Halborn CTF Secutiy Report

Halborn Offensive Security Engineer applying for a full time position

Meek Msaki

Version v1.0, 04.01.2024: Final Report

Table of Contents

Introduction

Document Revisions

0.1	Draft report	03.26.2024
1.0	Final report	04.01.2024
1.1	Fixes review	-

Summary

This security assessment was conducted on the commit e0e91e5…ca1ee of master CTFs repo starting on March 25th, 2024 and ended on April 1st, 2024.

Project Files

```
.
¬¬¬ .DS_Store
¬¬¬ .git
¬¬¬ .gitignore
¬¬¬ HalbornCTF_Golang_Cosmos
¬¬¬ HalbornCTF_Rust_Solana
¬¬¬ HalbornCTF_Rust_Substrate
¬¬¬ HalbornCTF_Solidity_Ethereum
¬¬¬ README.md

5 directories, 3 files
```

Table 1. Languages used and lines of codes

Language	Files	Lines	Blanks	Comments	Code
Protocol Buffers	129	10377	1628	3260	5489
Go	52	9424	1026	537	7861
YAML	48	629	150	11	468
Rust	40	4408	525	745	3138
Shell	40	1368	292	204	872
TOML	17	1001	159	315	527
Terraform	16	948	152	34	762
Markdown	12	417	132	0	285
Solidity	7	511	77	67	367
gitignore	7	53	9	10	34
JSON	6	57851	2	0	57849
Makefile	6	712	158	105	449

Language	Files	Lines	Blanks	Comments	Code
BASH	3	121	28	12	81
Jinja	3	920	127	0	793
License	3	1550	275	0	1275
Python	3	2299	196	199	1904
INI	2	243	45	155	43
SVG	2	26	0	0	26
Dockerfile	1	24	6	7	11
JavaScript	1	167	1	0	166
Plain Text	1	31	0	0	31
Stylus	1	69	10	0	59
Vue	1	11	1	0	10
Total	401	93160	4999	5661	82500

Part I: Critical

1. C-1 btcd mishandles witness size checking

Tags: `runtime`, CVE ID: CVE-2022-44797, GHSA ID: GHSA-2chg-86hq-7w38

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Golang_Cosmos/go.mod#L60

```
++++ ①
    github.com/btcsuite/btcd v0.22.0-beta // indirect
++++
```

1.1. CVSS Score: 9.8/10

Table 2. CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

CVSS base metrics	
Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	High
Integrity	High
Availability	High

btcd before 0.23.2, as used in Lightning Labs lnd before 0.15.2-beta and other Bitcoin-related products, mishandles witness size checking.

1.2. Specific Go Packages Affected

github.com/btcsuite/btcd/wire

2. C-2 crossbeam-deque Data Race before v0.7.4 and v0.8.1

Tags: `runtime`, Weakness: CWE-362, CVE ID: CVE-2021-32810, GHSA ID: GHSA-pqqp-xmhj-wgcw

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L1005

```
++++ ①
[[package]]
name = "crossbeam-deque"
version = "0.7.3"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"9f02af974daeee82218205558e51ec8768b48cf524bd01d550abe5573a608285"
++++
```

2.1. CVSS Score: 9.8/10

Table 3. CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

CVSS base metrics	
Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	High
Integrity	High
Availability	High

2.2. Impact

In the affected version of this crate, the result of the race condition is that one or more tasks in the worker queue can be popped twice instead of other tasks that are forgotten and never popped. If tasks are allocated on the heap, this can cause double free and a memory leak. If not, this still can cause a logical bug.

Crates using `Stealer::steal`, `Stealer::steal_batch`, or `Stealer::steal_batch_and_pop` are affected by this issue.

2.3. Patches

This has been fixed in crossbeam-deque 0.8.1 and 0.7.4.

2.4. Credits

This issue was reported and fixed by Maor Kleinberger.

2.5. License

This advisory is in the public domain.

3. C-3 crossbeam-deque Data Race before v0.7.4 and v0.8.1

Tags: `runtime`, Weakness: CWE-362, CVE ID: CVE-2021-32810, GHSA ID: GHSA-pqqp-xmhj-wgcw

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L1005

```
++++ ①
[[package]]
name = "crossbeam-deque"
version = "0.7.3"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"9f02af974daeee82218205558e51ec8768b48cf524bd01d550abe5573a608285"
++++
```

3.1. CVSS Score: 9.8/10

Table 4. CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

CVSS base metrics	
Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	High
Integrity	High
Availability	High

3.2. Impact

In the affected version of this crate, the result of the race condition is that one or more tasks in the worker queue can be popped twice instead of other tasks that are forgotten and never popped. If tasks are allocated on the heap, this can cause double free and a memory leak. If not, this still can cause a logical bug.

Crates using `Stealer::steal`, `Stealer::steal_batch`, or `Stealer::steal_batch_and_pop` are affected by this issue.

3.3. Patches

This has been fixed in crossbeam-deque 0.8.1 and 0.7.4.

3.4. Credits

This issue was reported and fixed by Maor Kleinberger.

3.5. License

This advisory is in the public domain.

4. C-4 Overflow in libsecp256k1

Tags: `runtime`, Weakness: CWE-190, CWE-347, CVE ID: CVE-2021-38195, GHSA ID: GHSA-g4vj-x7v9-h82m

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L3167

```
++++ ①
[[package]]
name = "libsecp256k1"
version = "0.3.5"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"1fc1e2c808481a63dc6da2074752fdd4336a3c8fcc68b83db6f1fd5224ae7962"
++++
```

4.1. CVSS Score: 9.8/10

Table 5. CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

CVSS base metrics	
Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	High
Integrity	High
Availability	High

An issue was discovered in the libsecp256k1 crate before 0.5.0 for Rust. It can verify an invalid signature because it allows the R or S parameter to be larger than the curve order, aka an overflow.

5. C-5 Out of bounds write in nalgebra

Tags: `runtime`, Weakness: CWE-119, CWE-787, CVE ID: CVE-2021-38190, GHSA ID: GHSA-3w8g-xr3f-2mp8

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L3688

```
++++ ①
[[package]]
name = "nalgebra"
version = "0.19.0"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"0abb021006c01b126a936a8dd1351e0720d83995f4fc942d0d426c654f990745"
++++
```

5.1. CVSS Score: 9.8/10

Table 6. CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

CVSS base metrics	
Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	High
Integrity	High
Availability	High

The `Deserialize` implementation for `VecStorage` did not maintain the invariant that the number of elements must equal `nrows * ncols`. Deserialization of specially crafted inputs could allow memory access beyond allocation of the vector.

This flaw was introduced in v0.11.0 (`086e6e`) due to the addition of an automatically derived implementation of `Deserialize` for `MatrixVec`. `MatrixVec` was later renamed to `VecStorage` in v0.16.13 (`0f66403`) and continued to use the automatically derived implementation of `Deserialize`.

6. C-6 Rust Failure Crate Vulnerable to Type confusion

Tags: `runtime`, Weakness: CWE-843, CVE ID: CVE-2019-25010, GHSA ID: GHSA-r98r-j25q-rmpr

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L1409

```
++++ ①
[[package]]
name = "failure"
version = "0.1.8"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"d32e9bd16cc02eae7db7ef620b392808b89f6a5e16bb3497d159c6b92a0f4f86"
++++
```

6.1. CVSS Score: 9.8/10

Table 7. CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

CVSS base metrics	
Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	High
Integrity	High
Availability	High

Safe Rust code can implement malfunctioning *`private_get_type_id`* and cause type confusion when downcasting, which is an undefined behavior.

Users who derive Fail trait are not affected.

7. C-7 Memory flaw in zeroize_derive

Tags: `runtime`, Weakness: CWE-459, CVE ID: CVE-2021-45706, GHSA ID: GHSA-c5hx-w945-j4pq

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L8936

```
++++ ①
[[package]]
name = "zeroize_derive"
version = "1.1.0"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"a2c1e130bebaeab2f23886bf9acbaca14b092408c452543c857f66399cd6dab1"
++++
```

7.1. CVSS Score: 9.8/10

Table 8. CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

CVSS base metrics	
Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	High
Integrity	High
Availability	High

An issue was discovered in the zeroize_derive crate before 1.1.1 for Rust. Dropped memory is not zeroed out for an enum.

8. C-8 Type confusion if *private_get_type_id* is overriden

Tags: `runtime`, Weakness: CWE-843, CVE ID: CVE-2020-25575, GHSA ID: GHSA-jq66-xh47-j9f3

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L1409

```
++++ ①
[[package]]
name = "failure"
version = "0.1.8"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"d32e9bd16cc02eae7db7ef620b392808b89f6a5e16bb3497d159c6b92a0f4f86"
++++
```

8.1. CVSS Score: 9.8/10

Table 9. CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

CVSS base metrics	
Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	High
Integrity	High
Availability	High

An issue was discovered in the failure crate through 0.1.5 for Rust. It has a type confusion flaw when downcasting. NOTE: This vulnerability only affects products that are no longer supported by the maintainer.

9. C-9 Deserialization of Untrusted Data in rust-cpuid

Tags: `runtime`, Weakness: CWE-502, CVE ID: CVE-2021-45687, GHSA ID: GHSA-w428-f65r-h4q2

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L5154

```
++++ ①
[[package]]
name = "raw-cpuid"
version = "8.1.2"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"1fdf7d9dbd43f3d81d94a49c1c3df73cc2b3827995147e6cf7f89d4ec5483e73"
++++
```

9.1. CVSS Score: 9.8/10

Table 10. CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

CVSS base metrics	
Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	High
Integrity	High
Availability	High

An issue was discovered in the raw-cpuid crate before 9.1.1 for Rust. If the serialize feature is used (which is not the the default), a Deserialize operation may lack sufficient validation, leading to memory corruption or a panic.

Part II: High

10. H-1 Uncontrolled Resource Consumption

Tags: `runtime`, Weakness: CWE-400, CVE ID: CVE-2022-41723, GHSA ID: GHSA-vvpx-j8f3-3w6h

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Golang_Cosmos/go.mod#L247

```
++++ ①
    golang.org/x/net v0.0.0-20211208012354-db4efeb81f4b // indirect
++++
```

10.1. CVSS Score: 7.5/10

Table 11. CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

CVSS base metrics	
Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	None
Integrity	None
Availability	High

A maliciously crafted HTTP/2 stream could cause excessive CPU consumption in the HPACK decoder, sufficient to cause a denial of service from a small number of small requests.

11. H-2 golang.org/x/crypto/ssh Denial of service via crafted Signer

Tags: `runtime`, Weakness: CWE-327, CVE ID: CVE-2022-27191, GHSA ID: GHSA-8c26-wmh5-6g9v

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Golang_Cosmos/go.mod#L245

```
++++ ①
    golang.org/x/crypto v0.0.0-20220214200702-86341886e292 // indirect
++++
```

11.1. CVSS Score: 7.5/10

Table 12. CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

CVSS base metrics	
Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	None
Integrity	None
Availability	High

The golang.org/x/crypto/ssh package before 0.0.0-20220314234659-1baeb1ce4c0b for Go allows an attacker to crash a server in certain circumstances involving AddHostKey.

12. H-3 golang.org/x/net/http2 Denial of Service vulnerability

Tags: `runtime`, CVE ID: CVE-2022-27664, GHSA ID: GHSA-69cg-p879-7622

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Golang_Cosmos/go.mod#L247

```
++++ ①
    golang.org/x/net v0.0.0-20211208012354-db4efeb81f4b // indirect
++++
```

12.1. CVSS Score: 7.5/10

Table 13. CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

CVSS base metrics	
Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	None
Integrity	None
Availability	High

In net/http in Go before 1.18.6 and 1.19.x before 1.19.1, attackers can cause a denial of service because an HTTP/2 connection can hang during closing if shutdown were preempted by a fatal error.

13. H-4 golang.org/x/text/language Denial of service via crafted Accept-Language header

Tags: `runtime`, Weakness: CWE-772, CVE ID: CVE-2022-32149, GHSA ID: GHSA-69ch-w2m2-3vjp

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Golang_Cosmos/go.mod#L251

```
++++ ①
    golang.org/x/text v0.3.7 // indirect
++++
```

13.1. CVSS Score: 7.5/10

Table 14. CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

CVSS base metrics	
Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	None
Integrity	None
Availability	High

The BCP 47 tag parser has quadratic time complexity due to inherent aspects of its design. Since the parser is, by design, exposed to untrusted user input, this can be leveraged to force a program to consume significant time parsing Accept-Language headers. The parser cannot be easily rewritten to fix this behavior for various reasons. Instead the solution implemented in this CL is to limit the total complexity of tags passed into ParseAcceptLanguage by limiting the number of dashes in the string to 1000. This should be more than enough for the majority of real world use cases, where the number of tags being sent is likely to be in the single digits.

13.2. Specific Go Packages Affected

golang.org/x/text/language

14. H-5 golang.org/x/net/http2/h2c vulnerable to request smuggling attack

Tags: `runtime`, Weakness: CWE-444, CVE ID: CVE-2022-41721, GHSA ID: GHSA-fxg5-wq6x-vr4w

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Golang_Cosmos/go.mod#L247

```
++++ ①
    golang.org/x/net v0.0.0-20211208012354-db4efeb81f4b // indirect
++++
```

14.1. CVSS Score: 7.5/10

Table 15. CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

CVSS base metrics	
Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	None
Integrity	None
Availability	High

A request smuggling attack is possible when using MaxBytesHandler. When using MaxBytesHandler, the body of an HTTP request is not fully consumed. When the server attempts to read HTTP2 frames from the connection, it will instead be reading the body of the HTTP request, which could be attacker-manipulated to represent arbitrary HTTP2 requests.

14.2. Specific Go Packages Affected

golang.org/x/net/http2/h2c

15. H-6 Opencontainers runc Incorrect Authorization vulnerability

Tags: `runtime`, Weakness: CWE-706, CVE ID: CVE-2023-27561, GHSA ID: GHSA-vpvm-3wq2-2wvm

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Golang_Cosmos/go.mod#L189

```
++++ ①
    github.com/opencontainers/runc v1.0.3 // indirect
++++
```

15.1. CVSS Score: 7.0/10

Table 16. CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:H/I:H/A:H

CVSS base metrics	
Attack vector	Local
Attack complexity	High
Privileges required	Low
User interaction	None
Scope	Unchange
Confidentiality	High
Integrity	High
Availability	High

runc 1.0.0-rc95 through 1.1.4 has Incorrect Access Control leading to Escalation of Privileges, related to `libcontainer/rootfs_linux.go`. To exploit this, an attacker must be able to spawn two containers with custom volume-mount configurations, and be able to run custom images. NOTE: this issue exists because of a CVE-2019-19921 regression.

16. H-7 Docker Swarm encrypted overlay network may be unauthenticated

Tags: `runtime`, Weakness: CWE-420, CWE-636, CVE ID: CVE-2023-28840, GHSA ID: GHSA-232p-vwff-86mp

File https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Golang_Cosmos/go.mod#L84

```
++++ ①
    github.com/docker/docker v20.10.7+incompatible // indirect
++++
```

16.1. CVSS Score: 7.5/10

Table 17. CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:C/C:H/I:N/A:L

CVSS base metrics	
Attack vector	Network
Attack complexity	High
Privileges required	None
User interaction	None
Scope	Changed
Confidentiality	High
Integrity	None
Availability	Low

Moby is an open source container framework developed by Docker Inc. that is distributed as Docker, Mirantis Container Runtime, and various other downstream projects/products. The Moby daemon component (`dockerd`), which is developed as moby/moby is commonly referred to as **Docker**.

Swarm Mode, which is compiled in and delivered by default in `dockerd` and is thus present in most major Moby downstreams, is a simple, built-in container orchestrator that is implemented through a combination of SwarmKit and supporting network code.

The `overlay` network driver is a core feature of Swarm Mode, providing isolated virtual LANs that allow communication between containers and services across the cluster. This driver is an implementation/user of VXLAN, which encapsulates link-layer (Ethernet) frames in UDP datagrams that tag the frame with a VXLAN Network ID (VNI) that identifies the originating overlay network. In addition, the overlay network driver supports an optional, off-by-default encrypted mode, which is especially useful when VXLAN packets traverses an untrusted network between nodes.

Encrypted overlay networks function by encapsulating the VXLAN datagrams through the use of the IPsec Encapsulating Security Payload protocol in Transport mode. By deploying IPSec encapsulation, encrypted overlay networks gain the additional properties of source authentication through cryptographic proof, data integrity through check-summing, and confidentiality through encryption.

