ui` لواجهة وحدة التحكم)
`cargo run -p sibuna-geoip --bin geoip-snapshot -- ...`	تحقق من بيانات البلدان واكتب لقطة GeoIP قابلة للتضمين
`guidedog dsc` / `guidedog build`	هذا الكتاب / سجلات SID

يبني سكربت بناء البرنامج الخدمي (`cmd/sibuna/build.rs`) وحدتي WebAssembly كلتيهما بملف التعريف `wasm` في مجلد الأهداف لمساحة العمل، ويُفشل البناء عندما تتجاوز واجهة وحدة التحكم 768 KiB. يسمي `SIBUNA_POW_WASM` و`SIBUNA_CONSOLE_WASM` وحدات مبنية مسبقًا بدلًا من ذلك، للبناءات التي لا يتوفر فيها الهدف

`wasm32-unknown-unknown`. يسمي `SIBUNA_GEOIP_DATA` لقطة GeoIP للتضمين؛ ويُفك ترميزها وقت البناء، ويرفضها البناء الذي لا يضم وحدة التحكم (`GEOIP001`).

تمرين 10.1. يتلقى عميل الحالة `400` وعنوانها `CLIENT FINGERPRINT MISMATCH` بعد انتقاله من Wi-Fi إلى الشبكة الخلوية في منتصف الحل. اشرح السبب انطلاقًا من تنسيقي الرمز والتحدي، واقترح أصغر تغيير في الصفحة الوسيطة يسمح بالتعافي السلس.

اشرح اللامتغير. دون النظر، اذكر نقاط النهاية التي يجب أن يوجهها وكيل عكسي أمام Sibuna إلى البرنامج الخدمي بدل الخادم الأصلي، واذكر سبب الحاجة إلى كل منها.

الجزء الحادي عشر

بطاقة مرجع سريع

أبقِ هذه البطاقة بجانب الطرفية: البناء، والتشغيل، والإعداد، والتوجيه، والمراقبة، والتشخيص. يضم الجزء العاشر الجداول الكاملة، وتضم هذه البطاقة ما يكتبه المشغل.

البناء

```
cargo build --release -p sibuna                          # البرنامج الخدمي + SQLite + وحدة التحكم
cargo build --release -p sibuna --no-default-features    # في الذاكرة، بلا أي حزمة خارجية
cargo build --release -p sibuna --features postgres-tls  # PostgreSQL عناقيد المشتركة
rustup target add wasm32-unknown-unknown   # أو اضبط SIBUNA_POW_WASM / SIBUNA_CONSOLE_WASM
cargo test            # اختبارات الوحدات والبرنامج الخدمي الحي والتخزين ووحدة التحكم
cargo clippy --all-targets && cargo fmt --check
cargo xtask check     # فحص حدود 70 سطرًا / 1,408 سطرًا / 99 عمودًا
cargo xtask browser   # فحوص Chromium بلا واجهة
cargo xtask bench product   # /wrk -> benchmarks/results تحت المنتجات كاملة
guidedog build        # هذا الكتاب (SID سجلات :guidedog dsc)
```

التشغيل

```
# Shield (الافتراضي): PoW + WAF، مع التمرير إلى المنفذ 3000
sibuna -p 8080 -u 3000 -s /etc/sibuna/secret
# Gate: إثبات العمل فقط
sibuna --gate -p 8080 -u 3000 -s /etc/sibuna/secret
# forward auth خلف Nginx/Caddy (يثق بـ X-Forwarded-For)
sibuna -m forward_auth --host 127.0.0.1 -p 8080 -s /etc/sibuna/secret
# Edge: سياسات دائمة وسمعة وأدلة جنائية في <dir>/sibuna.db
sibuna -D /var/lib/sibuna -s /etc/sibuna/secret
# JA4 من وكيل دخول موثوق (ترويسة Cloudflare؛ وحدة x-ja4؛ nginx: ja4; ATS: ja4)
sibuna --ja4-header cf-ja4 --trust-forwarded -p 8080 -u 3000 -s /etc/sibuna/secret
# عضو في عنقود: PostgreSQL مشتركة ومعرّف عقدة فريد
sibuna --database-url-file /etc/sibuna/database-url --cluster-node 1 -s /etc/sibuna/secret
head -c 32 /dev/urandom | xxd -p -c 64 > /etc/sibuna/secret
```

وحدة التحكم (معاينة تُفعَّل اختياريًا)

```
# هيّئ والبرنامج الخدمي متوقف، ثم شغّل وحدة التحكم
sibuna init-admin admin --data-dir ./data
sibuna --data-dir ./data --console 127.0.0.1:19446
```

افتح http://127.0.0.1:19446/console/ واستبدل كلمة المرور المؤقتة. أضف --console-location 1.3521,103.8198 لوضع العقدة على الكرة الأرضية. يتطلب الوصول البعيد وكيل HTTPS، ومصدرًا (origin) صريحًا، ونطاقات CIDR للوكلاء الموثوقين؛ اتبع الجزء التاسع. حالة SID 0007 هي Committed. تجتاز وحدة التحكم مسارات عملها في المتصفح على Chromium بلا واجهة (SID 0010، المعيار 5)؛ أما شرطها الرسمي لعزل الأداء فلم يُحسم بعد (الجزء الثامن).

