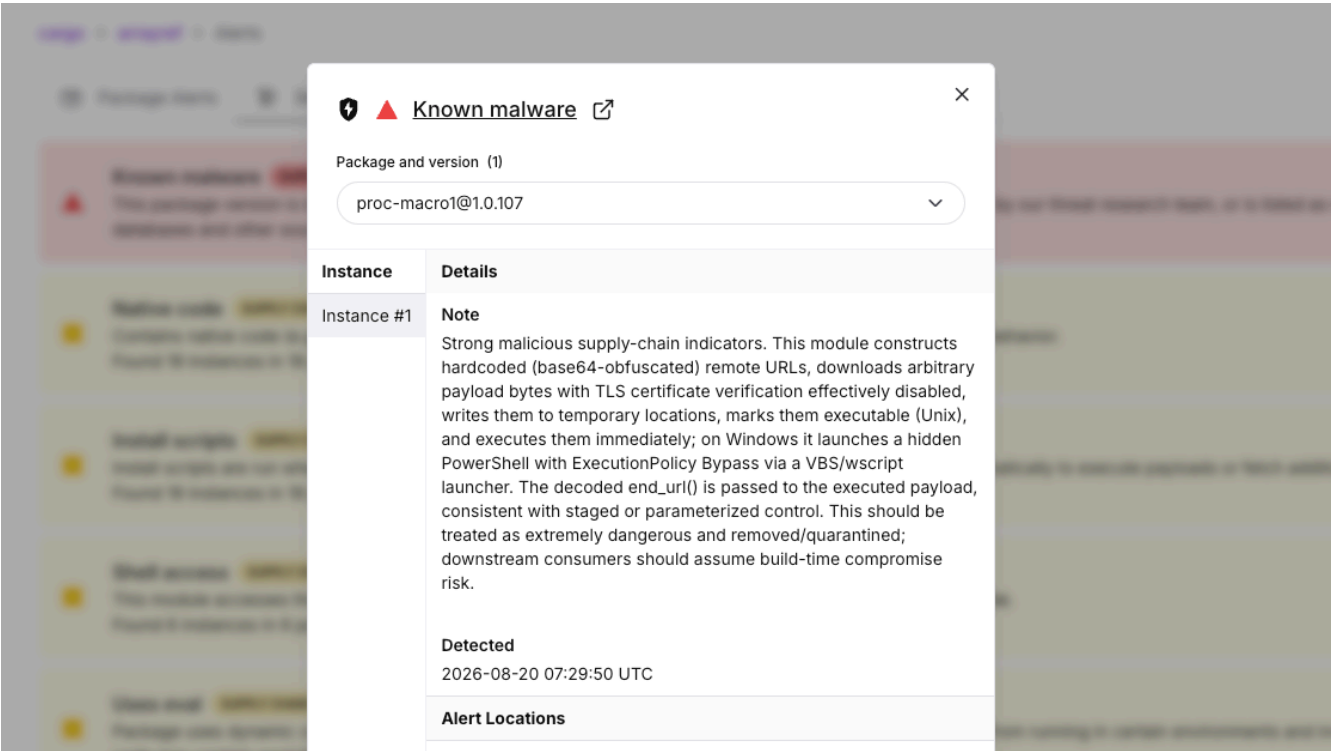


Popular Rust Crates Compromised in Build-Time Supply Chain Attack

 socket.dev/blog/popular-rust-crates-compromised

Socket Research Team

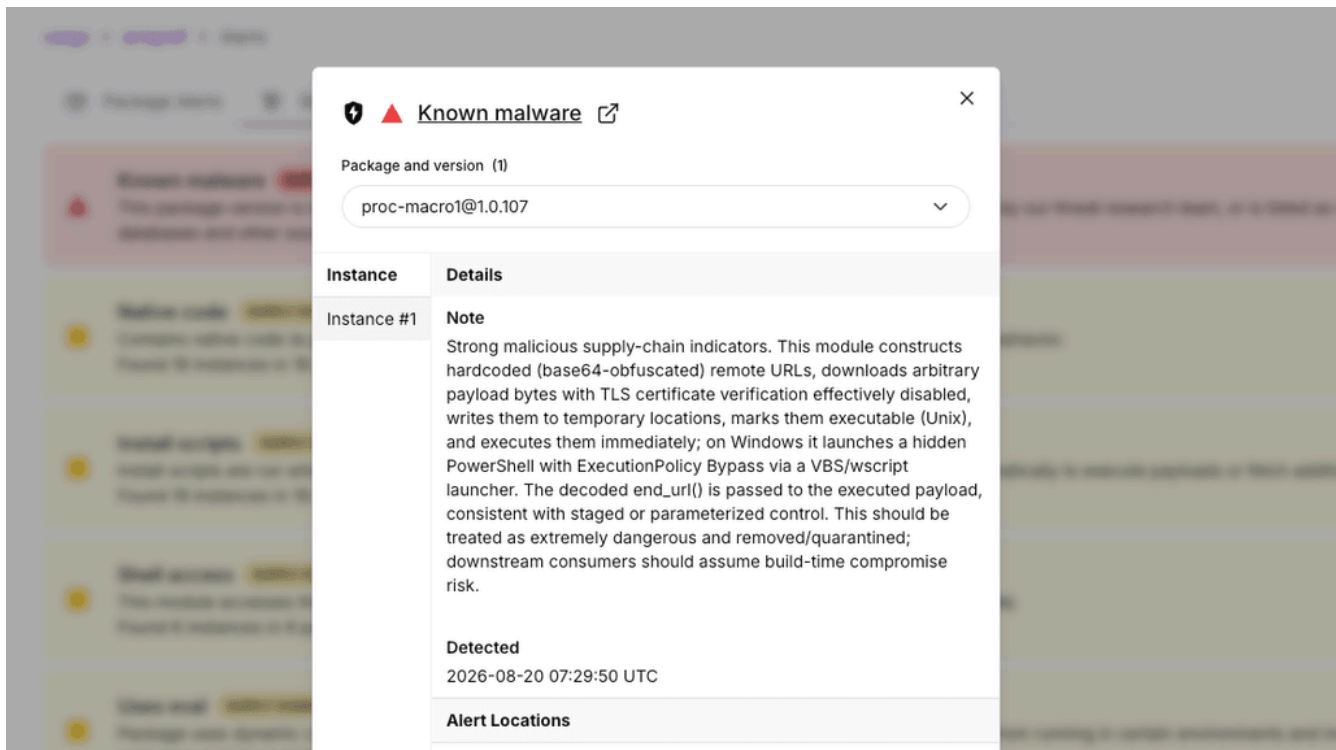
August 20, 2026



Three compromised Rust crates pulled in a malicious dependency that downloaded and executed cross-platform malware during Cargo builds.



Socket Research Team



A threat actor compromised legitimate Rust crates and injected a malicious `proc-macro1` dependency that executed cross-platform malware automatically during Cargo builds.

Socket's Threat Research Team analyzed a coordinated supply chain attack affecting three legitimate Rust crates maintained by David Roundy (droundy):

- [arrayref@0.3.10](#)
- [internment@0.8.7](#)
- [append-only-vec@0.1.9](#)

Socket's AI Scanner independently detected the malicious [proc-macro1](#) crate on August 20, 2026 at 07:29:50 UTC. At that point only `arrayref@0.3.10` had been republished with the malicious dependency; malicious `internment@0.8.7` and `append-only-vec@0.1.9` were published minutes later. Our subsequent analysis of the affected and related crates and recovered malware payloads confirmed the broader attack. Separately, researchers at Nextron Systems reported the activity to the Rust Security Response Team, with technical details also [disclosed](#) through RustSec.

Each malicious release added a dependency on [proc-macro1](#), a threat actor-controlled typosquat impersonating the widely used [proc-macro2](#) crate.

The Rust Security Response Team promptly [removed](#) the affected releases and locked the maintainer account as a precaution. The team stated that it does not believe the legitimate

maintainer acted maliciously and suspects that the maintainer's computer or `crates.io` publishing credentials were compromised, allowing the threat actor to publish malicious versions of `arrayref`, `internment`, and `append-only-vec`.

We recommend treating any system that built one of these malicious versions as potentially compromised.