When setting an endpoint up on an encrypted overlay network, Moby installs three iptables (Linux kernel firewall) rules that enforce both incoming and outgoing IPSec. These rules rely on the `u32` iptables extension provided by the `xt_u32` kernel module to directly filter on a VXLAN packet's VNI field, so that IPSec guarantees can be enforced on encrypted overlay networks without interfering with other overlay networks or other users of VXLAN.

Two iptables rules serve to filter incoming VXLAN datagrams with a VNI that corresponds to an encrypted network and discards unencrypted datagrams. The rules are appended to the end of the `INPUT` filter chain, following any rules that have been previously set by the system administrator. Administrator-set rules take precedence over the rules Moby sets to discard unencrypted VXLAN datagrams, which can potentially admit unencrypted datagrams that should have been discarded.

On Red Hat Enterprise Linux and derivatives such as CentOS and Rocky, the `xt_u32` module has been: * moved to the kernel-modules-extra package and no longer installed by default in RHEL 8.3 * officially deprecated in RHEL 8.6 * removed completely in RHEL 9

These rules are not created when `xt_u32` is unavailable, even though the container is still attached to the network.

16.2. Impact

Encrypted overlay networks on affected configurations silently accept cleartext VXLAN datagrams that are tagged with the VNI of an encrypted overlay network. As a result, it is possible to inject arbitrary Ethernet frames into the encrypted overlay network by encapsulating them in VXLAN datagrams.

The injection of arbitrary Ethernet frames can enable a Denial of Service attack. A sophisticated attacker may be able to establish a UDP or TCP connection by way of the container's outbound gateway that would otherwise be blocked by a stateful firewall, or carry out other escalations beyond simple injection by smuggling packets into the overlay network.

16.3. Patches

Patches are available in Moby releases 23.0.3, and 20.10.24. As Mirantis Container Runtime's 20.10 releases are numbered differently, users of that platform should update to 20.10.16.

16.4. Workarounds

- Close the VXLAN port (by default, UDP port 4789) to incoming traffic at the Internet boundary (see GHSA-vwm3-crmr-xfxw) to prevent all VXLAN packet injection.
- Ensure that the `xt_u32` kernel module is available on all nodes of the Swarm cluster.

16.5. Background

- #43382 partially discussed this concern, but did not consider the security implications.
- Mirantis FIELD-5788 essentially duplicates #43382, and was created six months earlier; it similarly overlooked the security implications.
- #45118 is the ancestor of the final patches, and was where the security implications were discovered.

16.6. Related

- CVE-2023-28841: Encrypted overlay network traffic may be unencrypted

- CVE-2023-28842: Encrypted overlay network with a single endpoint is unauthenticated
- GHSA-vwm3-crmr-xfxw: The Swarm VXLAN port may be exposed to attack due to ambiguous documentation
- GHSA-gvm4-2qqg-m333: Security issues in encrypted overlay networks (libnetwork)

17. H-8 gopkg.in/yaml.v3 Denial of Service

Tags: `runtime`, Weakness: CWE-502, CVE ID: CVE-2022-28948, GHSA ID: GHSA-hp87-p4gw-j4gq

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Golang_Cosmos/go.mod#L255

```
++++ ①
    gopkg.in/yaml.v3 v3.0.0-20210107192922-496545a6307b // indirect
++++
```

17.1. CVSS Score: 7.5/10

Table 18. CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

CVSS base metrics	
Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	None
Integrity	None
Availability	High

An issue in the Unmarshal function in Go-Yaml v3 can cause a program to panic when attempting to deserialize invalid input.

18. H-9 HTTP/2 rapid reset can cause excessive work in net/http

Tags: `runtime`, Weakness: CWE-400, CWE-770, CVE ID: CVE-2023-39325, GHSA ID: GHSA-4374-p667-p6c8

File https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Golang_Cosmos/go.mod#L247

```
++++ ①
    golang.org/x/net v0.0.0-20211208012354-db4efeb81f4b // indirect
++++
```

18.1. CVSS Score: 7.5/10

Table 19. CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

CVSS base metrics	
Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	None
Integrity	None
Availability	High

A malicious HTTP/2 client which rapidly creates requests and immediately resets them can cause excessive server resource consumption. While the total number of requests is bounded by the http2.Server.MaxConcurrentStreams setting, resetting an in-progress request allows the attacker to create a new request while the existing one is still executing.

With the fix applied, HTTP/2 servers now bound the number of simultaneously executing handler goroutines to the stream concurrency limit (MaxConcurrentStreams). New requests arriving when at the limit (which can only happen after the client has reset an existing, in-flight request) will be queued until a handler exits. If the request queue grows too large, the server will terminate the connection.

This issue is also fixed in golang.org/x/net/http2 for users manually configuring HTTP/2.

The default stream concurrency limit is 250 streams (requests) per HTTP/2 connection. This value may be adjusted using the golang.org/x/net/http2 package; see the Server.MaxConcurrentStreams setting and the ConfigureServer function.

19. H-10 gRPC-Go HTTP/2 Rapid Reset vulnerability

Tags: `runtime`, GHSA ID: GHSA-m425-mq94-257g

File https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Golang_Cosmos/go.mod#L28

```
++++ ①
    google.golang.org/grpc v1.45.0
++++
```

19.1. CVSS Score: 7.5/10

Table 20. CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

CVSS base metrics	
Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	None
Integrity	None
Availability	High

19.2. Impact

In affected releases of gRPC-Go, it is possible for an attacker to send HTTP/2 requests, cancel them, and send subsequent requests, which is valid by the HTTP/2 protocol, but would cause the gRPC-Go server to launch more concurrent method handlers than the configured maximum stream limit.

19.3. Patches

This vulnerability was addressed by #6703 and has been included in patch releases: 1.56.3, 1.57.1, 1.58.3. It is also included in the latest release, 1.59.0.

Along with applying the patch, users should also ensure they are using the `grpc.MaxConcurrentStreams` server option to apply a limit to the server's resources used for any single connection.

19.4. Workarounds

None.

19.5. References

#6703

20. H-11 runc vulnerable to container breakout through process.cwd trickery and leaked fds

Tags: `runtime`, Weakness: CWE-403, CWE-668, CVE ID: CVE-2024-21626, GHSA ID: GHSA-xr7r-f8xq-vfvv

File https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Golang_Cosmos/go.mod#L189

```
++++ ①
    github.com/opencontainers/runc v1.0.3 // indirect
++++
```

20.1. CVSS Score: 8.6/10

Table 21. CVSS:3.1/AV:L/AC:L/PR:N/UI:R/S:C/C:H/I:H/A:H

CVSS base metrics	
Attack vector	Local
Attack complexity	Low
Privileges required	None
User interaction	Required
Scope	Changed
Confidentiality	High
Integrity	High
Availability	High

20.2. Impact

In runc 1.1.11 and earlier, due to an internal file descriptor leak, an attacker could cause a newly-spawned container process (from `runc exec`) to have a working directory in the host filesystem namespace, allowing for a container escape by giving access to the host filesystem ("attack 2"). The same attack could be used by a malicious image to allow a container process to gain access to the host filesystem through `runc run` ("attack 1"). Variants of attacks 1 and 2 could be also be used to overwrite semi-arbitrary host binaries, allowing for complete container escapes ("attack 3a" and "attack 3b").

Strictly speaking, while attack 3a is the most severe from a CVSS perspective, attacks 2 and 3b are arguably more dangerous in practice because they allow for a breakout from inside a container as opposed to requiring a user execute a malicious image. The reason attacks 1 and 3a are scored higher is because being able to socially engineer users is treated as a given for UI:R vectors, despite attacks 2 and 3b requiring far more minimal user interaction (just reasonable `runc exec` operations on a container the attacker has access to). In any case, all four attacks can lead to full control of the host system.

20.3. Attack 1: `process.cwd` "mis-configuration"

In runc 1.1.11 and earlier, several file descriptors were inadvertently leaked internally within runc into `runc init`, including a handle to the host's `/sys/fs/cgroup` (this leak was added in v1.0.0-rc93). If the container was configured to have `process.cwd` set to `/proc/self/fd/7/` (the actual fd can change depending on file opening order in `runc`), the resulting pid1 process will have a working directory in the host mount namespace and thus the spawned process can access the entire host filesystem. This alone is not an exploit against runc, however a malicious image could make any innocuous-looking non-`/` path a symlink to `/proc/self/fd/7/` and thus trick a user into starting a container whose binary has access to the host filesystem.

Furthermore, prior to runc 1.1.12, runc also did not verify that the final working directory was inside the container's mount namespace after calling `chdir(2)` (as we have already joined the container namespace, it was incorrectly assumed there would be no way to chdir outside the container after `pivot_root(2)`).

The CVSS score for this attack is CVSS:3.1/AV:L/AC:L/PR:N/UI:R/S:C/C:H/I:H/A:N (8.2, high severity).

Note that this attack requires a privileged user to be tricked into running a malicious container image. It should be noted that when using higher-level runtimes (such as Docker or Kubernetes), this exploit can be considered critical as it can be done remotely by anyone with the rights to start a container image (and can be exploited from within Dockerfiles using `ONBUILD` in the case of Docker).

20.4. Attack 2: `runc exec` container breakout

(This is a modification of attack 1, constructed to allow for a process inside a container to break out.)

The same fd leak and lack of verification of the working directory in attack 1 also apply to `runc exec`. If a malicious process inside the container knows that some administrative process will call `runc exec` with the `--cwd` argument and a given path, in most cases they can replace that path with a symlink to `/proc/self/fd/7/`. Once the container process has executed the container binary, `PR_SET_DUMPABLE` protections no longer apply and the attacker can open `/proc/$exec_pid/cwd` to get access to the host filesystem.

`runc exec` defaults to a cwd of `/` (which cannot be replaced with a symlink), so this attack depends on the attacker getting a user (or some administrative process) to use `--cwd` and figuring out what path the target working directory is. Note that if the target working directory is a parent of the program binary being executed, the attacker might be unable to replace the path with a symlink (the `execve` will fail in most cases, unless the host filesystem layout specifically matches the container layout in specific ways and the attacker knows which binary the `runc exec` is executing).

The CVSS score for this attack is CVSS:3.1/AV:L/AC:H/PR:L/UI:R/S:C/C:H/I:H/A:N (7.2, high severity).

20.5. Attacks 3a and 3b: `process.args` host binary overwrite attack

(These are modifications of attacks 1 and 2, constructed to overwrite a host binary by using `execve` to bring a magic-link reference into the container.)

Attacks 1 and 2 can be adapted to overwrite a host binary by using a path like `/proc/self/fd/7/../../../bin/bash` as the `process.args` binary argument, causing a host binary to be executed by a container process. The `/proc/$pid/exe` handle can then be used to overwrite the host binary, as seen in CVE-2019-5736 (note that the same `#!` trick can be used to avoid detection as an attacker). As the overwritten binary could be something like `/bin/bash`, as soon as a privileged user executes the target binary on the host, the attacker can pivot to gain full access to the host.

For the purposes of CVSS scoring:

- Attack 3a is attack 1 but adapted to overwrite a host binary, where a malicious image is set up to execute `/proc/self/fd/7/../../../bin/bash` and run a shell script that overwrites `/proc/self/exe`, overwriting the host copy of `/bin/bash`. The CVSS score for this attack is CVSS:3.1/AV:L/AC:L/PR:N/UI:R/S:C/C:H/I:H/A:H (8.6, high severity).
- Attack 3b is attack 2 but adapted to overwrite a host binary, where the malicious container process overwrites all of the possible `runc exec` target binaries inside the container (such as `/bin/bash`) such that a host target binary is executed and then the container process opens `/proc/$pid/exe` to get access to the host binary and overwrite it. The CVSS score for this attack is CVSS:3.1/AV:L/AC:L/PR:L/UI:R/S:C/C:H/I:H/A:H (8.2, high severity).

As mentioned in attack 1, while 3b is scored lower it is more dangerous in practice as it doesn't require a user to run a malicious image.

20.6. Patches

runc 1.1.12 has been released, and includes patches for this issue. Note that there are four separate fixes applied:

- Checking that the working directory is actually inside the container by checking whether `os.Getwd` returns `ENOENT` (Linux provides a way of detecting if cwd is outside the current namespace root). This explicitly blocks runc from executing a container process when inside a non-container path and thus eliminates attacks 1 and 2 even in the case of fd leaks.
- Close all internal runc file descriptors in the final stage of `runc init`, right before `execve`. This ensures that internal file descriptors cannot be used as an argument to `execve` and thus eliminates attacks 3a and 3b, even in the case of fd leaks. This requires hooking into some Go runtime internals to make sure we don't close critical Go internal file descriptors.
- Fixing the specific fd leaks that made these bug exploitable (mark `/sys/fs/cgroup` as `O_CLOEXEC` and backport a fix for some `*os.File` leaks).
- In order to protect against future `runc init` file descriptor leaks, mark all non-stdio files as `O_CLOEXEC` before executing `runc init`.

20.7. Other Runtimes

We have discovered that several other container runtimes are either potentially vulnerable to similar attacks, or do not have sufficient protection against attacks of this nature. We recommend other container runtime authors look at #Patches[our patches] and make sure they at least add a `getcwd() != ENOENT` check as well as consider whether `close_range(3, UINT_MAX, CLOSE_RANGE_CLOEXEC)` before executing their equivalent of `runc init` is appropriate.

- crun 1.12 does not leak any useful file descriptors into the `runc init`-equivalent process (so this attack is *not exploitable* as far as we can tell), but no care is taken to make sure all non-stdio files are `O_CLOEXEC` and there is no check after `chdir(2)` to ensure the working directory is inside the container. If a file descriptor happened to be leaked in the future, this could be exploitable. In addition, any file descriptors passed to `crun` are not closed until the container process is executed, meaning that easily-overlooked programming errors by users of `crun` can lead to these attacks becoming exploitable.
- youki 0.3.1 does not leak any useful file descriptors into the `runc init`-equivalent process (so this attack is *not exploitable* as far as we can tell) however this appears to be pure luck. `youki` does leak a

directory file descriptor from the host mount namespace, but it just so happens that the directory is the rootfs of the container (which then gets `pivot_root'd into and so ends up as a in-root path thanks to `chroot_fs_refs`). In addition, no care is taken to make sure all non-stdio files are `O_CLOEXEC` and there is no check after `chdir(2)` to ensure the working directory is inside the container. If a file descriptor happened to be leaked in the future, this could be exploitable. In addition, any file descriptors passed to `youki` are not closed until the container process is executed, meaning that easily-overlooked programming errors by users of `youki` can lead to these attacks becoming exploitable.

- LXC 5.0.3 does not appear to leak any useful file descriptors, and they have comments noting the importance of not leaking file descriptors in `lxc-attach`. However, they don't seem to have any proactive protection against file descriptor leaks at the point of `chdir` such as using `close_range(…)` (they do have RAII-like `__do_fclose` closers but those don't necessarily stop all leaks in this context) nor do they have any check after `chdir(2)` to ensure the working directory is inside the container. Unfortunately it seems they cannot use `CLOSE_RANGE_CLOEXEC` because they don't need to re-exec themselves.

20.8. Workarounds

For attacks 1 and 2, only permit containers (and `runc exec`) to use a `process.cwd` of `/`. It is not possible for `/` to be replaced with a symlink (the path is resolved from within the container's mount namespace, and you cannot change the root of a mount namespace or an fs root to a symlink).

For attacks 1 and 3a, only permit users to run trusted images.

For attack 3b, there is no practical workaround other than never using `runc exec` because any binary you try to execute with `runc exec` could end up being a malicious binary target.

20.9. See Also

- https://www.cve.org/CVERecord?id=CVE-2024-21626
- https://github.com/opencontainers/runc/releases/tag/v1.1.12
- The runc 1.1.12 merge commit https://github.com/opencontainers/runc/commit/a9833ff391a71b30069a6c3f816db113379a4346, which contains the following security patches:
- https://github.com/opencontainers/runc/commit/506552a88bd3455e80a9b3829568e94ec0160309
- https://github.com/opencontainers/runc/commit/0994249a5ec4e363bfcf9af58a87a722e9a3a31b
- https://github.com/opencontainers/runc/commit/fbe3eed1e568a376f371d2ced1b4ac16b7d7adde
- https://github.com/opencontainers/runc/commit/284ba3057e428f8d6c7afcc3b0ac752e525957df
- https://github.com/opencontainers/runc/commit/b6633f48a8c970433737b9be5bfe4f25d58a5aa7
- https://github.com/opencontainers/runc/commit/683ad2ff3b01fb142ece7a8b3829de17150cf688
- https://github.com/opencontainers/runc/commit/e9665f4d606b64bf9c4652ab2510da368bfbd951

20.10. Credits

Thanks to Rory McNamara from Snyk for discovering and disclosing the original vulnerability (attack 1) to Docker, @lifubang from acmcoder for discovering how to adapt the attack to overwrite host binaries (attack 3a), and Aleksa Sarai from SUSE for discovering how to adapt the attacks to work as container breakouts using `runc exec` (attacks 2 and 3b).