الخيارات الأهم أولًا

الخيار	الافتراضي	المعنى
`-d, --difficulty`	16	بتات العمل (عمق PoSW = البتات - 3)
`-a, --algorithm`	posw	posw أو hashcash
`--posw-challenges`	16	عدد عمليات الفتح لكل إثبات
`--token-ttl`	86400	مدة الجلسة بالثواني
`--challenge-ttl`	300	مدة التحدي بالثواني
`--token-scheme`	mac	mac أو ed25519
`--rate-limit / --rate-window`	10 / 100	دفعة GCRA لكل نافذة
`--ban-seconds`	3600	حظر المصيدة
`-w, --workers`	عدد وحدات CPU	خيوط قبول الاتصالات
`--max-connections`	1024	ثم 503
`--max-connections-per-ip`	64	لكل عنوان نظير (في IPv6: لكل --ipv6-prefix)، ثم 503
`--trusted-hops`	1	العميل = العنصر رقم n في X-Forwarded-For من اليمين

الخيار	الافتراضي	المعنى
`--ja4-header`	معطل	ترويسة تحمل بصمة JA4 من وكيل الدخول؛ ولا تُقرأ إلا عند الوثوق بالترويسات الممررة
`--idle-timeout`	15	مهلة جامع الاتصالات الخاملة بالثواني
`-P, --policy-file`	لا شيء	سياسة JSON
`--trust-forwarded`	معطل	مفعّل في وضع forward auth
`--secure-cookie`	معطل	يضيف `Secure`

نقاط النهاية تحت `/__sibuna/`

المسار	الرد
`challenge`	الصفحة الوسيطة بصيغة HTML
`challenge.json?need=&path=`	`{id, algorithm, difficulty, challenges, expires_at}`
`verify` (POST)	`{challenge_id, nonce | proof}` -> 200 + `Set-Cookie`، وإلا 400
`wasm/sibuna-pow.wasm`	برنامج حل بحجم 15,181 بايت
`worker.js`	عامل يضم مُبرهِنَي WASM وJS
`honeypot`	يحظر المستدعي، 403؛ و404 لمورد فرعي أو جلب عبر المواقع
`health`	`{"status":"ok",...}`
`metrics`	نص Prometheus

وجّه هذه المسارات كلها إلى البرنامج الخدمي، ولا توجهها أبدًا إلى الخادم الأصلي.

رموز الحالة

الرمز	متى
200	مقبول (forward auth)، والصفحة الوسيطة، والمسارات الداخلية
400	طلب أو هدف (target) مشوّه، أو حل مرفوض (مع جسم تشخيصي)

الرمز	متى
401	مطلوب تحدٍّ: عميل لا يقبل HTML، أو وضع forward auth
403	رفض من السياسة أو WAF، أو حظر، أو المصيدة
404	مسار /sibuna__/ غير معروف، أو طلب مصيدة مرفوض
413	جسم الحل أكبر من المخزن المؤقت البالغ 64 KB
414	عنوان URL المُبلَّغ عنه للتحدي أطول من 8 KiB
417	توقُّع (Expect) غير مدعوم في الطلب
429	حد GCRA، مع Retry-After بالثواني
431	رأس الطلب أكبر من 16 KB
501	ترميز نقل غير chunked
502	الخادم الأصلي غير قابل للوصول أو استجابته مشوهة
503	بلوغ حدود الاتصالات، أو العقدة قيد التصريف

الترويسات

- نحو الخادم الأصلي *(الوكيل)*: X-Forwarded-For (نظير المقبس، بعد السلسلة المستلمة عندما تكون موثوقة)، وX-Forwarded-Proto، وX-Real-IP (العميل الذي بُني عليه القرار)، وX-Sibuna-Status: PASS، وX-Sibuna-Rule: <rule>؛ وتُحذف ترويسات hop-by-hop وحقول التمرير البديلة، وكذلك صيغها المكتوبة بشرطة سفلية.
- رد *forward auth*: X-Sibuna-Status، وX-Sibuna-Rule، وX-Sibuna-Rule-Hash.
- رد التحدي: X-Sibuna-Status: CHALLENGE.
- ملف تعريف الارتباط:

```
__sibuna_token=<75 chars, 139 with ed25519>; Path=/; Max-Age=<ttl>; HttpOnly;
SameSite=Lax[; Secure]
```

هيكل ملف السياسة

```
{ "default_action": "CHALLENGE", "waf": true,
  "thresholds": {"challenge_at": 10, "deny_at": 40, "bits_step": 5},
  "ip_rules": {"10.0.0.0/8": "ALLOW", "2001:db8::/32": "DENY"},
  "rules": [
```

```
    {"name": "api", "path": "/api/*", "headers": {"X-Api-Key": ".*"}, "action":
"ALLOW"},
    {"name": "checkout", "path": "/checkout/*", "action": "CHALLENGE",
     "challenge": {"difficulty": 20, "algorithm": "posw"}},
    {"name": "login", "path": "^/login$", "action": "CHALLENGE",
     "limits": {"rate": 10, "window_seconds": 60, "ban_seconds": 600}},
    {"name": "old-tls", "ja4": "t12*", "action": "CHALLENGE"},
    {"name": "headless", "user_agent": "Headless", "action": "WEIGH", "weight":
30} ] }
```