Known malware



Package and version (1)

proc-macro1@1.0.107

Instance	Details
Instance #1	Note Strong malicious supply-chain indicators. This module constructs hardcoded (base64-obfuscated) remote URLs, downloads arbitrary payload bytes with TLS certificate verification effectively disabled, writes them to temporary locations, marks them executable (Unix), and executes them immediately; on Windows it launches a hidden PowerShell with ExecutionPolicy Bypass via a VBS/wscript launcher. The decoded <code>end_url()</code> is passed to the executed payload, consistent with staged or parameterized control. This should be treated as extremely dangerous and removed/quarantined; downstream consumers should assume build-time compromise risk. Detected 2026-08-20 07:29:50 UTC Alert Locations  proc-macro1-1.0.107/build.rs

The Attack#

The compromised crates themselves contained little obviously malicious code. Instead, their manifests added:

```
proc-macro1 = "1.0.107"
```

When Cargo resolves the dependency, it downloads `proc-macro1` and automatically executes its `build.rs` during compilation. The application does not need to import or call any malicious functionality.

`proc-macro1` closely impersonates the legitimate `proc-macro2`, including copied source code, documentation, metadata, and a deceptive publisher identity. The legitimate-looking library code acts as camouflage while the malicious behavior resides in `build.rs`.

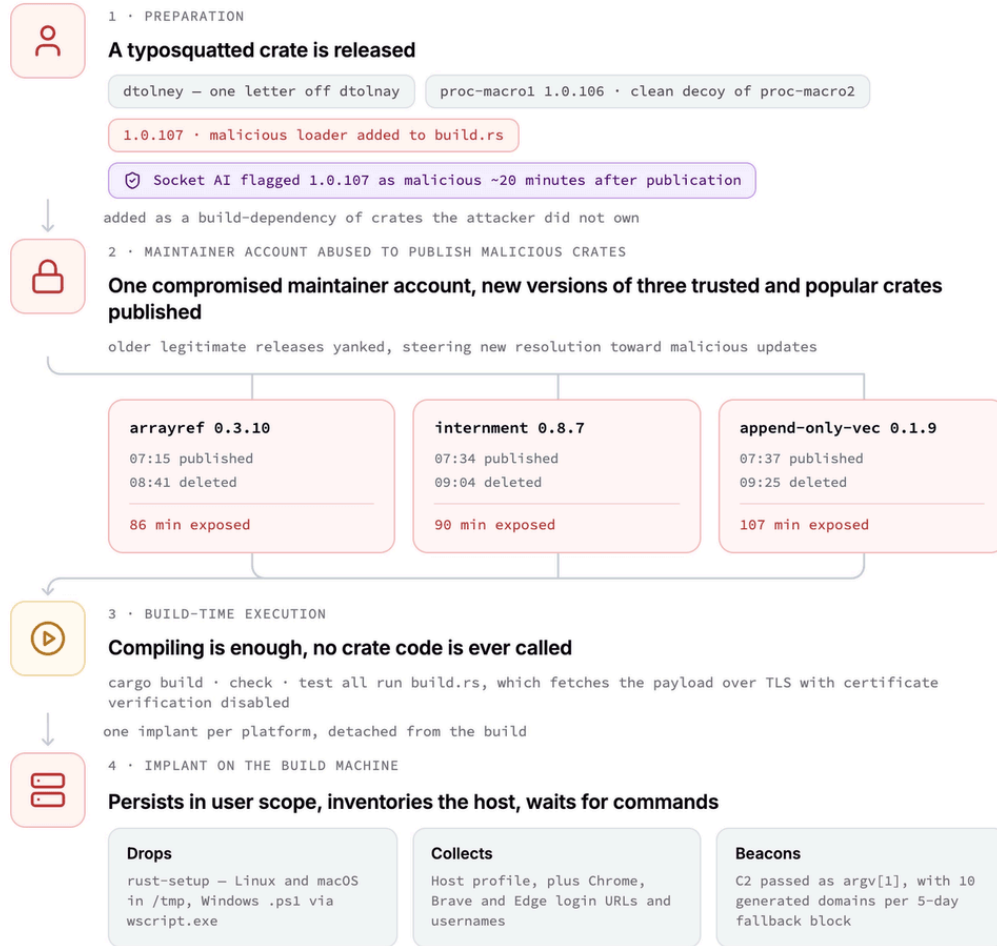
The loader:

- Reconstructs Base64-obfuscated C2 addresses.
- Detects the victim OS and architecture.
- Disables TLS certificate verification.
- Downloads a platform-specific payload from 23[.]254[.]165[.]112:9089.
- Executes it while passing 23[.]254[.]165[.]112:443 as a C2 endpoint.

RustSec [documented](#) the malicious dependency and build script behavior.