21. H-12 Memory access due to code generation flaw in Cranelift module

Tags: `runtime`, Weakness: CWE-125, CWE-788, CVE ID: CVE-2021-32629, GHSA ID: GHSA-hpqh-2wqx-7qp5

File https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L901

```
++++ ①
[[package]]
name = "cranelift-codegen"
version = "0.69.0"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"1a54e4beb833a3c873a18a8fe735d73d732044004c7539a072c8faa35ccb0c60"
++++
```

21.1. CVSS Score: 7.2/10

Table 22. CVSS:3.1/AV:L/AC:H/PR:L/UI:R/S:C/C:H/I:H/A:N

CVSS base metrics	
Attack vector	Local
Attack complexity	High
Privileges required	Low
User interaction	Required
Scope	Changed
Confidentiality	High
Integrity	High
Availability	None

There is a bug in 0.73.0 of the Cranelift x64 backend that can create a scenario that could result in a potential sandbox escape in a WebAssembly module. Users of versions 0.73.0 of Cranelift should upgrade to either 0.73.1 or 0.74 to remediate this vulnerability. Users of Cranelift prior to 0.73.0 should update to 0.73.1 or 0.74 if they were not using the old default backend.

21.2. Description

This bug was introduced in the new backend on 2020-09-08 and first included in a release on 2020-09-30, but the new backend was not the default prior to 0.73.0. The recently-released version 0.73.0 with default settings, and prior versions with an explicit build flag to select the new backend, are vulnerable. The bug in question performs a sign-extend instead of a zero-extend on a value loaded from the stack, under a specific set of circumstances. If those circumstances occur, the bug could allow access to memory addresses up to 2GiB before the start of the heap allocated for the WebAssembly module.

If the heap bound is larger than 2GiB, then it would be possible to read memory from a computable range dependent on the size of the heap's bound.

The impact of this bug is highly dependent on heap implementation; specifically: * if the heap has bounds checks, and * does not rely exclusively on guard pages, and * the heap bound is 2GiB or smaller

then this bug cannot be used to reach memory from another WebAssembly module heap.

The impact of the vulnerability is mitigated if there is no memory mapped in the range accessible using this bug, for example, if there is a 2 GiB guard region before the WebAssembly module heap.

The bug in question performs a sign-extend instead of a zero-extend on a value loaded from the stack when the register allocator reloads a spilled integer value narrower than 64 bits. This interacts poorly with another optimization: the instruction selector elides a 32-to-64-bit zero-extend operator when we know that an instruction producing a 32-bit value actually zeros the upper 32 bits of its destination register. Hence, we rely on these zeroed bits, but the type of the value is still i32, and the spill/reload reconstitutes those bits as the sign extension of the i32's MSB.

The issue would thus occur when: * An i32 value is greater than or equal to 0x8000_0000; * The value is spilled and reloaded by the register allocator due to high register pressure in the program between the value's definition and its use; * The value is produced by an instruction that we know to be "special" in that it zeroes the upper 32 bits of its destination: add, sub, mul, and, or; * The value is then zero-extended to 64 bits; * The resulting 64-bit value is used.

Under these circumstances there is a potential sandbox escape when the i32 value is a pointer. The usual code emitted for heap accesses zero-extends the WebAssembly heap address, adds it to a 64-bit heap base, and accesses the resulting address. If the zero-extend becomes a sign-extend, the module could reach backward and access memory up to 2GiB before the start of its heap.

This bug was identified by developers at Fastly following a report from Javier Cabrera Arteaga, KTH Royal Institute of Technology, with support from project Trustful of Stiftelsen för Strategisk Forskning. In addition to supporting the analysis and remediation of this vulnerability, Fastly will publish a related Fastly Security Advisory at https://www.fastly.com/security-advisories.

In addition to assessing the nature of the code generation bug in Cranelift, we have also determined that under specific circumstances, both Lucet and Wasmtime using this version of Cranelift may be exploitable.

21.3. General Impact to Lucet

Lucet inherits the heap address computation and bounds-checks of Cranelift, which it uses as its backend code generator. Of particular importance specifically is the address-space layout used by Lucet. In the default configuration for Lucet, only a single module is running, and therefore it is not possible to access memory from another module.

By default, the open source implementation of Lucet uses a maximum heap size of 4 GiB, and an instance slot size of 8 GiB, when invoking an instance from the lucet-wasi command-line tool. These settings are within the range of vulnerability described above, but only a single instance is running, so there is no other instance to read. When embedding the runtime (for example, in a long-running daemon), the default for the heap size as described in the source is 1MB; with this setting, the runtime is not vulnerable.

Lucet allocates its WebAssembly module instances into "instance slots", which are contiguous zones of virtual address space that contain the VM context at the bottom, the WebAssembly heap in the next page after that, a guard region in the middle, and other data at the top: the stack and the globals.

If the instance slot size is less than (max heap) + 2GiB, then the lowest accessible address using the bug will overlap with the prior instance's heap. If the size of VM context + stack + globals is greater than (4GiB -

heap limit), then the highest accessible address using the bug will overlap with this critical data. If neither of these conditions are true, the bug should only result in an access to the prior instance's guard region.

Generally, if the limit is between 2GiB and 4GiB - ~1MB (depending on stack/global size) and the instance slot size is less than 6GiB, the configuration is vulnerable. If the limit is greater than 4GiB - ~1MB, the configuration is vulnerable regardless of instance slot size. Otherwise, the configuration is not vulnerable.

21.4. General Impact on Wasmtime

In Wasmtime, the same Cranelift heap address computations and heap types are used as above. The memory layout, however, is slightly different, with different outcomes: * With the mmap implementation impact is mitigated probabilistically if ASLR is enabled. * With the pooling allocator, the vulnerability only exists if a memory reservation size lower than the default of 6GB is used.

With the default mmap-based instance memory implementation, Wasmtime uses mmap() to allocate a block of memory large enough for the heap and guard region, as specified in its configuration. If the underlying OS implements ASLR (modern Linux, macOS and Windows do) then this address will be randomized, and the region below it will (probabilistically) be free. Hence, the bug is mitigated probabilistically in the default configuration if ASLR is enabled.

If using the pooling allocator, the vulnerability exists if instance memory size (`memory_reservation_size` in InstanceLimit) is strictly less than 6GiB (4 GiB + 2 GiB of guard pages). The default is 6GiB, so the vulnerability is masked in the default pooling allocator configuration.

22. H-13 Overflow in prost-types

Tags: `runtime`, Weakness: CWE-120, CWE-190, CVE ID: CVE-2021-38192, GHSA ID: GHSA-x4qm-mcjq-v2gf

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L4937

```
++++ ①
[[package]]
name = "prost-types"
version = "0.7.0"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"b518d7cdd93dab1d1122cf07fa9a60771836c668dde9d9e2a139f957f0d9f1bb"
++++
```

22.1. CVSS Score: 7.5/10

Table 23. CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

CVSS base metrics	
Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	None
Integrity	None
Availability	High

Affected versions of this crate contained a bug in which untrusted input could cause an overflow and panic when converting a Timestamp to SystemTime. It is recommended to upgrade to prost-types v0.8 and switch the usage of From<Timestamp> for SystemTime to TryFrom<Timestamp> for SystemTime.

23. H-14 Soundness issue in raw-cpuid

Tags: runtime, Weakness: CWE-198, CWE-400, CVE ID: CVE-2021-26306, GHSA ID: GHSA-hvqc-pc78-x9wh

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L5154

```
++++ ①
[[package]]
name = "raw-cpuid"
version = "8.1.2"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"1fdf7d9dbd43f3d81d94a49c1c3df73cc2b3827995147e6cf7f89d4ec5483e73"
++++
```

23.1. CVSS Score: 7.5/10

Table 24. CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

CVSS base metrics	
Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	None
Integrity	None
Availability	High

VendorInfo::as_string(), SoCVendorBrand::as_string(), and ExtendedFunctionInfo::processor_brand_string() construct byte slices using std::slice::from_raw_parts(), with data coming from #[repr(Rust)] structs. This is always undefined behavior. This flaw has been fixed in v9.0.0, by making the relevant structs #[repr©].

24. H-15 Use After Free in lru

Tags: `runtime`, Weakness: CWE-416, CVE ID: CVE-2021-45720, GHSA ID: GHSA-v362-2895-h9r2

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L3247

```
++++ ①
[[package]]
name = "lru"
version = "0.6.5"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"1f374d42cdfc1d7dbf3d3dec28afab2eb97ffbf43a3234d795b5986dbf4b90ba"
++++
```

24.1. CVSS Score: 7.5/10

Table 25. CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

CVSS base metrics	
Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	None
Integrity	None
Availability	High

Lru crate has two functions for getting an iterator. Both iterators give references to key and value. Calling specific functions, like pop(), will remove and free the value, and but it’s still possible to access the reference of value which is already dropped causing use after free.

25. H-16 crossbeam-utils Race Condition vulnerability

Tags: `runtime`, Weakness: CWE-362, CVE ID: CVE-2022-23639, GHSA ID: GHSA-qc84-gqf4-9926

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L1066

```
++++ ①
[[package]]
name = "crossbeam-utils"
version = "0.7.2"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"c3c7c73a2d1e9fc0886a08b93e98eb643461230d5f1925e4036204d5f2e261a8"
++++
```

25.1. CVSS Score: 8.1/10

Table 26. CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H

CVSS base metrics	
Attack vector	Network
Attack complexity	High
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	High
Integrity	High
Availability	High

25.2. Impact

The affected version of this crate incorrectly assumed that the alignment of `{i,u}64` was always the same as `Atomic{I,U}64`.

However, the alignment of `{i,u}64` on a 32-bit target can be smaller than `Atomic{I,U}64`.

This can cause the following problems:

- Unaligned memory accesses
- Data race

Crates using `fetch_*` methods with `AtomicCell<{i,u}64>` are affected by this issue.

32-bit targets without `Atomic{I,U}64` and 64-bit targets are not affected by this issue. 32-bit targets with `Atomic{I,U}64` and `{i,u}64` have the same alignment are also not affected by this issue.

The following is a complete list of the builtin targets that may be affected. (last update: nightly-2022-02-11)

- armv7-apple-ios (tier 3)
- armv7s-apple-ios (tier 3)
- i386-apple-ios (tier 3)
- i586-unknown-linux-gnu
- i586-unknown-linux-musl
- i686-apple-darwin (tier 3)
- i686-linux-android
- i686-unknown-freebsd
- i686-unknown-haiku (tier 3)
- i686-unknown-linux-gnu
- i686-unknown-linux-musl
- i686-unknown-netbsd (tier 3)
- i686-unknown-openbsd (tier 3)
- i686-wrs-vxworks (tier 3)

(script to get list)

25.3. Patches

This has been fixed in crossbeam-utils 0.8.7.

Affected 0.8.x releases have been yanked.

25.4. References

https://github.com/crossbeam-rs/crossbeam/pull/781

25.5. License

This advisory is in the public domain.

26. H-17 Rust’s regex crate vulnerable to regular expression denial of service

Tags: `runtime`, Weakness: CWE-400, CWE-1333, CVE ID: CVE-2022-24713, GHSA ID: GHSA-m5pq-gvj9-9vr8

File https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L5261

```
++++ ①
[[package]]
name = "regex"
version = "1.5.4"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"d07a8629359eb56f1e2fb1652bb04212c072a87ba68546a04065d525673ac461"
++++
```

26.1. CVSS Score: 7.5/10

Table 27. CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

CVSS base metrics	
Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	None
Integrity	None
Availability	High

> This is a cross-post of [the official security advisory][advisory]. The official advisory contains a signed version with our PGP key, as well.

[advisory]: https://groups.google.com/g/rustlang-security-announcements/c/NcNNL1Jq7Yw

The Rust Security Response WG was notified that the `regex` crate did not properly limit the complexity of the regular expressions (regex) it parses. An attacker could use this security issue to perform a denial of service, by sending a specially crafted regex to a service accepting untrusted regexes. No known vulnerability is present when parsing untrusted input with trusted regexes.

This issue has been assigned CVE-2022-24713. The severity of this vulnerability is "high" when the `regex` crate is used to parse untrusted regexes. Other uses of the `regex` crate are not affected by this vulnerability.

26.2. Overview

The `regex` crate features built-in mitigations to prevent denial of service attacks caused by untrusted regexes, or untrusted input matched by trusted regexes. Those (tunable) mitigations already provide sane defaults to prevent attacks. This guarantee is documented and it's considered part of the crate's API.

Unfortunately a bug was discovered in the mitigations designed to prevent untrusted regexes to take an arbitrary amount of time during parsing, and it's possible to craft regexes that bypass such mitigations. This makes it possible to perform denial of service attacks by sending specially crafted regexes to services accepting user-controlled, untrusted regexes.

26.3. Affected versions

All versions of the `regex` crate before or equal to 1.5.4 are affected by this issue. The fix is include starting from `regex` 1.5.5.

26.4. Mitigations

We recommend everyone accepting user-controlled regexes to upgrade immediately to the latest version of the `regex` crate.

Unfortunately there is no fixed set of problematic regexes, as there are practically infinite regexes that could be crafted to exploit this vulnerability. Because of this, we do not recommend denying known problematic regexes.

26.5. Acknowledgements

We want to thank Addison Crump for responsibly disclosing this to us according to the Rust security policy, and for helping review the fix.

We also want to thank Andrew Gallant for developing the fix, and Pietro Albini for coordinating the disclosure and writing this advisory.

27. H-18 Use after free in Wasmtime

Tags: `runtime`, Weakness: CWE-416, CVE ID: CVE-2022-24791, GHSA ID: GHSA-gwc9-348x-qwv2

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L8590

```
++++ ①
[[package]]
name = "wasmtime"
version = "0.22.0"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"7426055cb92bd9a1e9469b48154d8d6119cd8c498c8b70284e420342c05dc45d"
++++
```

27.1. CVSS Score: 8.1/10

Table 28. CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H

CVSS base metrics	
Attack vector	Network
Attack complexity	High
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	High
Integrity	High
Availability	High

There is a use after free vulnerability in Wasmtime when both running Wasm that uses `externref`s and enabling epoch interruption in Wasmtime. If you are not explicitly enabling epoch interruption (it is disabled by default) then you are not affected. If you are explicitly disabling the Wasm reference types proposal (it is enabled by default) then you are also not affected.

The use after free is caused by Cranelift failing to emit stack maps when there are safepoints inside cold blocks. Cold blocks occur when epoch interruption is enabled. Cold blocks are emitted at the end of compiled functions, and change the order blocks are emitted versus defined. This reordering accidentally caused Cranelift to skip emitting some stack maps because it expected to emit the stack maps in block definition order, rather than block emission order. When Wasmtime would eventually collect garbage, it would fail to find live references on the stack because of the missing stack maps, think that they were unreferenced garbage, and therefore reclaim them. Then after the collection ended, the Wasm code could use the reclaimed-too-early references, which is a use after free.

This bug was discovered while extending our fuzz targets for `externref`s and GC in Wasmtime. The updated fuzz target thoroughly exercises these code paths and feature combinations now. We have also added a regression test for this bug. Released versions 0.34.2 and 0.35.2, which fix the vulnerability. We recommend all Wasmtime users upgrade to these patched versions. If upgrading is not an option for you at this time, you can avoid the vulnerability by either disabling the Wasm reference types proposal or by disabling epoch interruption if you were previously enabling it.

28. H-19 Parser creates invalid uninitialized value

Tags: `runtime`, GHSA ID: GHSA-f67m-9j94-qv9j

File https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L2259

```
++++ ①
[[package]]
name = "hyper"
version = "0.12.36"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"5c843caf6296fc1f93444735205af9ed4e109a539005abb2564ae1d6fad34c52"
++++
```

Affected versions of this crate called `mem::uninitialized()` in the HTTP1 parser to create values of type `httparse::Header` (from the `httparse` crate). This is unsound, since `Header` contains references and thus must be non-null.

The flaw was corrected by avoiding the use of `mem::uninitialized()`, using `MaybeUninit` instead.

29. H-20 Use after free in lru crate

Tags: `runtime`, GHSA ID: GHSA-qqmc-hwqp-8g2w

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L3247

```
++++ ①
[[package]]
name = "lru"
version = "0.6.5"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"1f374d42cdfc1d7dbf3d3dec28afab2eb97ffbf43a3234d795b5986dbf4b90ba"
++++
```

Lru crate has use after free vulnerability.