JA4: بصمة كاملة، أو بادئة مع *؛ لا بصمة، لا مطابقة. الأنماط: * أو *. لأي قيمة؛ $...^ للتطابق التام؛ * في النهاية لبادئة؛ / في البداية لمسار تام؛ وإلا فسلسلة فرعية دون تمييز حالة الأحرف. تُطابق المسارات في صيغتها المُطبَّعة. الترتيب: WAF -> رفض/سماح الشجرة -> القواعد -> الدرجة -> التجاوز -> تحدي الشجرة -> الروبوتات -> الافتراضي.

التخزين (Edge) بلغة SQL

```
INSERT INTO policies (id, name, priority, path_pattern, action, difficulty,
algorithm,
  header_matchers, cidr_matchers, weight, enabled, created_at, updated_at)
  VALUES ('p1', 'protect', 10, '/checkout/*', 'challenge', 20, 'posw', '{}', '[]',
0, 1,
  unixepoch(), unixepoch());
INSERT INTO ip_reputation (ip_or_cidr, reputation_score, banned_until,
trigger_rule, hits, last_seen)
  VALUES ('198.51.100.7', -100, unixepoch() + 86400, 'analyst', 1, unixepoch());
SELECT s.client_ip, s.violation_category, s.path, s.campaign_id
  FROM incidents_fts f JOIN security_incidents s ON s.id = f.rowid
  WHERE incidents_fts MATCH 'union' ORDER BY rank LIMIT 20;
```

SQLite (sqlite3 <dir>/sibuna.db)؛ تُكتب الإجراءات بأحرف صغيرة. الدرجة $-50 \le$ -> بادئة رفض؛ و$50 \ge$ -> سماح. العنقود: السر نفسه على كل عقدة، وقاعدة بيانات PostgreSQL واحدة، وpostgres-tls --features، و--cluster-node N --database-url-file <file> مع قيمة N فريدة من 1 إلى 8,388,607؛ بلا قائمة أعضاء، ولا قائد، ولا TLS للعنقود (SID 0011).

التشخيصات

العنوان	المعنى
MALFORMED CHALLENGE	عُدّل المعرف أو اقتُطع؛ اجلب معرفًا جديدًا
INVALID CHALLENGE TAG	لم يصدر عن هذه البذرة أو هذه العقدة
CHALLENGE EXPIRED	أقدم من TTL أو انحراف الساعة > 60 s

العنوان	المعنى
CLIENT FINGERPRINT MISMATCH	تغير العنوان أو User-Agent
DIFFICULTY NOT MET	لا تحقق القيمة المؤقتة عدد البتات الصفرية المطلوب
INVALID PROOF OF SEQUENTIAL WORK	لا تصل عمليات الفتح إلى الجذر
WRONG SOLUTION TYPE	قيمة مؤقتة لـ PoSW أو إثبات لـ Hashcash
DOUBLE SPEND PREVENTED	استُخدم التحدي من قبل
STORE FULL	امتلأت مجموعة الحلول المُنفَقة؛ اخفض TTL أو ارفع `--spent-tags`
NETWORK BINDING MISMATCH	نُسخ ملف تعريف الارتباط إلى عميل آخر
INVALID COMMAND LINE	خيار غير معروف، أو قيمة خاطئة، أو خيار من خيارات Zaxonlite المُزالة

المقاييس التي ينبغي التنبيه عليها: `denied`، و`rate_limited`، و`overloaded`، و`upstream_errors`، و`incidents_dropped`، و`incident_write_failures`، و`bans_refused`، و`forwarded_invalid` (كلها بصيغة `sibuna_*_total`).

أرقام جديرة بالحفظ

معرف التحدي 70 محرفًا (حمولة 36 بايت + وسم 16 بايت)؛ الرمز 75 محرفًا (40 + 16)، و139 مع Ed25519؛ بايتات الإثبات $32(1 + t(n + 1))$؛ عمق PoSW = البتات - 3، ضمن المدى 4 إلى 24؛ مخزن الطلب المؤقت 64 KB؛ حد الرأس 16 KB؛ الجزء المفحوص من الجسم 8 KB؛ مجموعة الحلول المُنفَقة 1,048,576 وسمًا (`--spent-tags`)؛ خلايا المعدل 262,144 لكل جدول (`--rate-table-cells`)؛ الحظر 65,536 موضعًا (`--ban-table-slots`)؛ 64 اتصالًا لكل عنوان عميل؛ حلقة الحوادث 512؛ الدفعة 32؛ مسافة الحملة 0.35 على أحدث 16,384 تضمينًا؛ keep-alive حتى 4,096 طلبًا لكل اتصال؛ مجمّع اتصالات الخادم الأصلي 256 مقبسًا؛ طابور مهام التخزين 32.