One compromised maintainer, three backdoored Rust crates including popular arrayref library

New releases of three widely used crates added a typosquatted build dependency that ran during compilation.



All times 2026-08-20 UTC. The Rust Security Response Team does not believe the maintainer acted maliciously and suspects compromise of their computer or credentials. Other attacker-controlled crates removed in the same time frame: proc-macro-en, aovine, arone, aronenaio, tinymember.

Cross-Platform Execution#

On Linux and macOS, the malware writes:

```
/tmp/rust-setup
```

It makes the file executable and launches it detached.

On Windows, it creates:

```
%TEMP%\rust-setup.ps1  
%TEMP%\rust-setup-launch.vbs
```

It then uses `wscript.exe` to launch hidden PowerShell with `ExecutionPolicy Bypass`, allowing the malicious process to continue independently of Cargo.

The attack therefore provides build-time remote code execution across Linux, macOS, and Windows.

Stage-2 Backdoor Capabilities#

We analyzed three recovered stage-2 payloads for Linux x86-64, Windows x86-64, and macOS ARM64. Despite different implementations, the recovered samples share the same protocol, configuration structure, command set, and cryptographic key material, confirming they are variants of the same cross-platform backdoor.

Once executed, the malware:

- Profiles the host, including username, hostname, OS, architecture, privilege level, and installed applications.
- Inventories Chromium-based browsers, collecting visited login origins, usernames, and installed extension identifiers. The analyzed stage does **not** decrypt stored passwords.
- Establishes user-level persistence through an HKCU Run key on Windows, a `systemd` user service on Linux, or a `LaunchAgent` on macOS.
- Beacons to `/49890878` and supports C2 commands to change configuration, establish persistence, download and execute scripts or shell commands, and terminate the implant.
- Falls back to ten deterministic, date-based `.com` domains when the primary C2 is unavailable and permanently adopts a responsive fallback.

The stage-2 payloads do not embed `23[.]254[.]165[.]112`. Instead, `proc-macro1` passes the C2 address to the payload at execution time, allowing the threat actor to rotate infrastructure without rebuilding the implants. Socket also found little meaningful anti-analysis or obfuscation beyond the Base64-split infrastructure in the stage-1 loader.

Our analysis additionally identified a weakness in the backdoor's command-authentication design. The same RSA private key is embedded across the recovered payloads, and the implementation does not provide exclusive operator authentication. Combined with the predictable DGA, this creates additional opportunities for defensive sinkholing and victim identification, although defenders should not issue commands to third-party systems.

Coordinated Compromise of Legitimate Crates#

Socket confirmed that the malicious versions of all three legitimate crates introduced the same [proc-macro1](#) dependency:

Crate	Malicious	Last Known Clean
<code>arrayref</code>	<code>0.3.10</code>	<code>0.3.9</code>
<code>internment</code>	<code>0.8.7</code>	<code>0.8.6</code>
<code>append-only-vec</code>	<code>0.1.9</code>	<code>0.1.8</code>

All were published within minutes of one another on August 20, 2026, shortly after `proc-macro1` appeared.

The threat actor also [yanked](#) several older legitimate `arrayref` releases, potentially steering new dependency resolution toward malicious `0.3.10`. Rust later restored the legitimate versions.

Broader Attacker Infrastructure#

`proc-macro-en@1.0.10` contained the same malicious `build.rs` as `proc-macro1@1.0.107`, while Socket's analysis of `aovine`, `arone`, and `aronenao` found benign or test-like build scripts consistent with staging activity.

Crates.io also removed:

- `proc-macro-en`
- `aovine`
- `arone`
- `aronenao`
- `tinymember`

Our telemetry provides additional context. `aovine`, `arone`, and `aronenao` appear to be threat actor-controlled clones or staging packages derived from legitimate `droudy` crates. The versions Socket analyzed contained benign or test-like `build.rs` scripts rather than the final C2 loader.

`tinymember` impersonated `tiny-skia` and referenced related packages, but we did not observe the active `proc-macro1` payload in the analyzed version.

These crates appear consistent with development, testing, or staging infrastructure surrounding the attack. They should still be blocked, consistent with Rust's remediation guidance.

Developer and CI/CD Impact#

The compromise occurs during compilation. A developer can be infected simply by running a normal Cargo build that resolves the malicious dependency.

This puts particularly sensitive environments at risk, including:

- Developer workstations
- CI/CD runners
- Release infrastructure
- Automated build systems.