Lru crate has two functions for getting an iterator. Both iterators give references to key and value. Calling specific functions, like pop(), will remove and free the value, and but it’s still possible to access the reference of value which is already dropped causing use after free.

30. H-21 Data race in `Iter` and `IterMut`

Tags: `runtime`, Weakness: CWE-362, GHSA ID: GHSA-9hpw-r23r-xgm5

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L7766

```
++++ ①
[[package]]
name = "thread_local"
version = "1.1.3"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"8018d24e04c95ac8790716a5987d0fec4f8b27249ffa0f7d33f1369bdfb88cbd"
++++
```

In the affected version of this crate, `{Iter, IterMut}::next` used a weaker memory ordering when loading values than what was required, exposing a potential data race when iterating over a `ThreadLocal's values.

Crates using `Iter::next`, or `IterMut::next` are affected by this issue.

31. H-22 Wasmtime may have data leakage between instances in the pooling allocator

Tags: `runtime`, Weakness: CWE-212, CWE-226, CVE ID: CVE-2022-39393, GHSA ID: GHSA-wh6w-3828-g9qf

File https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L8590

```
++++ ①
[[package]]
name = "wasmtime"
version = "0.22.0"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"7426055cb92bd9a1e9469b48154d8d6119cd8c498c8b70284e420342c05dc45d"
++++
```

31.1. CVSS Score: 8.6/10

Table 29. CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:N/A:N

CVSS base metrics	
Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Changed
Confidentiality	High
Integrity	None
Availability	None

31.2. Impact

There is a bug in Wasmtime's implementation of it's pooling instance allocator where when a linear memory is reused for another instance the initial heap snapshot of the prior instance can be visible, erroneously to the next instance. The pooling instance allocator in Wasmtime works by preallocating virtual memory for a fixed number of instances to reside in and then new instantiations pick a slot to use. Most conventional modules additionally have an initial copy-on-write "heap image" which is mapped in Wasmtime into the linear memory slot. When a heap slot is deallocated Wasmtime resets all of its contents back to the initial state but it does not unmap the image in case the next instance is an instantiation of the same module.

The bug in Wasmtime occurs when a slot in the pooling allocator previously was used for a module with a heap image, meaning that its current state of memory contains the initial heap contents of that module. If the next instantiation within that slot does not itself contain a heap image then Wasmtime would leave the old heap image in place erroneously and continue with instantiation. This means that instantiations of modules without a heap image can see the initial heap image of the prior instantiation within that slot.

Heap images in Wasmtime are created by precomputing WebAssembly `data` segments into one large mapping to be placed into linear memory at a particular offset. Most modules produced by toolchains today will have a heap image and an initialization snapshot. Creating a module without a heap image would require a hand-crafted `*.wat` file or a specially crafted source program. This consequence means that this bug is highly unlikely to be accidentally triggered and would otherwise require an intentional trigger with a hand-crafted module.

One important part of this vulnerability is Wasmtime is highly likely to segfault when the slot is reused again with a module that itself has an initialization image. For example if module A has a heap initialization image and module B does not have a heap initialization image, then the following sequence of events could happen if they all are instantiated into the same instance slot:

- Module A is instantiated, and then deallocated. This leaves A's heap image in place, reset to its initial contents.
- Module B is instantiated and erroneously can see the initial heap contents of A. Module B is then deallocated and the entire heap is unmapped and reset back to zero.
- Module A is instantiated again, but the state tracking the slot did not account for module B so it thinks the module image is still mapped and proceeds with instantiation. Any action on A's part to access linear memory will then trap and if the host accesses A's memory it will segfault because the data that's supposed to be mapped is all unmapped.

Adding this all together this means that in practice modules must be deliberately crafted to not have an initial heap image to view the contents of a prior image. If this module is instantiated though then when the slot is reused the next, likely image-using, module will believe its memory is mapped when it isn't, causing the host to segfault on unmapped memory it believed was mapped.

31.3. Patches

This bug has been patched and users should upgrade to Wasmtime 2.0.2.

31.4. Workarounds

Trigging this bug requires the pooling allocator to be configured and for copy-on-write heap images to also be enabled. Pooling allocation is not enabled by default but copy-on-write heap images are. Mitigations for this bug include:

- Disabling the pooling allocator - note that pooling allocation is not enabled by default in Wasmtime
- Disabling the `memory-init-cow` feature or with `Config::memory_init_cow`

31.5. References

- `Config::allocation_strategy` - configuration required to enable the pooling allocator.
- `Config::memory_init_cow` - configuration required to enable or disable copy-on-write (this is enabled by default).
- Mailing list announcement
- Patch for `release-2.0.0` branch
- Patch for `main`

31.6. For more information

If you have any questions or comments about this advisory:

- Reach out to us on the Bytecode Alliance Zulip chat
- Open an issue in the bytecodealliance/wasmtime repository

32. H-23 libp2p DoS vulnerability from lack of resource management

Tags: `runtime`, Weakness: CWE-400, CWE-770, CVE ID: CVE-2022-23486, GHSA ID: GHSA-jvgw-gccv-q5p8

File https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L2748

```
++++ ①
[[package]]
name = "libp2p"
version = "0.34.0"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"d5133112ce42be9482f6a87be92a605dd6bbc9e93c297aee77d172ff06908f3a"
++++
```

32.1. CVSS Score: 7.5/10

Table 30. CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

CVSS base metrics	
Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	None
Integrity	None
Availability	High

32.2. Impact

An attacker node can cause a victim node to allocate a large number of small memory chunks, which can ultimately lead to the victim’s process running out of memory and thus getting killed by its operating system. When executed continuously, this can lead to a denial of service attack, especially relevant on a larger scale when run against more than one node of a libp2p based network.

32.3. Details

In the original version of the attack, the malicious node would continuously open new streams on a single connection using a stream multiplexer that doesn’t provide sufficient back pressure (mplex or yamux). While allocations per stream might be considered small, they multiply with the number of streams and connections. It is easy to defend against this one attack, e.g. by setting a strict per connection stream limit and connection limit. But there are other variations of this attack, e.g. causing memory allocations by sending partial payloads on various protocol levels, forcing the victim to buffer the partial payload for a period of time or by tricking the victim into pre-allocating buffers for messages which are never sent by the attacker.

32.4. Patches

Users are advised to upgrade to `libp2p` `v0.45.1` or above.

32.5. References

Please see our DoS Mitigation page for more information on how to incorporate mitigation strategies, monitor your application, and respond to attacks: https://docs.libp2p.io/reference/dos-mitigation/.

Please see the related disclosure for go-libp2p: https://github.com/libp2p/go-libp2p/security/advisories/GHSA-j7qp-mfxf-8xjw and js-libp2p: https://github.com/libp2p/js-libp2p/security/advisories/GHSA-f44q-634c-jvwv

32.6. For more information

If you have any questions or comments about this advisory, please email us at security@libp2p.io.

33. H-24 Race Condition in tokio

Tags: `runtime`, Weakness: CWE-362, CVE ID: CVE-2021-45710, GHSA ID: GHSA-fg7r-2g4j-5cgr

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L7837

```
++++ ①
[[package]]
name = "tokio"
version = "0.1.22"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"5a09c0b5bb588872ab2f09afa13ee6e9dac11e10a0ec9e8e3ba39a5a5d530af6"
++++
```

33.1. CVSS Score: 8.1/10

Table 31. CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H

CVSS base metrics	
Attack vector	Network
Attack complexity	High
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	High
Integrity	High
Availability	High

If a tokio::sync::oneshot channel is closed (via the oneshot::Receiver::close method), a data race may occur if the oneshot::Sender::send method is called while the corresponding oneshot::Receiver is awaited or calling try_recv.

When these methods are called concurrently on a closed channel, the two halves of the channel can concurrently access a shared memory location, resulting in a data race. This has been observed to cause memory corruption.

Note that the race only occurs when both halves of the channel are used after the Receiver half has called close. Code where close is not used, or where the Receiver is not awaited and try_recv is not called after calling close, is not affected.

34. H-25 webpki: CPU denial of service in certificate path building

Tags: `runtime`, Weakness: CWE-400, GHSA ID: GHSA-8qv2-5vq6-g2g7

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L8805

```
++++ ①
[[package]]
name = "webpki"
version = "0.21.4"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"b8e38c0608262c46d4a56202ebabdeb094cef7e560ca7a226c6bf055188aa4ea"
++++
```

34.1. CVSS Score: 7.5/10

Table 32. CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

CVSS base metrics	
Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	None
Integrity	None
Availability	High

When this crate is given a pathological certificate chain to validate, it will spend CPU time exponential with the number of candidate certificates at each step of path building.

Both TLS clients and TLS servers that accept client certificate are affected.

This was previously reported in https://github.com/briansmith/webpki/issues/69.

`rustls-webpki` is a fork of this crate which contains a fix for this issue and is actively maintained.

35. H-26 Multiple issues involving quote API in shlex

Tags: `runtime`, GHSA ID: GHSA-r7qv-8r2h-pg27

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L6680

```
++++ ①
[[package]]
name = "shlex"
version = "0.1.1"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"7fdf1b9db47230893d76faad238fd6097fd6d6a9245cd7a4d90dbd639536bbd2"
++++
```

35.1. Issue 1: Failure to quote characters

Affected versions of this crate allowed the bytes `{` and `\xa0` to appear unquoted and unescaped in command arguments.

If the output of `quote` or `join` is passed to a shell, then what should be a single command argument could be interpreted as multiple arguments.

This does not **directly** allow arbitrary command execution (you can't inject a command substitution or similar). But depending on the command you're running, being able to inject multiple arguments where only one is expected could lead to undesired consequences, potentially including arbitrary command execution.

The flaw was corrected in version 1.2.1 by escaping additional characters. Updating to 1.3.0 is recommended, but 1.2.1 offers a more minimal fix if desired.

Workaround: Check for the bytes `{` and `\xa0` in `quote`/`join` input or output.

(Note: `{` is problematic because it is used for glob expansion. `\xa0` is problematic because it's treated as a word separator in [specific environments][solved-xa0].)

35.2. Issue 2: Dangerous API w.r.t. nul bytes

Version 1.3.0 deprecates the `quote` and `join` APIs in favor of `try_quote` and `try_join`, which behave the same except that they have `Result` return type, returning `Err` if the input contains nul bytes.

Strings containing nul bytes generally cannot be used in Unix command arguments or environment variables, and most shells cannot handle nul bytes even internally. If you try to pass one anyway, then the results might be security-sensitive in uncommon scenarios. [More details here.][nul-bytes]

Due to the low severity, the behavior of the original `quote` and `join` APIs has not changed; they continue to allow nuls.

Workaround: Manually check for nul bytes in `quote`/`join` input or output.

35.3. Issue 3: Lack of documentation for interactive shell risks

The `quote` family of functions does not and cannot escape control characters. With non-interactive shells this is perfectly safe, as control characters have no special effect. But if you writing directly to the standard input of an interactive shell (or through a pty), then control characters [can cause misbehavior including arbitrary command injection.][control-characters]

This is essentially unfixable, and has not been patched. But as of version 1.3.0, documentation has been added.

Future versions of `shlex` may add API variants that avoid the issue at the cost of reduced portability.

[solved-xa0]: https://docs.rs/shlex/latest/shlex/quoting_warning/index.html#solved-xa0 [nul-bytes]: https://docs.rs/shlex/latest/shlex/quoting_warning/index.html#nul-bytes [control-characters]: https://docs.rs/shlex/latest/shlex/quoting_warning/index.html#control-characters-interactive-contexts-only

Part III: Medium

36. M-1 Default inheritable capabilities for linux container should be empty

Tags: `runtime`, Weakness: CWE-276, CVE ID: CVE-2022-29162, GHSA ID: GHSA-f3fp-gc8g-vw66

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Golang_Cosmos/go.mod#L189

```
++++ ①
    github.com/opencontainers/runc v1.0.3 // indirect
++++
```

36.1. CVSS Score: 5.9/10

Table 33. CVSS:3.1/AV:L/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:L

CVSS base metrics	
Attack vector	Local
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	Low
Integrity	Low
Availability	Low

36.2. Impact

A bug was found in runc where `runc exec --cap` executed processes with non-empty inheritable Linux process capabilities, creating an atypical Linux environment and enabling programs with inheritable file capabilities to elevate those capabilities to the permitted set during execve(2).

This bug did not affect the container security sandbox as the inheritable set never contained more capabilities than were included in the container's bounding set.

36.3. Patches

This bug has been fixed in runc 1.1.2. Users should update to this version as soon as possible.

This fix changes `runc exec --cap` behavior such that the additional capabilities granted to the process being executed (as specified via `--cap` arguments) do not include inheritable capabilities.

In addition, `runc spec` is changed to not set any inheritable capabilities in the created example OCI spec (`config.json`) file.

36.4. Credits

The opencontainers project would like to thank Andrew G. Morgan for responsibly disclosing this issue in accordance with the opencontainers org security policy.

36.5. For more information

If you have any questions or comments about this advisory:

- Open an issue
- Email us at security@opencontainers.org if you think you've found a security bug

37. M-2 golang.org/x/sys/unix has Incorrect privilege reporting in syscall

Tags: `runtime`, Weakness: CWE-269, CVE ID: CVE-2022-29526, GHSA ID: GHSA-p782-xgp4-8hr8

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Golang_Cosmos/go.mod#L249

```
++++ ①
    golang.org/x/sys v0.0.0-20220209214540-3681064d5158 // indirect
++++
```

37.1. CVSS Score: 5.3/10

Table 34. CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N

CVSS base metrics	
Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	Low
Integrity	None
Availability	None

Go before 1.17.10 and 1.18.x before 1.18.2 has Incorrect Privilege Reporting in syscall. When called with a non-zero flags parameter, the Faccessat function could incorrectly report that a file is accessible.

37.2. Specific Go Packages Affected

golang.org/x/sys/unix

38. M-3 runc AppArmor bypass with symlinked /proc

Tags: `runtime`, Weakness: CWE-59, CWE-281, CVE ID: CVE-2023-28642, GHSA ID: GHSA-g2j6-57v7-gm8c

File https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Golang_Cosmos/go.mod#L189

```
++++ ①
    github.com/opencontainers/runc v1.0.3 // indirect
++++
```

38.1. CVSS Score: 6.1/10

Table 35. CVSS:3.1/AV:L/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:L

CVSS base metrics	
Attack vector	Local
Attack complexity	Low
Privileges required	None
User interaction	Required
Scope	Changed
Confidentiality	Low
Integrity	Low
Availability	Low

38.2. Impact

It was found that AppArmor, and potentially SELinux, can be bypassed when `/proc` inside the container is symlinked with a specific mount configuration.

38.3. Patches

Fixed in runc v1.1.5, by prohibiting symlinked `/proc`: https://github.com/opencontainers/runc/pull/3785

This PR fixes CVE-2023-27561 as well.

38.4. Workarounds

Avoid using an untrusted container image.

39. M-4 Docker Swarm encrypted overlay network with a single endpoint is unauthenticated

Tags: `runtime`, Weakness: CWE-420, CWE-636, CVE ID: CVE-2023-28842, GHSA ID: GHSA-6wrf-mxfj-pf5p

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Golang_Cosmos/go.mod#L84

```
++++ ①
    github.com/docker/docker v20.10.7+incompatible // indirect
++++
```

39.1. CVSS Score: 6.8/10

Table 36. CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:C/C:N/I:H/A:N

CVSS base metrics	
Attack vector	Network
Attack complexity	High
Privileges required	None
User interaction	None
Scope	Changed
Confidentiality	None
Integrity	High
Availability	None

Moby is an open source container framework developed by Docker Inc. that is distributed as Docker, Mirantis Container Runtime, and various other downstream projects/products. The Moby daemon component (`dockerd`), which is developed as moby/moby is commonly referred to as **Docker**.

Swarm Mode, which is compiled in and delivered by default in `dockerd` and is thus present in most major Moby downstreams, is a simple, built-in container orchestrator that is implemented through a combination of SwarmKit and supporting network code.

The `overlay` network driver is a core feature of Swarm Mode, providing isolated virtual LANs that allow communication between containers and services across the cluster. This driver is an implementation/user of VXLAN, which encapsulates link-layer (Ethernet) frames in UDP datagrams that tag the frame with a VXLAN Network ID (VNI) that identifies the originating overlay network. In addition, the overlay network driver supports an optional, off-by-default encrypted mode, which is especially useful when VXLAN packets traverses an untrusted network between nodes.