توجيه وكيل الدخول

استخدم وصفة Nginx أو Caddy في الجزء التاسع، Forward Auth خلف وكيل دخول. وجّه `/__sibuna/*` مباشرة؛ وقرّر الإذن لطلبات التطبيق بناءً على عنوان URI الأصلي، والطريقة، وعنوان العميل الموثوق. استبدل ترويسات تدقيق Sibuna الواردة بنتيجة التفويض. يحتاج Nginx إلى معالجة صريحة للتحدي، وللحالة 429، ولتعذر خدمة المصادقة.

ينقل `reverse_proxy` أجسام HTTP/1.1 المقبولة واتصالات WebSocket؛ أما `forward_auth` فيمنح القبول وينقلها وكيل الدخول. ولا يستطيع Sibuna فحص الأجسام التي لا تصله في طلب المصادقة، ولا استجابات الخادم

الأصلي، ولا عدّها. ينتهي TLS عند وكيل الدخول؛ ودعم HTTP/2 الأصلي مؤجل. تستخدم عمليات الرفع Content-Length أو الترميز المجزأ (chunked)؛ ولا يرى الفحص إلا بادئة 8 KiB على الأكثر، ولا تُفسَّر بايتات الملفات.

حلول مختارة

هذه الحلول وسائل لفحص الاستدلال، وليست بديلًا عن التمارين. وإذا اختلفت نتيجة ما، نقارن الافتراضات أولًا: فنهايات الفترات الزمنية، وما يُحسب عملًا، وهل أُودعت عملية الكتابة أم لا، كلها مصادر شائعة للاختلاف.

الجزء الأول: التكلفة والاحتمال

1.1. مليون محاولة مقسومة على 100 طلب تساوي 10,000 محاولة لكل طلب. ولا تغير مشاركة الجلسة هذا الحساب إذا بقي العدد الكلي للطلبات 100. أما النظام الفعلي فعليه أن يقرر هل تستطيع عملية ثانية تحقيق روابط البصمة والتوجيه؛ وهذه الشروط خارج معادلة توزيع التكلفة.

1.2. الفائض $1000 - 600 = 400$ سجل في الثانية. ويشتري طابور فارغ يتسع لـ 512 سجلًا $512/400 = 1.28$ ثانية في نموذج التدفق المستمر. أما المنتجون الحقيقيون فيصلون في دفعات، والمستهلك يودع دفعات مجمعة، لذا يعتمد موعد أول إسقاط بالضبط على أوقات الوصول والإيداع.

الجزء الثاني: الأعمال السابقة

2.1. مصدر واحد يُترجم مرتين لا يمكن أن يختلف مع نفسه في نص المسألة، أو ترميز القيمة المؤقتة، أو شكل الشجرة، أو تخطيط عمليات الفتح؛ فالانحراف يتطلب علة في المترجم. لكنه لا يقول شيئًا عن سلامة البناء (وهذه مهمة المراجعة التشفيرية)، ولا عن صحة تنفيذه في بيئة تشغيل المتصفح (وتغطي ذلك مقارنة متطابقة بايتًا ببايت بين مُبرهِني WebAssembly وJavaScript)، ولا عن معايرة المعاملات (وهذه مهمة القياس).

2.2. (أ) Sibuna Gate أو Anubis، ملف تنفيذي واحد؛ (ب) Sibuna Shield أو SafeLine أو Cloudflare (خطة Pro أو أعلى)؛ (ج) Sibuna Edge مع قاعدة بيانات PostgreSQL مشتركة (`--database-url`)، أو Anubis مع مخزن Valkey مشترك (للحالة، لا للحظر)، أو Cloudflare. والمنتج الوحيد الذي يحقق الشروط الثلاثة دون مضيف Docker أو اشتراك هو Sibuna Shield مع عقد تتشارك قاعدة بيانات PostgreSQL واحدة (`--database-url` و`--cluster-node`)؛ والصفوف "طريقة التشغيل" و"الفحص" و"العقد المتعددة" هي التي تحسم ذلك.

الجزء الثالث: التشفير

3.0. تقدير الذيل $e^{-4.6} \approx 0.010$. تقسيم فضاء القيم المؤقتة يغير المعدل الذي تكتمل به المحاولات، لا احتمال نجاح كل محاولة مستقلة. وقد تجعل المحاولات المكررة أو عمل التنسيق التنفيذ المتوازي الحقيقي أقل كفاءة.

3.1. إذا نجح كل التزام باحتمال q، تفشل الالتزامات g كلها باحتمال $(1-q)^g$. وينجح واحد منها على الأقل باحتمال $1-(1-q)^g$. لا تحسب الصيغة تكلفة بناء الالتزامات، ولا ميزانيات الاستعلام، ولا الاعتماديات بين المحاولات، ولا العمل اللازم لإيجاد التزام باحتمال قبول معين.

3.2. عند العمق 17 و16 عملية فتح، يبلغ حجم الإثبات $32(1 + 16 \times 18) = 9248$ بايت. وعند 32 عملية فتح يبلغ $32(1 + 32 \times 18) = 18464$ بايت. تتضاعف بايتات عمليات الفتح؛ أما الجذر ذو 32 بايت فيُرسل مرة واحدة كما كان.