Such systems frequently contain source-control credentials, package publishing tokens, cloud credentials, signing material, and deployment secrets.

arrayref is also deeply embedded in the Rust dependency ecosystem. RustSec [noted](#) roughly 152 million downloads for the prior clean release and transitive dependency paths into widely used Rust GUI frameworks. Download counts do not represent compromised hosts, however. Exposure depends on whether a system actually resolved and built one of the malicious releases.

Recommended Actions#

Organizations should search Cargo.lock, dependency inventories, build logs, and Cargo caches for:

- arrayref 0.3.10
- internment 0.8.7
- append-only-vec 0.1.9
- proc-macro1
- proc-macro-en
- aovine
- arone
- aronenao
- tinymember

Pin affected legitimate crates to:

- arrayref <= 0.3.9
- internment <= 0.8.6
- append-only-vec <= 0.1.8

For systems that built a malicious version:

- Treat the host as potentially compromised.
- Hunt for connections to 23[.]254[.]165[.]112 on ports 9089 and 443.
- Hunt for user-level persistence established after the build, including HKCU Run entries, systemd user services, and macOS LaunchAgents.
- Search for /tmp/rust-setup, rust-setup.ps1, and rust-setup-launch.vbs.
- Investigate suspicious wscript.exe or PowerShell execution associated with Cargo builds.
- Rotate credentials and secrets accessible to affected build environments.
- Rebuild affected software from a known-clean system.

Indicators of Compromise (IOCs)<#>

Network Indicators<#>

- 23[.]254[.]165[.]112:9089
- 23[.]254[.]165[.]112:443
- hxxps://23[.]254[.]165[.]112:443/49890878

Payload Download URLs<#>

- hxxps://23[.]254[.]165[.]112:9089/rust-crate_0.1.0 — Linux x86-64
- hxxps://23[.]254[.]165[.]112:9089/rust-crate_0.2.0 — Windows x86-64
- hxxps://23[.]254[.]165[.]112:9089/rust-crate_0.3.0 — macOS x86-64
- hxxps://23[.]254[.]165[.]112:9089/rust-crate_0.4.0 — macOS ARM64

Stage-2 Payload SHA-256<#>

- 408ef22050ffc5a67e005802809026b29f297a8019f8fda91a2afa8e877ba434 — Linux x86-64 (rust-crate_0.1.0)
- 492f2ab86f8d8911adc79c10ec1541704f5311d207d9d799b0d2a57fcc6a4391 — Windows x86-64 (rust-crate_0.2.0)
- c9561a3b00a0fa38b7772675d987f84bd429c55cd024fc08a98245c2d1632848 — macOS x86-64 (rust-crate_0.3.0)
- 74d3447e7cf99c99ea01a16332ec27432dfb0f491e10e67cd118065a60483306 — macOS ARM64 (rust-crate_0.4.0)

Malicious Loader SHA-256#

cb7778eb6dda91028abf087eb7c3553f981a67e756769507d348e8c201805568 — build.rs
shared by proc-macro1@1.0.107 and proc-macro-en@1.0.10

Host Artifacts#

- /tmp/rust-setup
- %TEMP%\rust-setup.ps1
- %TEMP%\rust-setup.ps1.cfg
- %TEMP%\rust-setup-launch.vbs
- %TEMP%\ps-<GUID>.ps1
- %APPDATA%\<operator-controlled folder>\<name>.ps1
- %APPDATA%\<operator-controlled folder>\<name>.ps1.cfg

DGA Hunting Indicators#

The following domains are deterministically generated by the backdoor for August 20 - 24, 2026 UTC. They are hunting indicators and are not necessarily registered or active.

- rasGThauFD[.]com
- feVVKIiEiU[.]com
- phrpjTNckF[.]com
- PrOkXLgfjW[.]com
- ackeoTaWtl[.]com
- GAFWVCMAja[.]com
- RNSsddnEgK[.]com
- pfHlV0qEeg[.]com
- aBEc0rkups[.]com
- ep0dIaTMaM[.]com

Compromised and Threat Actor-Controlled Crates#

Compromised Legitimate Crates#

- arrayref@0.3.10
- internment@0.8.7
- append-only-vec@0.1.9

Threat Actor-Controlled Crates#

- `proc-macro1`
- `proc-macro-en`
- `aovine`
- `arone`
- `aronenao`
- `tinymember`