Encrypted overlay networks function by encapsulating the VXLAN datagrams through the use of the IPsec Encapsulating Security Payload protocol in Transport mode. By deploying IPSec encapsulation, encrypted overlay networks gain the additional properties of source authentication through cryptographic proof, data integrity through check-summing, and confidentiality through encryption.

When setting an endpoint up on an encrypted overlay network, Moby installs three iptables (Linux kernel firewall) rules that enforce both incoming and outgoing IPSec. These rules rely on the `u32` iptables extension provided by the `xt_u32` kernel module to directly filter on a VXLAN packet's VNI field, so that IPSec guarantees can be enforced on encrypted overlay networks without interfering with other overlay networks or other users of VXLAN.

The `overlay` driver dynamically and lazily defines the kernel configuration for the VXLAN network on each node as containers are attached and detached. Routes and encryption parameters are only defined for destination nodes that participate in the network. The iptables rules that prevent encrypted overlay networks from accepting unencrypted packets are not created until a peer is available with which to communicate.

39.2. Impact

Encrypted overlay networks silently accept cleartext VXLAN datagrams that are tagged with the VNI of an encrypted overlay network. As a result, it is possible to inject arbitrary Ethernet frames into the encrypted overlay network by encapsulating them in VXLAN datagrams. The implications of this can be quite dire, and GHSA-vwm3-crmr-xfxw should be referenced for a deeper exploration.

39.3. Patches

Patches are available in Moby releases 23.0.3, and 20.10.24. As Mirantis Container Runtime's 20.10 releases are numbered differently, users of that platform should update to 20.10.16.

39.4. Workarounds

- In multi-node clusters, deploy a global 'pause' container for each encrypted overlay network, on every node. For example, use the `registry.k8s.io/pause` image and a `--mode global` service.
- For a single-node cluster, do not use overlay networks of any sort. Bridge networks provide the same connectivity on a single node and have no multi-node features. The Swarm ingress feature is implemented using an overlay network, but can be disabled by publishing ports in `host` mode instead of `ingress` mode (allowing the use of an external load balancer), and removing the `ingress` network.
- If encrypted overlay networks are in exclusive use, block UDP port 4789 from traffic that has not been validated by IPSec. For example, `iptables -A INPUT -m udp —-dport 4789 -m policy --dir in --pol none -j DROP`.

39.5. Background

- This issue was discovered while characterizing and mitigating CVE-2023-28840 and CVE-2023-28841.

39.6. Related

- CVE-2023-28841: Encrypted overlay network traffic may be unencrypted
- CVE-2023-28840: Encrypted overlay network may be unauthenticated
- GHSA-vwm3-crmr-xfxw: The Swarm VXLAN port may be exposed to attack due to ambiguous documentation
- GHSA-gvm4-2qqg-m333: Security issues in encrypted overlay networks (libnetwork)

40. M-5 Docker Swarm encrypted overlay network traffic may be unencrypted

Tags: `runtime`, Weakness: CWE-311, CWE-636, CVE ID: CVE-2023-28841, GHSA ID: GHSA-33pg-m6jh-5237

File https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Golang_Cosmos/go.mod#L84

```
++++ ①
    github.com/docker/docker v20.10.7+incompatible // indirect
++++
```

40.1. CVSS Score: 6.8/10

Table 37. CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:C/C:H/I:N/A:N

CVSS base metrics	
Attack vector	Network
Attack complexity	High
Privileges required	None
User interaction	None
Scope	Changed
Confidentiality	High
Integrity	None
Availability	None

Moby is an open source container framework developed by Docker Inc. that is distributed as Docker, Mirantis Container Runtime, and various other downstream projects/products. The Moby daemon component (`dockerd`), which is developed as moby/moby is commonly referred to as **Docker**.

Swarm Mode, which is compiled in and delivered by default in `dockerd` and is thus present in most major Moby downstreams, is a simple, built-in container orchestrator that is implemented through a combination of SwarmKit and supporting network code.

The `overlay` network driver is a core feature of Swarm Mode, providing isolated virtual LANs that allow communication between containers and services across the cluster. This driver is an implementation/user of VXLAN, which encapsulates link-layer (Ethernet) frames in UDP datagrams that tag the frame with a VXLAN Network ID (VNI) that identifies the originating overlay network. In addition, the overlay network driver supports an optional, off-by-default encrypted mode, which is especially useful when VXLAN packets traverses an untrusted network between nodes.

Encrypted overlay networks function by encapsulating the VXLAN datagrams through the use of the IPsec Encapsulating Security Payload protocol in Transport mode. By deploying IPSec encapsulation, encrypted overlay networks gain the additional properties of source authentication through cryptographic proof, data integrity through check-summing, and confidentiality through encryption.

When setting an endpoint up on an encrypted overlay network, Moby installs three iptables (Linux kernel firewall) rules that enforce both incoming and outgoing IPSec. These rules rely on the `u32` iptables extension provided by the `xt_u32` kernel module to directly filter on a VXLAN packet's VNI field, so that IPSec guarantees can be enforced on encrypted overlay networks without interfering with other overlay networks or other users of VXLAN.

An iptables rule designates outgoing VXLAN datagrams with a VNI that corresponds to an encrypted overlay network for IPsec encapsulation.

On Red Hat Enterprise Linux and derivatives such as CentOS and Rocky, the `xt_u32` module has been: * moved to the kernel-modules-extra package and no longer installed by default in RHEL 8.3 * officially deprecated in RHEL 8.6 * removed completely in RHEL 9

This rule is not created when `xt_u32` is unavailable, even though the container is still attached to the network.

40.2. Impact

Encrypted overlay networks on affected platforms silently transmit unencrypted data. As a result, `overlay` networks may appear to be functional, passing traffic as expected, but without any of the expected confidentiality or data integrity guarantees.

It is possible for an attacker sitting in a trusted position on the network to read all of the application traffic that is moving across the overlay network, resulting in unexpected secrets or user data disclosure. Thus, because many database protocols, internal APIs, etc. are not protected by a second layer of encryption, a user may rely on Swarm encrypted overlay networks to provide confidentiality, which due to this vulnerability is no longer guaranteed.

40.3. Patches

Patches are available in Moby releases 23.0.3, and 20.10.24. As Mirantis Container Runtime's 20.10 releases are numbered differently, users of that platform should update to 20.10.16.

40.4. Workarounds

- Close the VXLAN port (by default, UDP port 4789) to outgoing traffic at the Internet boundary (see GHSA-vwm3-crmr-xfxw) in order to prevent unintentionally leaking unencrypted traffic over the Internet.
- Ensure that the `xt_u32` kernel module is available on all nodes of the Swarm cluster.

40.5. Background

- #43382 partially discussed this concern, but did not consider the security implications.
- Mirantis FIELD-5788 essentially duplicates #43382, and was created six months earlier; it similarly overlooked the security implications.
- #45118 is the ancestor of the final patches, and was where the security implications were discovered.

40.6. Related

- CVE-2023-28840: Encrypted overlay network may be unauthenticated
- CVE-2023-28842: Encrypted overlay network with a single endpoint is unauthenticated

- GHSA-vwm3-crmr-xfxw: The Swarm VXLAN port may be exposed to attack due to ambiguous documentation
- GHSA-gvm4-2qqg-m333: Security issues in encrypted overlay networks (libnetwork)

41. M-6 Cosmos-SDK Cosmovisor component may be vulnerable to denial of service

Tags: `runtime`, GHSA ID: GHSA-23px-mw2p-46qm

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Golang_Cosmos/go.mod#L6

```
++++ ①
    github.com/cosmos/cosmos-sdk v0.45.4
++++
```

Component: Cosmovisor **Criticality**: Medium **Affected Versions**: Cosmovisor < v1.0.0 (distributed with Cosmos-SDK < 0.46) **Affected Users**: Validators and Node operators utilizing unsupported versions of Cosmovisor **Impact**: DOS, potential RCE on node depending on configuration

An issue has been identified on unsupported versions of Cosmovisor which may result in a Denial of Service or Remote Code Execution path depending on configuration for a node or validator using the vulnerable version to manage their node.

If a validator is utilizing an affected version of Cosmovisor with `DAEMON_ALLOW_DOWNLOAD_BINARIES` set to true, a non-default configuration, it may be possible for an attacker to trigger a Remote Code Execution path as well on the host. In this configuration it is recommended to immediately stop use of the `DAEMON_ALLOW_DOWNLOAD_BINARIES` feature, and then proceed with an upgrade of Cosmovisor.

It is recommended that all validators utilizing unsupported versions of Cosmovisor to upgrade to the latest supported versions immediately. If you are utilizing a forked version of Cosmos-SDK, it is recommended to stop use of Cosmovisor until it is possible to update to a supported version of Cosmovisor, whether through your project's fork, or directly compiled from the Cosmos-SDK. At the time of this advisory, the latest version of Cosmovisor is v1.5.0.

Additionally, the Amulet team recommends that developers building chains powered by Cosmos-SDK share this advisory with validators and node operators to ensure this information is available to all impacted parties within their ecosystems.

For more information about Cosmovisor, see https://docs.cosmos.network/main/tooling/cosmovisor

This issue was discovered by Maxwell Dulin and Nathan Kirkland, who reported it to the Cosmos Bug Bounty Program. If you believe you have found a bug in the Interchain Stack or would like to contribute to the program by reporting a bug, please see https://hackerone.com/cosmos.

41.1. How to tell if I am affected?

Running the following command will output whether your cosmovisor version is vulnerable to this issue or not.

Vulnerable to this issue:

```
strings ./cosmovisor | grep -q "NEEDED at" && echo "vulnerable" || echo
"NOT vulnerable"
```

```
vulnerable
```

NOT vulnerable to this issue:

```
strings ./cosmovisor_new | grep -q "NEEDED at" && echo "vulnerable" ||
echo "NOT vulnerable"

NOT vulnerable
```

A Note from Amulet on the Security Advisory Process

In the interest of timely resolution of this issue for validators and node operators, the Amulet team has chosen to use existing processes and resources for distributing security advisories within the Cosmos and Interchain Ecosystems. Stay tuned as we implement an improved, more robust security advisory distribution system that will provide equitable access to information about security issues in the Interchain Stack.

42. M-7 Improper rendering of text nodes in golang.org/x/net/html

Tags: `runtime`, Weakness: CWE-79, CVE ID: CVE-2023-3978, GHSA ID: GHSA-2wrh-6pvc-2jm9

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Golang_Cosmos/go.mod#L247

```
++++ ①
    golang.org/x/net v0.0.0-20211208012354-db4efeb81f4b // indirect
++++
```

42.1. CVSS Score: 6.1/10

Table 38. CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N

CVSS base metrics	
Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	Required
Scope	Changed
Confidentiality	Low
Integrity	Low
Availability	None

Text nodes not in the HTML namespace are incorrectly literally rendered, causing text which should be escaped to not be. This could lead to an XSS attack.

43. M-8 HTTP/2 Stream Cancellation Attack

Tags: `runtime`, Weakness: CWE-400, CVE ID: CVE-2023-44487, GHSA ID: GHSA-qppj-fm5r-hxr3

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Golang_Cosmos/go.mod#L247

```
++++ ①
    golang.org/x/net v0.0.0-20211208012354-db4efeb81f4b // indirect
++++
```

43.1. CVSS Score: 5.3/10

Table 39. CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:L

CVSS base metrics	
Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	None
Integrity	None
Availability	Low

43.2. HTTP/2 Rapid reset attack

The HTTP/2 protocol allows clients to indicate to the server that a previous stream should be canceled by sending a RST_STREAM frame. The protocol does not require the client and server to coordinate the cancellation in any way, the client may do it unilaterally. The client may also assume that the cancellation will take effect immediately when the server receives the RST_STREAM frame, before any other data from that TCP connection is processed.

Abuse of this feature is called a Rapid Reset attack because it relies on the ability for an endpoint to send a RST_STREAM frame immediately after sending a request frame, which makes the other endpoint start working and then rapidly resets the request. The request is canceled, but leaves the HTTP/2 connection open.

The HTTP/2 Rapid Reset attack built on this capability is simple: The client opens a large number of streams at once as in the standard HTTP/2 attack, but rather than waiting for a response to each request stream from the server or proxy, the client cancels each request immediately.

The ability to reset streams immediately allows each connection to have an indefinite number of requests in flight. By explicitly canceling the requests, the attacker never exceeds the limit on the number of concurrent open streams. The number of in-flight requests is no longer dependent on the round-trip time (RTT), but only on the available network bandwidth.

In a typical HTTP/2 server implementation, the server will still have to do significant amounts of work for canceled requests, such as allocating new stream data structures, parsing the query and doing header decompression, and mapping the URL to a resource. For reverse proxy implementations, the request may be proxied to the backend server before the RST_STREAM frame is processed. The client on the other hand paid almost no costs for sending the requests. This creates an exploitable cost asymmetry between the server and the client.

Multiple software artifacts implementing HTTP/2 are affected. This advisory was originally ingested from the `swift-nio-http2` repo advisory and their original conent follows.

43.3. swift-nio-http2 specific advisory

swift-nio-http2 is vulnerable to a denial-of-service vulnerability in which a malicious client can create and then reset a large number of HTTP/2 streams in a short period of time. This causes swift-nio-http2 to commit to a large amount of expensive work which it then throws away, including creating entirely new `` Channel`s to serve the traffic. This can easily overwhelm an `EventLoop `` and prevent it from making forward progress.

swift-nio-http2 1.28 contains a remediation for this issue that applies reset counter using a sliding window. This constrains the number of stream resets that may occur in a given window of time. Clients violating this limit will have their connections torn down. This allows clients to continue to cancel streams for legitimate reasons, while constraining malicious actors.

44. M-9 HTTP/2 Stream Cancellation Attack

Tags: `runtime`, Weakness: CWE-400, CVE ID: CVE-2023-44487, GHSA ID: GHSA-qppj-fm5r-hxr3

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Golang_Cosmos/go.mod#L28

```
++++ ①
    google.golang.org/grpc v1.45.0
++++
```

44.1. CVSS Score: 5.3/10

Table 40. CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:L

CVSS base metrics	
Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	None
Integrity	None
Availability	Low

44.2. HTTP/2 Rapid reset attack

The HTTP/2 protocol allows clients to indicate to the server that a previous stream should be canceled by sending a RST_STREAM frame. The protocol does not require the client and server to coordinate the cancellation in any way, the client may do it unilaterally. The client may also assume that the cancellation will take effect immediately when the server receives the RST_STREAM frame, before any other data from that TCP connection is processed.

Abuse of this feature is called a Rapid Reset attack because it relies on the ability for an endpoint to send a RST_STREAM frame immediately after sending a request frame, which makes the other endpoint start working and then rapidly resets the request. The request is canceled, but leaves the HTTP/2 connection open.

The HTTP/2 Rapid Reset attack built on this capability is simple: The client opens a large number of streams at once as in the standard HTTP/2 attack, but rather than waiting for a response to each request stream from the server or proxy, the client cancels each request immediately.

The ability to reset streams immediately allows each connection to have an indefinite number of requests in flight. By explicitly canceling the requests, the attacker never exceeds the limit on the number of concurrent open streams. The number of in-flight requests is no longer dependent on the round-trip time (RTT), but only on the available network bandwidth.

In a typical HTTP/2 server implementation, the server will still have to do significant amounts of work for canceled requests, such as allocating new stream data structures, parsing the query and doing header decompression, and mapping the URL to a resource. For reverse proxy implementations, the request may be proxied to the backend server before the RST_STREAM frame is processed. The client on the other hand paid almost no costs for sending the requests. This creates an exploitable cost asymmetry between the server and the client.

Multiple software artifacts implementing HTTP/2 are affected. This advisory was originally ingested from the `swift-nio-http2` repo advisory and their original conent follows.

44.3. swift-nio-http2 specific advisory

swift-nio-http2 is vulnerable to a denial-of-service vulnerability in which a malicious client can create and then reset a large number of HTTP/2 streams in a short period of time. This causes swift-nio-http2 to commit to a large amount of expensive work which it then throws away, including creating entirely new `` Channel`s to serve the traffic. This can easily overwhelm an `EventLoop `` and prevent it from making forward progress.

swift-nio-http2 1.28 contains a remediation for this issue that applies reset counter using a sliding window. This constrains the number of stream resets that may occur in a given window of time. Clients violating this limit will have their connections torn down. This allows clients to continue to cancel streams for legitimate reasons, while constraining malicious actors.