3.3. من دون مستوى العمل، كانت الجلسة المكتسبة على مسار رخيص ستقبل طلبًا إلى مسار مكلف يتطلب عملًا أكثر؛ ومن دون `timestamp` كان يمكن سك رمز "في المستقبل" فيعيش بعد سياسته؛ ومن دون `expiry` لن ينتهي أبدًا؛ ومن دون `rule_hash` لن يعرف الخادم الأصلي أي قاعدة قبلت العميل؛ ومن دون `fingerprint` سيُقبل ملف تعريف ارتباط منسوخ إلى جهاز آخر. ويجب أن يغطي الوسم الحقول الخمسة كلها، لأن أي حقل يبقى خارجه يمكن تعديله بحرية دون أن يلاحظ المتحقق.

الجزء الرابع: البروتوكول

4.1. يتشارك المتصفحان العنوان وUser-Agent، فيتشاركان البصمة، وخلية حد المعدل، وموضع الحظر. ولا يتشاركان التحدي (إذ يجلب كل منهما سجله الخاص بقيمته المؤقتة الخاصة)، ولا مُدخَل مجموعة الحلول المُنفَقة، ولا ملف تعريف الارتباط. ولكي يعيد مهاجم خلف NAT نفسه استخدام ملف تعريف الارتباط الخاص بالمتصفح الآخر، عليه أن يقرأه من الجهاز الآخر: عندئذ ستتطابق البصمة، ولهذا يهم `HttpOnly` و`SameSite` أكثر من الربط عند العنوان المشترك.

الجزء الخامس: الحالة والترتيب

5.1. الرأس والجزء الأول من الجسم (حتى ما يتسع بعد الرأس في 64 KB) موجودان في مخزن الاتصال المؤقت ويُرسلان بـ `write_all(request.body)`. أما البايتات المتبقية فلا تدخل المخزن المؤقت كاملة أبدًا: يقرؤها `relay_body` في قطع من 16 KB من قارئ العميل ويكتب كلًّا منها إلى الخادم الأصلي. ويجب أن يُرسل الرأس أولًا لأن الخادم الأصلي يحتاج إلى `Content-Length` قبل الجسم، ولأن ترويسات التدقيق جزء من الرأس.

5.2. فاصل الإصدار 100 ms، والتسامح مع الدفعات 9,900 ms. يستطيع عميل خامل أن يرسل 100 طلب فورًا، ثم طلبًا واحدًا كل 100 ms. وحتى نقطة النهاية الشاملة عند 3,000 ms، يكون الحد 130. أما العداد الثابت لمدة 10 ثوانٍ فسيسمح بـ 100 طلب في النافذة الأولى و100 أخرى لحظة انتقال النافذة؛ والسجل المنزلق سيسمح بـ 100 طلب بالضبط في أي 10 ثوانٍ، لكن عليه تخزين 100 طابع زمني لكل عميل.

الجزء السادس: الخوارزميات

6.1. الفاصلة العليا في اسم ما، ومعامل عادي اسمه `order`، لا يشكلان تعبير SQL. وقد يخطئ كاشف السلاسل الفرعية فيعد أيًّا منهما صياغة. أما الكاشف البنيوي فيبحث عن أدلة إضافية، منها العوامل والعلاقات بين القيم الحرفية. ويبقى فحصًا محدودًا، لا محللًا لكل لغة تطبيقات.

6.2. تُبنى مرشحات LDAP من الأقواس و| و& و! و=. يحتاج الكاشف إلى فئة بايت لـ) و((جديدة)، ويستطيع إعادة استخدام الفئتين الموجودتين لـ = ولفواصل أوامر الصدفة من أجل | و&. وينبغي أن ينطلق عند تتابع مثل)(أو)|)، لا عند قوس واحد، للسبب نفسه الذي يجعل كاشف SQL يشترط علامة اقتباس أو تحصيل حاصل.

الجزء السابع: محرك المتصفح

7.1. عند 16 بت عمل الافتراضية (العمق 13 مع 16 عملية فتح)، تصبح الـ 20 ms المفترضة في WebAssembly $20 \times 50 = 1000$ ms، أي ثانية واحدة، في JavaScript على الحاسوب المحمول، وأربع ثوانٍ على الهاتف الأبطأ. هذه الأرقام مفترضة؛ ونقيس أجهزة عملائنا قبل أن نضع سياسة. والسياسة المعقولة: أن ندع الصفحة تمرر الرسالة `fallback`، ونسجلها في الخادم مقياسًا، ونخفض الصعوبة لقاعدة تطابق العملاء المعروفين بحجب WebAssembly، لا للجميع.

الجزء الثامن: التقييم

8.2. في صف `admitted` الخاص بـ Gate، يكون $11.84 \times 211,345/10^6 \approx 2.50$ نواة مشغولة من أصل أربع، فلم يكن البرنامج الخدمي محدودًا بوحدة CPU، بل كان مولد الحمل هو ما وضع الحد. ووقت CPU البالغ 11.84 us لكل طلب أعلى بكثير من تكلفة التصنيف التي تقل عن ميكروثانية، لأن صف المنتج يتضمن أيضًا قراءة النواة من المقبس والكتابة إليه، والمحلل، وفحص وسم الجلسة، وزيادات المقاييس، وتنسيق الاستجابة، وتفريغ الكاتب، وتسليم المجدول بين مولد الحمل وخيط الاتصال. ولا تقيس مجموعة اختبارات الأدوات الأولية أيًّا من ذلك.