45. M-10 Prefix Truncation Attack against ChaCha20-Poly1305 and Encrypt-then-MAC aka Terrapin

Tags: `runtime`, Weakness: CWE-345, CWE-354, CVE ID: CVE-2023-48795, GHSA ID: GHSA-45x7-px36-x8w8

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Golang_Cosmos/go.mod#L245

```
++++ ①
    golang.org/x/crypto v0.0.0-20220214200702-86341886e292 // indirect
++++
```

45.1. CVSS Score: 5.9/10

Table 41. CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:H/A:N

CVSS base metrics	
Attack vector	Network
Attack complexity	High
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	None
Integrity	High
Availability	None

45.2. Summary

Terrapin is a prefix truncation attack targeting the SSH protocol. More precisely, Terrapin breaks the integrity of SSH's secure channel. By carefully adjusting the sequence numbers during the handshake, an attacker can remove an arbitrary amount of messages sent by the client or server at the beginning of the secure channel without the client or server noticing it.

45.3. Mitigations

To mitigate this protocol vulnerability, OpenSSH suggested a so-called "strict kex" which alters the SSH handshake to ensure a Man-in-the-Middle attacker cannot introduce unauthenticated messages as well as convey sequence number manipulation across handshakes.

Warning: To take effect, both the client and server must support this countermeasure.

As a stop-gap measure, peers may also (temporarily) disable the affected algorithms and use unaffected alternatives like AES-GCM instead until patches are available.

45.4. Details

The SSH specifications of ChaCha20-Poly1305 (chacha20-poly1305@openssh.com) and Encrypt-then-MAC (*-etm@openssh.com MACs) are vulnerable against an arbitrary prefix truncation attack (a.k.a. Terrapin attack). This allows for an extension negotiation downgrade by stripping the SSH_MSG_EXT_INFO sent after the first message after SSH_MSG_NEWKEYS, downgrading security, and disabling attack countermeasures in some versions of OpenSSH. When targeting Encrypt-then-MAC, this attack requires the use of a CBC cipher to be practically exploitable due to the internal workings of the cipher mode. Additionally, this novel attack technique can be used to exploit previously unexploitable implementation flaws in a Man-in-the-Middle scenario.

The attack works by an attacker injecting an arbitrary number of SSH_MSG_IGNORE messages during the initial key exchange and consequently removing the same number of messages just after the initial key exchange has concluded. This is possible due to missing authentication of the excess SSH_MSG_IGNORE messages and the fact that the implicit sequence numbers used within the SSH protocol are only checked after the initial key exchange.

In the case of ChaCha20-Poly1305, the attack is guaranteed to work on every connection as this cipher does not maintain an internal state other than the message’s sequence number. In the case of Encrypt-Then-MAC, practical exploitation requires the use of a CBC cipher; while theoretical integrity is broken for all ciphers when using this mode, message processing will fail at the application layer for CTR and stream ciphers.

For more details see https://terrapin-attack.com.

45.5. Impact

This attack targets the specification of ChaCha20-Poly1305 (chacha20-poly1305@openssh.com) and Encrypt-then-MAC (*-etm@openssh.com), which are widely adopted by well-known SSH implementations and can be considered de-facto standard. These algorithms can be practically exploited; however, in the case of Encrypt-Then-MAC, we additionally require the use of a CBC cipher. As a consequence, this attack works against all well-behaving SSH implementations supporting either of those algorithms and can be used to downgrade (but not fully strip) connection security in case SSH extension negotiation (RFC8308) is supported. The attack may also enable attackers to exploit certain implementation flaws in a man-in-the-middle (MitM) scenario.

46. M-11 Denial of service when decrypting attack controlled input in github.com/dvsekhvalnov/jose2go

Tags: `runtime`, Weakness: CWE-400, GHSA ID: GHSA-mhpq-9638-x6pw

File https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Golang_Cosmos/go.mod#L88

```
++++ ①
    github.com/dvsekhvalnov/jose2go v0.0.0-20200901110807-248326c1351b
// indirect
++++
```

46.1. CVSS Score: 5.3/10

Table 42. CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:L

CVSS base metrics	
Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	None
Integrity	None
Availability	Low

An attacker controlled input of a PBES2 encrypted JWE blob can have a very large p2c value that, when decrypted, produces a denial-of-service.

47. M-12 /sys/devices/virtual/powercap accessible by default to containers

Tags: `runtime`, GHSA ID: GHSA-jq35-85cj-fj4p

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Golang_Cosmos/go.mod#L84

```
++++ ①
    github.com/docker/docker v20.10.7+incompatible // indirect
++++
```

Intel's RAPL (Running Average Power Limit) feature, introduced by the Sandy Bridge microarchitecture, provides software insights into hardware energy consumption. To facilitate this, Intel introduced the powercap framework in Linux kernel 3.13, which reads values via relevant MSRs (model specific registers) and provides unprivileged userspace access via `sysfs`. As RAPL is an interface to access a hardware feature, it is only available when running on bare metal with the module compiled into the kernel.

By 2019, it was realized that in some cases unprivileged access to RAPL readings could be exploited as a power-based side-channel against security features including AES-NI (potentially inside a SGX enclave) and KASLR (kernel address space layout randomization). Also known as the PLATYPUS attack, Intel assigned CVE-2020-8694 and CVE-2020-8695, and AMD assigned CVE-2020-12912.

Several mitigations were applied; Intel reduced the sampling resolution via a microcode update, and the Linux kernel prevents access by non-root users since 5.10. However, this kernel-based mitigation does not apply to many container-based scenarios: * Unless using user namespaces, root inside a container has the same level of privilege as root outside the container, but with a slightly more narrow view of the system * `sysfs` is mounted inside containers read-only; however only read access is needed to carry out this attack on an unpatched CPU

While this is not a direct vulnerability in container runtimes, defense in depth and safe defaults are valuable and preferred, especially as this poses a risk to multi-tenant container environments running directly on affected hardware. This is provided by masking `/sys/devices/virtual/powercap` in the default mount configuration, and adding an additional set of rules to deny it in the default AppArmor profile.

While `sysfs` is not the only way to read from the RAPL subsystem, other ways of accessing it require additional capabilities such as `CAP_SYS_RAWIO` which is not available to containers by default, or `perf` paranoia level less than 1, which is a non-default kernel tunable.

47.1. References

- https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-8694
- https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-8695
- https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-12912
- https://platypusattack.com/
- https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=949dd0104c496fa7c14991a23c03c62e44637e71
- https://web.eece.maine.edu/~vweaver/projects/rapl/

48. M-13 ASA-2024-002: Default `PrepareProposalHandler` may produce invalid proposals when used with default `SenderNonceMempool`

Tags: `runtime`, Weakness: CWE-1285, GHSA ID: GHSA-2557-x9mg-76w8

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Golang_Cosmos/go.mod#L6

```
++++ ①
    github.com/cosmos/cosmos-sdk v0.45.4
++++
```

48.1. CVSS Score: 5.3/10

Table 43. CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:L

CVSS base metrics	
Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	None
Integrity	None
Availability	Low

48.2. ASA-2024-002: Default `PrepareProposalHandler` may produce invalid proposals when used with default `SenderNonceMempool`

Component: Cosmos SDK **Criticality**: Medium **Affected** Versions: Cosmos SDK versions ¬ 0.50.3; ¬ 0.47.8 **Affected** Users: Chain developers, Validator and Node operators **Impact**: Denial of Service

48.3. Summary

When using the default `PrepareProposalHandler` and the default `SenderNonceMempool`, an issue was identified which may allow invalid blocks to be proposed when a single sender includes multiple transactions with non-sequential sequence numbers in certain conditions. If this state is reached, it can lead to a reduction in block production for a network.

48.4. Next Steps for Impacted Parties

If you are a chain developer on an affected version of the Cosmos SDK, it is advised to update to the latest available version of the Cosmos SDK for your project. Once a patched version is available, it is recommended that network operators upgrade.

A Github Security Advisory for this issue is available in the Cosmos-SDK repository. For more information about Cosmos SDK, see https://docs.cosmos.network/.

This issue was found by KonradStaniec, gitferry, SebastianElvis, and vitsalis who reported it to the Cosmos Bug Bounty Program on HackerOne on January 16, 2024. If you believe you have found a bug in the Interchain Stack or would like to contribute to the program by reporting a bug, please see https://hackerone.com/cosmos.

49. M-14 jose2go vulnerable to denial of service via large p2c value

Tags: `runtime`, CVE ID: CVE-2023-50658, GHSA ID: GHSA-6294-6rgp-fr7r

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Golang_Cosmos/go.mod#L88

```
++++ ①
    github.com/dvsekhvalnov/jose2go v0.0.0-20200901110807-248326c1351b
// indirect
++++
```

The jose2go component before 1.6.0 for Go allows attackers to cause a denial of service (CPU consumption) via a large p2c (aka PBES2 Count) value.

50. M-15 Golang protojson.Unmarshal function infinite loop when unmarshaling certain forms of invalid JSON

Tags: `runtime`, Weakness: CWE-835, CVE ID: CVE-2024-24786, GHSA ID: GHSA-8r3f-844c-mc37

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Golang_Cosmos/go.mod#L29

```
++++ ①
    google.golang.org/protobuf v1.27.1
++++
```

The protojson.Unmarshal function can enter an infinite loop when unmarshaling certain forms of invalid JSON. This condition can occur when unmarshaling into a message which contains a google.protobuf.Any value, or when the UnmarshalOptions.DiscardUnknown option is set.

51. M-16 Classic builder cache poisoning

Tags: `runtime`, Weakness: CWE-345, CWE-346, CVE ID: CVE-2024-24557, GHSA ID: GHSA-xw73-rw38-6vjc

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Golang_Cosmos/go.mod#L84

```
++++ ①
    github.com/docker/docker v20.10.7+incompatible // indirect
++++
```

51.1. CVSS Score: 6.9/10

Table 44. CVSS:3.1/AV:L/AC:H/PR:N/UI:R/S:C/C:L/I:H/A:L

CVSS base metrics	
Attack vector	Local
Attack complexity	High
Privileges required	None
User interaction	Required
Scope	Changed
Confidentiality	Low
Integrity	High
Availability	Low

The classic builder cache system is prone to cache poisoning if the image is built `FROM scratch`. Also, changes to some instructions (most important being `HEALTHCHECK` and `ONBUILD`) would not cause a cache miss.

An attacker with the knowledge of the Dockerfile someone is using could poison their cache by making them pull a specially crafted image that would be considered as a valid cache candidate for some build steps.

For example, an attacker could create an image that is considered as a valid cache candidate for:

```
FROM scratch
MAINTAINER Pawel
```

when in fact the malicious image used as a cache would be an image built from a different Dockerfile.

In the second case, the attacker could for example substitute a different `HEALTCHECK` command.

51.2. Impact

23.0+ users are only affected if they explicitly opted out of Buildkit (`DOCKER_BUILDKIT=0` environment variable) or are using the `/build` API endpoint (which uses the classic builder by default).

All users on versions older than 23.0 could be impacted. An example could be a CI with a shared cache, or just a regular Docker user pulling a malicious image due to misspelling/typosquatting.

Image build API endpoint (`/build`) and `ImageBuild` function from `github.com/docker/docker/client` is also affected as it the uses classic builder by default.

51.3. Patches

Patches are included in Moby releases:

- v25.0.2
- v24.0.9

51.4. Workarounds

- Use `--no-cache` or use Buildkit if possible (`DOCKER_BUILDKIT=1`, it's default on 23.0+ assuming that the buildx plugin is installed).
- Use `Version = types.BuilderBuildKit` or `NoCache = true` in `ImageBuildOptions` for `ImageBuild` call.

52. M-17 Moby’s external DNS requests from 'internal' networks could lead to data exfiltration

Tags: `runtime`, Weakness: CWE-669, CVE ID: CVE-2024-29018, GHSA ID: GHSA-mq39-4gv4-mvpx

File https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Golang_Cosmos/go.mod#L84

```
++++ ①
    github.com/docker/docker v20.10.7+incompatible // indirect
++++
```

52.1. CVSS Score: 5.9/10

Table 45. CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:N/A:N

CVSS base metrics	
Attack vector	Network
Attack complexity	High
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	High
Integrity	None
Availability	None

Moby is an open source container framework originally developed by Docker Inc. as Docker. It is a key component of Docker Engine, Docker Desktop, and other distributions of container tooling or runtimes. As a batteries-included container runtime, Moby comes with a built-in networking implementation that enables communication between containers, and between containers and external resources.

Moby’s networking implementation allows for creating and using many networks, each with their own subnet and gateway. This feature is frequently referred to as custom networks, as each network can have a different driver, set of parameters, and thus behaviors. When creating a network, the `--internal` flag is used to designate a network as *internal*. The `internal` attribute in a docker-compose.yml file may also be used to mark a network *internal*, and other API clients may specify the `internal` parameter as well.

When containers with networking are created, they are assigned unique network interfaces and IP addresses (typically from a non-routable RFC 1918 subnet). The root network namespace (hereafter referred to as the 'host') serves as a router for non-internal networks, with a gateway IP that provides SNAT/DNAT to/from container IPs.

Containers on an *internal* network may communicate between each other, but are precluded from communicating with any networks the host has access to (LAN or WAN) as no default route is configured, and firewall rules are set up to drop all outgoing traffic. Communication with the gateway IP address (and thus appropriately configured host services) is possible, and the host may communicate with any container IP directly.

In addition to configuring the Linux kernel's various networking features to enable container networking, `dockerd` directly provides some services to container networks. Principal among these is serving as a resolver, enabling service discovery (looking up other containers on the network by name), and resolution of names from an upstream resolver.

When a DNS request for a name that does not correspond to a container is received, the request is forwarded to the configured upstream resolver (by default, the host's configured resolver). This request is made from the container network namespace: the level of access and routing of traffic is the same as if the request was made by the container itself.

As a consequence of this design, containers solely attached to *internal* network(s) will be unable to resolve names using the upstream resolver, as the container itself is unable to communicate with that nameserver. Only the names of containers also attached to the internal network are able to be resolved.

Many systems will run a local forwarding DNS resolver, typically present on a loopback address (`127.0.0.0/8`), such as systemd-resolved or dnsmasq. Common loopback address examples include `127.0.0.1` or `127.0.0.53`. As the host and any containers have separate loopback devices, a consequence of the design described above is that containers are unable to resolve names from the host's configured resolver, as they cannot reach these addresses on the host loopback device.

To bridge this gap, and to allow containers to properly resolve names even when a local forwarding resolver is used on a loopback address, `dockerd` will detect this scenario and instead forward DNS requests from the host/root network namespace. The loopback resolver will then forward the requests to its configured upstream resolvers, as expected.

52.2. Impact

Because `dockerd` will forward DNS requests to the host loopback device, bypassing the container network namespace's normal routing semantics entirely, *internal* networks can unexpectedly forward DNS requests to an external nameserver.

By registering a domain for which they control the authoritative nameservers, an attacker could arrange for a compromised container to exfiltrate data by encoding it in DNS queries that will eventually be answered by their nameservers. For example, if the domain `evil.example` was registered, the authoritative nameserver(s) for that domain could (eventually and indirectly) receive a request for `this-is-a-secret.evil.example`.

Docker Desktop is not affected, as Docker Desktop always runs an internal resolver on a RFC 1918 address.

52.3. Patches

Moby releases 26.0.0-rc3, 25.0.5 (released) and 23.0.11 (to be released) are patched to prevent forwarding DNS requests from internal networks.

52.4. Workarounds

- Run containers intended to be solely attached to *internal* networks with a custom upstream address (`--dns` argument to `docker run`, or API equivalent), which will force all upstream DNS queries to be resolved from the container network namespace.

52.5. Background

- yair zak originally reported this issue to the Docker security team.
- PR https://github.com/moby/moby/pull/46609 was opened in public to fix this issue, as it was not originally considered to have a security implication.
- The official documentation claims that "the `--internal` flag that will completely isolate containers on a network from any communications external to that network," which necessitated this advisory and CVE.

53. M-18 Integer Overflow in Chunked Transfer-Encoding

Tags: `runtime`, Weakness: CWE-190, CVE ID: CVE-2021-32714, GHSA ID: GHSA-5h46-h7hh-c6x9

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L2259

```
++++ ①
[[package]]
name = "hyper"
version = "0.12.36"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"5c843caf6296fc1f93444735205af9ed4e109a539005abb2564ae1d6fad34c52"
++++
```

53.1. CVSS Score: 5.9/10

Table 46. CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:H

CVSS base metrics	
Attack vector	Network
Attack complexity	High
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	None
Integrity	None
Availability	High

53.2. Summary

hyper's HTTP server and client code had a flaw that could trigger an integer overflow when decoding chunk sizes that are too big. This allows possible data loss, or if combined with an upstream HTTP proxy that allows chunk sizes larger than hyper does, can result in "request smuggling" or "desync attacks".