الجزء التاسع: التشغيل

9.1. إذا أُودعت البيانات قبل تحديث منفصل للإيصال، فإن انهيارًا بينهما يترك الإيصال القديم، وقد تطبق إعادة المحاولة الأثر مرة أخرى. وإذا أُودع الإيصال أولًا، فقد يجعل انهيار قبل كتابة البيانات إعادة المحاولة تتخطى بيانات لم توجد قط. والمعاملة الذرية الواحدة تزيل الفجوتين كلتيهما. أما فقدان إقرار المعاملة فلا يزال يتطلب إعادة المحاولة المحتفظ بها.

9.2. `ip_rules: {"10.20.0.0/16": "ALLOW"}` (أو صف `ip_reputation` بدرجة 100) يقبل بيئة الاختبار؛ والقاعدة `{"name": "admin", "path": "/admin/*", "action": "CHALLENGE", "challenge": {"difficulty": 20, "algorithm": "posw"}}` تطلب العمل؛ والقاعدة التي تطابق `headers: {"X-Partner-Key": ".*"}` مع `ALLOW` و `"limits": {"rate": 30, "window_seconds": 10}` تقبل الشريك بمعدل 30 طلبًا لكل 10 ثوانٍ. وتُفحص حصة القاعدة لكل طلب تطابقه، بعد حد البرنامج الخدمي لكل عميل (`--rate-limit` و`--rate-window`)، الذي يجب لذلك أن يكون سخيًّا بالقدر نفسه على الأقل.

الجزء العاشر: المرجع

10.1. يحمل سجل التحدي البصمة ذات المفتاح للعنوان الذي جلبه؛ والحل يصل من عنوان جديد، لذا يعيد `verify_and_mint` الخطأ `FingerprintMismatch` قبل أن ينظر في الإثبات. وأصغر إصلاح يكون في الصفحة الوسيطة: عند استجابة `400` عنوانها `CLIENT FINGERPRINT MISMATCH`، تجلب الصفحة تحديًا جديدًا وتحله مرة أخرى بدل عرض الخطأ، لأن العمل المنجز لا يمكن نقله.

واصل البحث

نغيّر لامتغيرًا واحدًا في كل مرة في اختبار مؤقت. نزيل إعادة فحص اللقطة، أو نحذف إيصال الحادث، أو نقرّب فاصل المعدل إلى الأسفل. ونتوقع جدولًا زمنيًا للفشل قبل أن نشغل الاختبار. والاختبار المفيد لكشف التراجع يفعل أكثر من التحقق من مخرجات المسار السعيد: إنه يبين لماذا كان الشرط المحذوف ضروريًا.

مسرد المصطلحات

القبول (Admission)

قرار تمرير طلب إلى الخادم الأصلي. في Sibuna، يصدر هذا القرار عن جلسة صالحة، أو قاعدة `ALLOW`، أو مسار تجاوز ثابت، أو بادئة سمعة `allow`، ولا يتجاوز أبدًا رفضًا صادرًا عن WAF.

آلة Aho-Corasick

آلة حالات منتهية تجد أي نمط من مجموعة ثابتة من الأنماط بمرور واحد على المدخل. يدمج Sibuna روابط الفشل في جدول انتقال كثيف، فيكلف كل بايت تحميلًا واحدًا من الجدول.

جدول الحظر

4,096 موضعًا بلا أقفال لعناوين العملاء المحظورين مع وقت انتهاء الحظر. تكتب فيه المصيدةُ والسمعةُ المشتركة عبر العنقود، ويُقرأ في كل طلب.

الحملة

مجموعة من الحوادث الأمنية تقع تضمينات حمولاتها ضمن مسافة جيب التمام 0.35 بعضها من بعض؛ وهي تجميع استدلالي، لا برهان على مصدر مشترك.

سجل التحدي

معرّف عديم الحالة من 70 محرفًا يجب أن يحله العميل: الإصدار، والخوارزمية، والصعوبة، وعدد عمليات الفتح، ووقت الإصدار، وبصمة العميل، وقيمة PRF مؤقتة، وتجزئة القاعدة، ووسم BLAKE3 ذو مفتاح من 16 بايت. لا يكتب إصداره شيئًا على الخادم.

الصعوبة، بتات العمل

العدد الصحيح الواحد الذي يضبط المستويين كليهما: عدد البتات الصفرية البادئة في Hashcash، أو عمق الشجرة مضافًا إليه ثلاثة في العمل المتسلسل.

Edge

نشر لـ Sibuna مع `--data-dir` (ملف SQLite على عقدة واحدة)، أو مع `--database-url` و `--cluster-node` (قاعدة بيانات PostgreSQL واحدة يتشاركها العنقود)، يمنح أيًّا من الوضعين سياسات دائمة، وسمعة مشتركة، وأدلة جنائية عن الحوادث.