53.3. Vulnerability

Example:

```
GET / HTTP/1.1
Host: example.com
Transfer-Encoding: chunked
```

```
f0000000000000003
abc
0
```

hyper only reads the rightmost 64-bit integer as the chunk size. So it reads `f0000000000000003` as `3`. A loss of data can occur since hyper would then read only 3 bytes of the body. Additionally, an HTTP request smuggling vulnerability would occur if using a proxy which instead has prefix truncation in the chunk size, or that understands larger than 64-bit chunk sizes.

Read more about desync attacks: https://portswigger.net/research/http-desync-attacks-request-smuggling-reborn

53.4. Impact

To determine if vulnerable to *data loss*, these things must be true:

- **Using HTTP/1.1.** Since HTTP/2 does not use chunked encoding, it is not vulnerable.
- **Using hyper as a server or client.** The body would be improperly truncated in either case.
- **Users send requests or responses with chunk sizes greater than 18 exabytes**.

To determine if vulnerable to *desync attacks*, these things must be true:

- **Using an upstream proxy that allows chunks sizes larger than 64-bit.** If the proxy rejects chunk sizes that are too large, that request won't be forwarded to hyper.

53.5. Patches

We have released the following patch versions:

- v0.14.10 (to be released when this advisory is published)

53.6. Workarounds

Besides upgrading hyper, you can take the following options:

- Reject requests manually that contain a `Transfer-Encoding` header.
- Ensure any upstream proxy rejects `Transfer-Encoding` chunk sizes greater than what fits in 64-bit unsigned integers.

53.7. Credits

This issue was initially reported by [Mattias Grenfeldt](#) and Asta Olofsson.

54. M-19 Error on unsupported architectures in raw-cpuid

Tags: `runtime`, Weakness: CWE-400, CWE-657, CVE ID: CVE-2021-26307, GHSA ID: GHSA-jrf8-cmgg-gv2m

File https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L5154

```
++++ ①
[[package]]
name = "raw-cpuid"
version = "8.1.2"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"1fdf7d9dbd43f3d81d94a49c1c3df73cc2b3827995147e6cf7f89d4ec5483e73"
++++
```

54.1. CVSS Score: 5.5/10

Table 47. CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H

CVSS base metrics	
Attack vector	Local
Attack complexity	Low
Privileges required	Low
User interaction	None
Scope	Unchange
Confidentiality	None
Integrity	None
Availability	High

native_cpuid::cpuid_count() exposes the unsafe __cpuid_count() intrinsic from core::arch::x86 or core::arch::x86_64 as a safe function, and uses it internally, without checking the safety requirement:

- The CPU the program is currently running on supports the function being called.

CPUID is available in most, but not all, x86/x86_64 environments. The crate compiles only on these architectures, so others are unaffected. This issue is mitigated by the fact that affected programs are expected to crash deterministically every time.

The flaw has been fixed in v9.0.0, by intentionally breaking compilation when targeting SGX or 32-bit x86 without SSE. This covers all affected CPUs.

55. M-20 Data races in lock_api

Tags: `runtime`, Weakness: CWE-362, CVE ID: CVE-2020-35913, GHSA ID: GHSA-hj9h-wrgg-hgmx

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L3219

```
++++ ①
[[package]]
name = "lock_api"
version = "0.3.4"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"c4da24a77a3d8a6d4862d95f72e6fdb9c09a643ecdb402d754004a557f2bec75"
++++
```

55.1. CVSS Score: 4.7/10

Table 48. CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:N/I:N/A:H

CVSS base metrics	
Attack vector	Local
Attack complexity	High
Privileges required	Low
User interaction	None
Scope	Unchange
Confidentiality	None
Integrity	None
Availability	High

An issue was discovered in the lock_api crate before 0.4.2 for Rust. A data race can occur because of RwLockReadGuard unsoundness.

56. M-21 Data races in lock_api

Tags: `runtime`, Weakness: CWE-362, CVE ID: CVE-2020-35912, GHSA ID: GHSA-5wg8-7c9q-794v

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L3219

```
++++ ①
[[package]]
name = "lock_api"
version = "0.3.4"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"c4da24a77a3d8a6d4862d95f72e6fdb9c09a643ecdb402d754004a557f2bec75"
++++
```

56.1. CVSS Score: 4.7/10

Table 49. CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:N/I:N/A:H

CVSS base metrics	
Attack vector	Local
Attack complexity	High
Privileges required	Low
User interaction	None
Scope	Unchange
Confidentiality	None
Integrity	None
Availability	High

An issue was discovered in the lock_api crate before 0.4.2 for Rust. A data race can occur because of MappedRwLockWriteGuard unsoundness.

57. M-22 Data races in lock_api

Tags: `runtime`, Weakness: CWE-362, CVE ID: CVE-2020-35911, GHSA ID: GHSA-vh4p-6j7g-f4j9

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L3219

```
++++ ①
[[package]]
name = "lock_api"
version = "0.3.4"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"c4da24a77a3d8a6d4862d95f72e6fdb9c09a643ecdb402d754004a557f2bec75"
++++
```

57.1. CVSS Score: 4.7/10

Table 50. CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:N/I:N/A:H

CVSS base metrics	
Attack vector	Local
Attack complexity	High
Privileges required	Low
User interaction	None
Scope	Unchange
Confidentiality	None
Integrity	None
Availability	High

An issue was discovered in the lock_api crate before 0.4.2 for Rust. A data race can occur because of MappedRwLockReadGuard unsoundness.

58. M-23 Data races in lock_api

Tags: `runtime`, Weakness: CWE-362, CVE ID: CVE-2020-35914, GHSA ID: GHSA-gmv4-vmx3-x9f3

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L3219

```
++++ ①
[[package]]
name = "lock_api"
version = "0.3.4"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"c4da24a77a3d8a6d4862d95f72e6fdb9c09a643ecdb402d754004a557f2bec75"
++++
```

58.1. CVSS Score: 4.7/10

Table 51. CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:N/I:N/A:H

CVSS base metrics	
Attack vector	Local
Attack complexity	High
Privileges required	Low
User interaction	None
Scope	Unchange
Confidentiality	None
Integrity	None
Availability	High

An issue was discovered in the lock_api crate before 0.4.2 for Rust. A data race can occur because of RwLockWriteGuard unsoundness.

59. M-24 Data races in lock_api

Tags: `runtime`, Weakness: CWE-362, CVE ID: CVE-2020-35910, GHSA ID: GHSA-ppj3-7jw3-8vc4

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L3219

```
++++ ①
[[package]]
name = "lock_api"
version = "0.3.4"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"c4da24a77a3d8a6d4862d95f72e6fdb9c09a643ecdb402d754004a557f2bec75"
++++
```

59.1. CVSS Score: 5.5/10

Table 52. CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H

CVSS base metrics	
Attack vector	Local
Attack complexity	Low
Privileges required	Low
User interaction	None
Scope	Unchange
Confidentiality	None
Integrity	None
Availability	High

An issue was discovered in the lock_api crate before 0.4.2 for Rust. A data race can occur because of MappedMutexGuard unsoundness.

60. M-25 Invalid drop of partially-initialized instances in the pooling instance allocator for modules with defined `externref` globals

Tags: `runtime`, Weakness: CWE-824, CVE ID: CVE-2022-23636, GHSA ID: GHSA-88xq-w8cq-xfg7

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L8590

```
++++ ①
[[package]]
name = "wasmtime"
version = "0.22.0"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"7426055cb92bd9a1e9469b48154d8d6119cd8c498c8b70284e420342c05dc45d"
++++
```

60.1. CVSS Score: 5.1/10

Table 53. CVSS:3.1/AV:L/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:H

CVSS base metrics	
Attack vector	Local
Attack complexity	High
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	None
Integrity	None
Availability	High

60.2. Impact

There exists a bug in the pooling instance allocator in Wasmtime's runtime where a failure to instantiate an instance for a module that defines an `externref` global will result in an invalid drop of a `VMExternRef` via an uninitialized pointer.

As instance slots may be reused between consecutive instantiations, the value of the uninitialized pointer may be from a previous instantiation and therefore under the control of an attacker via a module's initial values for its globals. If the attacker can somehow determine an address under their control inside the mapped memory representing the instance pool, it is possible to trick the runtime to call `drop_in_place` on a trait object under the attacker's control and therefore cause remote code execution.

Exploiting the bug to cause remote code execution would be very difficult as attackers cannot determine the addresses of globals from code executing within the WebAssembly VM and the memory space for the instance pool cannot be statically determined. Operating system mitigations, such as address space layout randomization, would additionally increase the difficulty for attackers to determine useful executable code to target with an exploit. It is also very unlikely that attackers will be able to directly influence the conditions that trigger the bug as described below.

When the conditions to trigger the bug are met, however, it is much easier to exploit this bug to cause a denial of service by crashing the host with an invalid memory read.

The following engine configuration (via Config) is required to be impacted by this bug:

- support for the reference types proposal must be enabled (this is the default for `Config`).
- a pooling allocation strategy must be configured via Config::allocation_strategy, which is **not the default allocation strategy**.

A module must be instantiated with **all the following characteristics**:

- The module defines at least one table or memory.
- The module defines at least one `externref` global.

During instantiation, **one of the following** must occur to cause the instantiation to fail:

- a call to `mprotect` or `VirtualAlloc` fails (e.g. out-of-memory conditions).
- a resource limiter was configured in the associated `Store` (via Store::limiter or Store::limiter_async) and the limiter returns `false` from the initial call to `memory_growing` or `table_growing`. **Stores do not have a resource limiter set by default**.

This results in a partially-initialized instance being dropped and that attempts to drop the uninitialized `VMExternRef` representing the defined `externref` global.

We have reason to believe that the effective impact of this bug is relatively small because the usage of `externref` is still uncommon and without a resource limiter configured on the `Store`, which is not the default configuration, it is only possible to trigger the bug from an error returned by `mprotect` or `VirtualAlloc`.

Note that on Linux with the `uffd` feature enabled, it is only possible to trigger the bug from a resource limiter as the call to `mprotect` is skipped; if no resource limiter is used, then this configuration is not vulnerable.

60.3. Patches

The bug has been fixed in 0.34.1 and 0.33.1; users are encouraged to upgrade as soon as possible.

60.4. Workarounds

If it is not possible to upgrade to 0.34.1 or 0.33.1 of the `wasmtime` crate, it is recommend that support for the reference types proposal be disabled by passing `false` to Config::wasm_reference_types.

Doing so will prevent modules that use `externref` from being loaded entirely.

60.5. For more information

If you have any questions or comments about this advisory:

- Reach out to us on the Bytecode Alliance Zulip chat
- Open an issue in the bytecodealliance/wasmtime repository

61. M-26 Wrong type for Linker-define functions when used across two `Engine`s

Tags: runtime, Weakness: CWE-843, CVE ID: CVE-2021-39219, GHSA ID: GHSA-q879-9g95-56mx

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L8590

```
++++ ①
[[package]]
name = "wasmtime"
version = "0.22.0"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"7426055cb92bd9a1e9469b48154d8d6119cd8c498c8b70284e420342c05dc45d"
++++
```

61.1. CVSS Score: 6.3/10

Table 54. CVSS:3.1/AV:L/AC:H/PR:N/UI:R/S:U/C:N/I:H/A:H

CVSS base metrics	
Attack vector	Local
Attack complexity	High
Privileges required	None
User interaction	Required
Scope	Unchange
Confidentiality	None
Integrity	High
Availability	High

61.2. Impact

As a Rust library the `wasmtime` crate clearly marks which functions are safe and which are `unsafe`, guaranteeing that if consumers never use `unsafe` then it should not be possible to have memory unsafety issues in their embeddings of Wasmtime. An issue was discovered in the safe API of `Linker::func_*` APIs. These APIs were previously not sound when one `Engine` was used to create the `Linker` and then a different `Engine` was used to create a `Store` and then the `Linker` was used to instantiate a module into that `Store`. Cross-`Engine` usage of functions is not supported in Wasmtime and this can result in type confusion of function pointers, resulting in being able to safely call a function with the wrong type.

Triggering this bug requires using at least two `Engine` values in an embedding and then additionally using two different values with a `Linker` (one at the creation time of the `Linker` and another when instantiating a module with the `Linker`).

It's expected that usage of more-than-one `Engine` in an embedding is relatively rare since an `Engine` is intended to be a globally shared resource, so the expectation is that the impact of this issue is relatively small.

The fix implemented is to change this behavior to `panic!()` in Rust instead of silently allowing it. Using different `Engine` instances with a `Linker` is a programmer bug that `wasmtime` catches at runtime.

61.3. Patches

This bug has been patched and users should upgrade to Wasmtime version 0.30.0.

61.4. Workarounds

If you cannot upgrade Wasmtime and are using more than one `Engine` in your embedding it's recommended to instead use only one `Engine` for the entire program if possible. An `Engine` is designed to be a globally shared resource that is suitable to have only one for the lifetime of an entire process. If using multiple `` Engine`s is required then code should be audited to ensure that `Linker `` is only used with one `Engine`.

61.5. For more information

If you have any questions or comments about this advisory:

- Reach out to us on the Bytecode Alliance Zulip chat
- Open an issue in the `bytecodealliance/wasmtime` repository

62. M-27 Miscompilation of `i8x16.swizzle` and `select` with v128 inputs

Tags: `runtime`, Weakness: CWE-682, CVE ID: CVE-2022-31104, GHSA ID: GHSA-jqwc-c49r-4w2x

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L8590

```
++++ ①
[[package]]
name = "wasmtime"
version = "0.22.0"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"7426055cb92bd9a1e9469b48154d8d6119cd8c498c8b70284e420342c05dc45d"
++++
```

62.1. CVSS Score: 4.8/10

Table 55. CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:L/A:L

CVSS base metrics	
Attack vector	Network
Attack complexity	High
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	None
Integrity	Low
Availability	Low

62.2. Impact

Wasmtime's implementation of the SIMD proposal for WebAssembly on x86_64 contained two distinct bugs in the instruction lowerings implemented in Cranelift. The aarch64 implementation of the simd proposal is not affected. The bugs were presented in the `i8x16.swizzle` and `select` WebAssembly instructions. The `select` instruction is only affected when the inputs are of `v128` type. The correspondingly affected Cranelift instructions were `swizzle` and `select`.

The `swizzle` instruction lowering in Cranelift erroneously overwrote the mask input register which could corrupt a constant value, for example. This means that future uses of the same constant may see a different value than the constant itself.

The `select` instruction lowering in Cranelift wasn't correctly implemented for vector types that are 128-bits wide. When the condition was 0 the wrong instruction was used to move the correct input to the output of the instruction meaning that only the low 32 bits were moved and the upper 96 bits of the result were left as whatever the register previously contained (instead of the input being moved from). The `select` instruction worked correctly if the condition was nonzero, however.

This bug in Wasmtime's implementation of these instructions on x86_64 represents an incorrect implementation of the specified semantics of these instructions according to the WebAssembly specification. The impact of this is benign for hosts running WebAssembly but represents possible vulnerabilities within the execution of a guest program. For example a WebAssembly program could take unintended branches or materialize incorrect values internally which runs the risk of exposing the program itself to other related vulnerabilities which can occur from miscompilations.

62.3. Patches

We have released Wasmtime 0.38.1 and cranelift-codegen (and other associated cranelift crates) 0.85.1 which contain the corrected implementations of these two instructions in Cranelift.

62.4. Workarounds

If upgrading is not an option for you at this time, you can avoid the vulnerability by disabling the Wasm simd proposal

```
config.wasm_simd(false);
```

Additionally the bug is only present on x86_64 hosts. Other aarch64 hosts are not affected. Note that s390x hosts don't yet implement the simd proposal and are not affected.

62.5. References

- The WebAssembly simd proposal
- Original test case showing the erroneous behavior
- Fix for the `swizzle` instruction
- Fix for the `select` instruction

62.6. For more information

If you have any questions or comments about this advisory:

- Reach out to us on the Bytecode Alliance Zulip chat
- Open an issue in the bytecodealliance/wasmtime repository

63. M-28 Cranelift vulnerable to miscompilation of constant values in division on AArch64

Tags: `runtime`, Weakness: CWE-682, CVE ID: CVE-2022-31169, GHSA ID: GHSA-7f6x-jwh5-m9r4

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L8590

```
++++ ①
[[package]]
name = "wasmtime"
version = "0.22.0"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"7426055cb92bd9a1e9469b48154d8d6119cd8c498c8b70284e420342c05dc45d"
++++
```

63.1. CVSS Score: 5.9/10

Table 56. CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:H/A:N

CVSS base metrics	
Attack vector	Network
Attack complexity	High
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	None
Integrity	High
Availability	None

63.2. Impact

There was a bug in Wasmtime's code generator, Cranelift, for AArch64 targets where constant divisors could result in incorrect division results at runtime. The translation rules for constants did not take into account whether sign- or zero-extension should happen, which resulted in an incorrect value being placed into a register when a division was encountered. For example, a constant 32-bit unsigned divisor of `0xfffffffe` would be incorrectly sign-extended to 64-bits to `0xfffffffffffffffe`. Any kind of division of operands smaller than 64 bits is implemented with a 64-bit division instruction which would then result in an incorrect result because the divisor was larger than expected.