موضع المحرك

المكان الذي يقيم فيه محرك السياسة المنشور، على هيئة مؤشر بعدّ المراجع. يمسك العمال مرجعًا أثناء التصنيف؛ ويبني خيط التخزين محركًا جديدًا خارج مسار الطلب وينشره بتبديل المؤشر، ويُحرر المحرك القديم عندما يتخلى عنه آخر قارئ له.

البصمة

تجزئة ذات مفتاح لعنوان العميل وUser-Agent، تُربط في التحديات والرموز كي لا يمكن إعادة إرسال أي منهما من عميل آخر.

Forward auth

وضع نشر يسأل فيه وكيل الدخول Sibuna هل يقبل الطلب (200) أو يتحداه أو يرفضه أو يحد معدله (401 و403 و429)، ويمرر الطلب إلى الخادم الأصلي بنفسه.

Gate

الوضع الذي يقدم القبول بإثبات العمل، والجلسات، والقواعد، والسمعة، وحدود GCRA، والحظر، دون فحص للتطبيق (--gate).

GCRA

خوارزمية Generic Cell Rate Algorithm: وقت وصول نظري واحد لكل عميل، يسمح بدفعة من N طلبًا ثم بطلب واحد لكل فاصل إصدار، دون حدود نوافذ.

Hashcash

المستوى الأول: أوجد قيمة مؤقتة (nonce) تبدأ معها قيمة SHA-256 للتحدي ونقطتين رأسيتين والقيمة المؤقتة العشرية بـ b بتًا صفريًا. العدد المتوقع للمحاولات 2^b، والتحقق بتجزئة واحدة.

المصيدة

رابط غير مرئي إلى /sibuna/honeypot__؛ يُحظر العميل الذي يتبعه، ويُسجل حادث بسمعة 100−.

جامع الاتصالات الخاملة

خيط يغلق الاتصالات التي بقيت صامتة أطول من --idle-timeout، كي لا يستطيع عميل بطيء أن يحتجز خيط اتصال.

الصفحة الوسيطة

صفحة HTML مدمجة تُقدَّم بدل الصفحة المحمية؛ تجلب تحديًا، وتحله في Web Worker، وترسل الحل، ثم تعيد التحميل.

JA4

بصمة من 36 محرفًا، عرّفتها FoxIO، للطريقة التي يفتح بها العميل اتصال TLS: البروتوكول، وإصدار TLS، واسم الخادم، وعدد مجموعات التشفير وعدد الامتدادات، وALPN، وتجزئات مجموعات التشفير والامتدادات بعد ترتيبها. لا يحسبها Sibuna؛ بل يقرأ القيمة التي يمررها وكيل دخول موثوق في الترويسة التي يسميها --ja4-header، وتطابقها القواعد بالحقل ja4 (SID 0014). ولا صلة لها ببصمة العميل المربوطة في التحديات والرموز.

BLAKE3 ذو مفتاح

الدالة شبه العشوائية التي يقوم عليها جدول مفاتيح Sibuna، ووسوم التحديات، ووسوم الجلسات، والبصمات، والقيم المؤقتة للتحديات.

رمز MAC

ملف تعريف الارتباط الافتراضي للجلسة: حمولة من 40 بايت (الإصدار، ومستوى العمل، والطابع الزمني، ووقت الانتهاء، وتجزئة القاعدة، والبصمة) ووسم BLAKE3 ذو مفتاح من 16 بايت، يُتحقق منه في زمن ثابت.

مستوى العمل

الآلية وبتات العمل التي حلها حامل الجلسة فعلًا، ويحملها الرمز؛ ولا يقبل المسار جلسة إلا إذا بلغ مستواها متطلب المسار.

الفتح (Opening)

في إثبات العمل المتسلسل، تسمية ورقة واحدة مع تسميات أشقاء العقد على مسارها إلى الجذر؛ يعيد المتحقق حساب المسار ويقارنه بالالتزام.

إثبات العمل المتسلسل (PoSW)

المستوى الثاني: مخطط التجزئة (hash graph) لـ Cohen-Pietrzak الذي تعتمد تسمياته على كل التسميات السابقة، فيفرض $2^{n+1} - 1$ تجزئة متسلسلة مهما بلغ عدد أنوية المُبرهِن. يُتحقق منه بـ $t(n+1)$ تجزئة.

شجرة السمعة

شجرة radix من 128 بت تحمل بادئات `allow` و`deny` و`challenge` من ملف السياسة ومن جدول `ip_reputation`.

مجموعة Robin Hood للحلول المُنفَقة

جدول ثابت السعة بعنونة مفتوحة لوسوم التحديات المحلولة، يمنع تقديم حل صالح مرتين.

تجزئة القاعدة

تجزئة من 64 بت لاسم القاعدة التي طلبت تحديًا، يحملها التحدي والرمز وترويسة `X-Sibuna-Rule-Hash`.

Shield

الوضع الافتراضي: Gate مضافًا إليه الجدار الناري الدلالي (`--shield`).

SID

سجل Sibuna Discussion: وثيقة تصميم بأسلوب الورقة البحثية تحت `docs/sid/records`، تضم الافتراضات والبراهين والقياسات التي يقوم عليها جزء واحد من النظام.

الوضع (Surface)

أحد شكلي المنتج، Gate أو Shield؛ والتخزين والعنقدة خياران لكل منهما.

WEIGH

إجراء قاعدة يضيف وزنًا ذا إشارة إلى درجة الطلب بدل أن يتخذ قرارًا؛ والمجاميع التي تبلغ العتبات أو تتجاوزها تؤدي إلى تحدٍّ ببتات عمل إضافية أو إلى الرفض.

Zaxonlite

محرك تكرار SQLite المدمج (Multi-Paxos) في SID 0005، الذي حل محله التخزين في SQLite وPostgreSQL في SID 0011؛ وخيارات العنقود الخاصة به توقف الآن بدء التشغيل بالخطأ `RemovedOption`.

المراجع

- Aho, A. V., and Corasick, M. J. “Efficient string matching: an aid to bibliographic search.” *Communications of the ACM* 18(6), 1975.
- ATM Forum. *Traffic Management Specification Version 4.0*, af-tm-0056.000, 1996 (خوارزمية Generic Cell Rate Algorithm).
- Back, A. "Hashcash: a denial of service counter-measure." تقرير تقني، 2002.
- Blocki, J., Lee, S., and Zhou, S. “On the security of proofs of sequential work in a post-quantum world.” *Information-Theoretic Cryptography (ITC)*, 2021.
- Celis, P. *Robin Hood Hashing*. أطروحة دكتوراه، University of Waterloo، 1986.
- Cohen, B., and Pietrzak, K. “Simple proofs of sequential work.” *EUROCRYPT*, 2018.
- Dwork, C., and Naor, M. “Pricing via processing or combatting junk mail.” *CRYPTO*, 1992.
- Fiat, A., and Shamir, A. “How to prove yourself: practical solutions to identification and signature problems.” *CRYPTO*, 1986.
- Fielding, R., Nottingham, M., and Reschke, J. *HTTP Semantics*, RFC 9110, and *HTTP/1.1*, RFC 9112. IETF, 2022.
- FoxIO. *JA4+ Network Fingerprinting*. https://github.com/FoxIO-LLC/ja4، 2023 (بصمة عميل TLS من نوع JA4).
- Hanson, N. “libinjection: SQL injection detection by tokenization.” Black Hat USA, 2012.
- Juels, A., and Brainard, J. “Client puzzles: a cryptographic countermeasure against connection depletion attacks.” *NDSS*, 1999.
- Lamport, L. “The part-time parliament.” *ACM Transactions on Computer Systems* 16(2), 1998; and “Paxos made simple.” *ACM SIGACT News* 32(4), 2001.
- Mahmoody, M., Moran, T., and Vadhan, S. “Publicly verifiable proofs of sequential work.” *Innovations in Theoretical Computer Science (ITCS)*, 2013.
- O'Connor, J., Aumasson, J.-P., Neves, S., and Wilcox-O'Hearn, Z. *BLAKE3: one function, fast everywhere*. مواصفة، 2020.
- Vyukov, D. "Bounded MPMC queue." 1024cores.net, 2011 (الحلقة الموسومة بأرقام التسلسل المستخدمة لطابور الحوادث).
- Wang, X., Hong, Y., Chang, H., Park, K., Langdale, G., Hu, J., and Zhu, H. “Hyperscan: a fast multi-pattern regex matcher for modern CPUs.” *NSDI*, 2019.
- Sibuna Discussions من 0001 إلى 0014، `docs/sid/records`، 2026: العملية، والمعمارية الأساسية، ومحرك السياسات التصريحي، والفحص الدلالي، ومعمارية تخزين Zaxonlite، والأسس الرياضية، وواجهة إدارة وحدة التحكم، ومراقبة حركة روبوتات الذكاء الاصطناعي، وأجسام الطلبات المجزأة، والنقل إلى Rust، والتخزين عبر sqlx، وTLS الصادر لوحدة التحكم عبر rustls، وتخزين وحدة التحكم، وبصمات JA4 الموثوقة من وكيل الدخول.

الفهرس

B

BLAKE3 ذو مفتاح, **153**

E

Edge, **152**

F

Forward auth, **153**

G

Gate, **153**
GCRA, **153**

H

Hashcash, **153**

J

JA4, **153**

S

Shield, **154**
SID, **154**

W

WEIGH, **154**

Z

Zaxonlite, **154**

ا

البصمة, **152**
الحملة, **152**
الصعوبة، بتات العمل, **152**
الصفحة الوسيطة, **153**
الفتح (Opening), **154**
القبول (Admission), **152**
المصيدة, **153**
الوضع (Surface), **154**
آلة Aho-Corasick, **152**
إثبات العمل المتسلسل (PoSW), **154**

ت

تجزئة القاعدة, **154**

ج

جامع الاتصالات الخاملة, **153**
جدول الحظر, **152**

ر

رمز MAC, **153**

س

سجل التحدي, **152**

ش

شجرة السمعة, **154**

م

مجموعة Robin Hood للحلول المُنفَقة, **154**
مستوى العمل, **153**
موضع المحرك, **152**