The impact of this bug is that programs executing within the WebAssembly sandbox would not behave according to the WebAssembly specification. This means that it is hypothetically possible for execution within the sandbox to go awry and WebAssembly programs could produce unexpected results. This should not impact hosts executing WebAssembly, but does affect the correctness of guest programs.

This bug was found with differential fuzzing of Wasmtime against other engines on the AArch64 platform. Fuzzing on AArch64 is not regularly performed at this time and the Wasmtime team is investigating how best to continuously fuzz AArch64 in the same manner as x86_64.

63.3. Patches

This bug has been patched and users should upgrade to Wasmtime version 0.38.2.

63.4. Workarounds

If upgrading is not an option at this time, direct users of Cranelift that control the exact Cranelift instructions being compiled can avoid the vulnerability by explicitly extending constant divisors to 64 bits using either the `sextend.i64` or the `uextend.i64` operation.

Note, though, that this issue only affects the AArch64 targets. Other platforms are not affected.

63.5. For more information

If you have any questions or comments about this advisory:

- Reach out to us on the Bytecode Alliance Zulip chat
- Open an issue in the bytecodealliance/wasmtime repository

64. M-29 Cranelift vulnerable to miscompilation of constant values in division on AArch64

Tags: `runtime`, Weakness: CWE-682, CVE ID: CVE-2022-31169, GHSA ID: GHSA-7f6x-jwh5-m9r4

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L901

```
++++ ①
[[package]]
name = "cranelift-codegen"
version = "0.69.0"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"1a54e4beb833a3c873a18a8fe735d73d732044004c7539a072c8faa35ccb0c60"
++++
```

64.1. CVSS Score: 5.9/10

Table 57. CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:H/A:N

CVSS base metrics	
Attack vector	Network
Attack complexity	High
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	None
Integrity	High
Availability	None

64.2. Impact

There was a bug in Wasmtime's code generator, Cranelift, for AArch64 targets where constant divisors could result in incorrect division results at runtime. The translation rules for constants did not take into account whether sign- or zero-extension should happen, which resulted in an incorrect value being placed into a register when a division was encountered. For example, a constant 32-bit unsigned divisor of `0xfffffffe` would be incorrectly sign-extended to 64-bits to `0xfffffffffffffffe`. Any kind of division of operands smaller than 64 bits is implemented with a 64-bit division instruction which would then result in an incorrect result because the divisor was larger than expected.

The impact of this bug is that programs executing within the WebAssembly sandbox would not behave according to the WebAssembly specification. This means that it is hypothetically possible for execution within the sandbox to go awry and WebAssembly programs could produce unexpected results. This should not impact hosts executing WebAssembly, but does affect the correctness of guest programs.

This bug was found with differential fuzzing of Wasmtime against other engines on the AArch64 platform. Fuzzing on AArch64 is not regularly performed at this time and the Wasmtime team is investigating how best to continuously fuzz AArch64 in the same manner as x86_64.

64.3. Patches

This bug has been patched and users should upgrade to Wasmtime version 0.38.2.

64.4. Workarounds

If upgrading is not an option at this time, direct users of Cranelift that control the exact Cranelift instructions being compiled can avoid the vulnerability by explicitly extending constant divisors to 64 bits using either the `sextend.i64` or the `uextend.i64` operation.

Note, though, that this issue only affects the AArch64 targets. Other platforms are not affected.

64.5. For more information

If you have any questions or comments about this advisory:

- Reach out to us on the Bytecode Alliance Zulip chat
- Open an issue in the bytecodealliance/wasmtime repository

65. M-30 owning_ref vulnerable to multiple soundness issues

Tags: `runtime`, GHSA ID: GHSA-9qxh-258v-666c

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L3891

```
++++ ①
[[package]]
name = "owning_ref"
version = "0.4.1"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"6ff55baddef9e4ad00f88b6c743a2a8062d4c6ade126c2a528644b8e444d52ce"
++++
```

- `OwningRef::map_with_owner` is unsound and may result in a use-after-free.
- `OwningRef::map` is unsound and may result in a use-after-free.
- `OwningRefMut::as_owner` and `OwningRefMut::as_owner_mut` are unsound and may result in a use-after-free.
- The crate violates Rust's aliasing rules, which may cause miscompilations on recent compilers that emit the LLVM `noalias` attribute.

No patched versions are available at this time. While a pull request with some fixes is outstanding, the maintainer appears to be unresponsive.

66. M-31 rocksdb vulnerable to out-of-bounds read

Tags: runtime, GHSA ID: GHSA-xpp3-xrff-w6rh

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L5330

```
++++ ①
[[package]]
name = "rocksdb"
version = "0.16.0"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"c749134fda8bfc90d0de643d59bfc841dcb3ac8a1062e12b6754bd60235c48b3"
++++
```

Affected versions of this crate called the RocksDB C API `rocksdb_open_column_families_with_ttl()` with a pointer to a single integer TTL value, but one TTL value for each column family is expected.

This is only relevant when using `rocksdb::DBWithThreadMode::open_cf_descriptors_with_ttl()` with multiple column families.

This bug has been fixed in v0.19.0.

67. M-32 bumpalo has use-after-free due to a lifetime error in `Vec::into_iter()`

Tags: `runtime`, GHSA ID: GHSA-f85w-wvc7-crwc

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L634

```
++++ ①
[[package]]
name = "bumpalo"
version = "3.7.0"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"9c59e7af012c713f529e7a3ee57ce9b31ddd858d4b512923602f74608b009631"
++++
```

In affected versions of this crate, the lifetime of the iterator produced by `Vec::into_iter()` is not constrained to the lifetime of the `Bump` that allocated the vector's memory. Using the iterator after the `Bump` is dropped causes use-after-free accesses.

The following example demonstrates memory corruption arising from a misuse of this unsoundness.

```
use bumpalo::{collections::Vec, Bump};

fn main() {
    let bump = Bump::new();
    let mut vec = Vec::new_in(&bump);
    vec.extend([0x01u8; 32]);
    let into_iter = vec.into_iter();
    drop(bump);

    for _ in 0..100 {
        let reuse_bump = Bump::new();
        let _reuse_alloc = reuse_bump.alloc([0x41u8; 10]);
    }

    for x in into_iter {
        print!("0x{:02x} ", x);
    }
    println!();
}
```

The issue was corrected in version 3.11.1 by adding a lifetime to the `IntoIter` type, and updating the signature of `Vec::into_iter()` to constrain this lifetime.

68. M-33 Wasmtime out of bounds read/write with zero-memory-pages configuration

Tags: `runtime`, Weakness: CWE-119, CWE-125, CWE-787, CVE ID: CVE-2022-39392, GHSA ID: GHSA-44mr-8vmm-wjhg

File https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L8590

```
++++ ①
[[package]]
name = "wasmtime"
version = "0.22.0"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"7426055cb92bd9a1e9469b48154d8d6119cd8c498c8b70284e420342c05dc45d"
++++
```

68.1. CVSS Score: 5.9/10

Table 58. CVSS:3.1/AV:N/AC:H/PR:H/UI:N/S:U/C:H/I:H/A:N

CVSS base metrics	
Attack vector	Network
Attack complexity	High
Privileges required	High
User interaction	None
Scope	Unchange
Confidentiality	High
Integrity	High
Availability	None

68.2. Impact

There is a bug in Wasmtime's implementation of its pooling instance allocator when the allocator is configured to give WebAssembly instances a maximum of zero pages of memory. In this configuration the virtual memory mapping for WebAssembly memories did not meet the compiler-required configuration requirements for safely executing WebAssembly modules. Wasmtime's default settings require virtual memory page faults to indicate that wasm reads/writes are out-of-bounds, but the pooling allocator's configuration would not create an appropriate virtual memory mapping for this meaning out of bounds reads/writes can successfully read/write memory unrelated to the wasm sandbox within range of the base address of the memory mapping created by the pooling allocator.

This bug can only be triggered by setting `InstanceLimits::memory_pages` to zero. This is expected to be a very rare configuration since this means that wasm modules cannot allocate any pages of linear memory. All wasm modules produced by all current toolchains are highly likely to use linear memory, so it's expected to be unlikely that this configuration is set to zero by any production embedding of Wasmtime, hence the low severity of this bug despite the critical consequences.

68.3. Patches

This bug has been patched and users should upgrade to Wasmtime 2.0.2.

68.4. Workarounds

One way to mitigate this issue is to disable usage of the pooling allocator. Note that the pooling allocator is not enabled by default.

This bug can also only be worked around by increasing the `memory_pages` allotment when configuring the pooling allocator to a value greater than zero. If an embedding wishes to still prevent memory from actually being used then the `Store::limiter` method can be used to dynamically disallow growth of memory beyond 0 bytes large. Note that the default `memory_pages` value is greater than zero.

This bug is not applicable with the default settings of the `wasmtime` crate.

68.5. References

- `Config::allocation_strategy` - configuration required to enable the pooling allocator.
- `InstanceLimits::memory_pages` - configuration field that, when zero, exhibits this bug.
- `Store::limiter` - means of limiting memory without using `memory_pages`
- Mailing list announcement
- Patch for the `release-2.0.0` branch

68.6. For more information

If you have any questions or comments about this advisory:

- Reach out to us on the Bytecode Alliance Zulip chat
- Open an issue in the bytecodealliance/wasmtime repository

69. M-34 Segmentation fault in time

Tags: `runtime`, Weakness: CWE-476, CVE ID: CVE-2020-26235, GHSA ID: GHSA-wcg3-cvx6-7396

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L7784

```
++++ ①
[[package]]
name = "time"
version = "0.1.44"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"6db9e6914ab8b1ae1c260a4ae7a49b6c5611b40328a735b21862567685e73255"
++++
```

69.1. CVSS Score: 6.2/10

Table 59. CVSS:3.1/AV:L/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

CVSS base metrics	
Attack vector	Local
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	None
Integrity	None
Availability	High

69.2. Impact

Unix-like operating systems may segfault due to dereferencing a dangling pointer in specific circumstances. This requires an environment variable to be set in a different thread than the affected functions. This may occur without the user's knowledge, notably in a third-party library.

The affected functions from time 0.2.7 through 0.2.22 are:

- `time::UtcOffset::local_offset_at`
- `time::UtcOffset::try_local_offset_at`
- `time::UtcOffset::current_local_offset`
- `time::UtcOffset::try_current_local_offset`
- `time::OffsetDateTime::now_local`
- `time::OffsetDateTime::try_now_local`

The affected functions in time 0.1 (all versions) are:

- `at`
- `at_utc`
- `now`

Non-Unix targets (including Windows and wasm) are unaffected.

69.3. Patches

In some versions of `time`, the internal method that determines the local offset has been modified to always return `None` on the affected operating systems. This has the effect of returning an `Err` on the `try_*` methods and `UTC` on the non-`try_*` methods. In later versions, `time` will attempt to determine the number of threads running in the process. If the process is single-threaded, the call will proceed as its safety invariant is upheld.

Users and library authors with time in their dependency tree must perform `cargo update`, which will pull in the updated, unaffected code.

Users of time 0.1 do not have a patch and must upgrade to an unaffected version: time 0.2.23 or greater or the 0.3 series.

69.4. Workarounds

Library authors must ensure that the program only has one running thread at the time of calling any affected method. Binary authors may do the same and/or ensure that no other thread is actively mutating the environment.

69.5. References

time-rs/time#293.

70. M-35 h2 vulnerable to denial of service

Tags: `runtime`, Weakness: CWE-770, CVE ID: CVE-2023-26964, GHSA ID: GHSA-f8vr-r385-rh5r

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L2051

```
++++ ①
[[package]]
name = "h2"
version = "0.1.26"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"a5b34c246847f938a410a03c5458c7fee2274436675e76d8b903c08efc29c462"
++++
```

Hyper is an HTTP library for Rust and h2 is an HTTP 2.0 client & server implementation for Rust. An issue was discovered in h2 v0.2.4 when processing header frames. It incorrectly processes the HTTP2 `RST_STREAM` frames by not always releasing the memory immediately upon receiving the reset frame, leading to stream stacking. As a result, the memory and CPU usage are high which can lead to a Denial of Service (DoS).

This issue affects users only when dealing with http2 connections.

71. M-36 Optional `Deserialize` implementations lacking validation

Tags: `runtime`, GHSA ID: GHSA-jf5h-cf95-w759

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L5154

```
++++ ①
[[package]]
name = "raw-cpuid"
version = "8.1.2"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"1fdf7d9dbd43f3d81d94a49c1c3df73cc2b3827995147e6cf7f89d4ec5483e73"
++++
```

When activating the non-default feature `serialize`, most structs implement `serde::Deserialize` without sufficient validation. This allows breaking invariants in safe code, leading to:

- Undefined behavior in `as_string()` methods (which use `std::str::from_utf8_unchecked()` internally).
- Panics due to failed assertions.

See https://github.com/gz/rust-cpuid/issues/43.

72. M-37 Use after free passing `externref`s to Wasm in Wasmtime

Tags: `runtime`, Weakness: CWE-416, CVE ID: CVE-2021-39216, GHSA ID: GHSA-v4cp-h94r-m7xf

File https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L8590

```
++++ ①
[[package]]
name = "wasmtime"
version = "0.22.0"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"7426055cb92bd9a1e9469b48154d8d6119cd8c498c8b70284e420342c05dc45d"
++++
```

72.1. CVSS Score: 6.3/10

Table 60. CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:N/I:H/A:H

CVSS base metrics	
Attack vector	Local
Attack complexity	High
Privileges required	Low
User interaction	None
Scope	Unchange
Confidentiality	None
Integrity	High
Availability	High

72.2. Impact

There was a use-after-free bug when passing `externref`s from the host to guest Wasm content.

To trigger the bug, you have to explicitly pass multiple `externref`s from the host to a Wasm instance at the same time, either by

- passing multiple `externref`s as arguments from host code to a Wasm function,
- or returning multiple `externref`s to Wasm from a multi-value return function defined in the host.

If you do not have host code that matches one of these shapes, then you are not impacted.

If Wasmtime's `VMExternRefActivationsTable` became filled to capacity after passing the first `externref` in, then passing in the second `externref` could trigger a garbage collection. However the first `externref` is not rooted until we pass control to Wasm, and therefore could be reclaimed by the collector if nothing else was holding a reference to it or otherwise keeping it alive. Then, when control was passed to Wasm after the garbage collection, Wasm could use the first `externref`, which at this point has already been freed.

We have reason to believe that the effective impact of this bug is relatively small because usage of `externref` is currently quite rare.

72.3. Patches

The bug has been fixed, and users should upgrade to Wasmtime 0.30.0.

Additionally, we have updated our primary `externref` fuzz target such that it better exercises these code paths and we can have greater confidence in their correctness going forward.

72.4. Workarounds

If you cannot upgrade Wasmtime yet, you can avoid the bug by disabling reference types support in Wasmtime by passing `false` to `wasmtime::Config::wasm_reference_types`.

72.5. References

- The reference types Wasm proposal, which introduces `externref`

72.6. For more information

If you have any questions or comments about this advisory:

- Reach out to us on the Bytecode Alliance Zulip chat
- Open an issue in the `bytecodealliance/wasmtime` repository

73. M-38 memoffset allows reading uninitialized memory

Tags: `runtime`, GHSA ID: GHSA-wfg4-322g-9vqv

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L3468

```
++++ ①
[[package]]
name = "memoffset"
version = "0.5.6"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"043175f069eda7b85febe4a74abbaeff828d9f8b448515d3151a14a3542811aa"
++++
```

memoffset allows attempt of reading data from address `0` with arbitrary type. This behavior is an undefined behavior because address `0` to `std::mem::size_of<T>` may not have valid bit-pattern with `T`. Old implementation dereferences uninitialized memory obtained from `std::mem::align_of`. Older implementation prior to it allows using uninitialized data obtained from `std::mem::uninitialized` with arbitrary type then compute offset by taking the address of field-projection. This may also result in an undefined behavior for "father" that includes (directly or transitively) type that does not allow to be uninitialized.

This flaw was corrected by using `std::ptr::addr_of` in https://github.com/Gilnaa/memoffset/pull/50.

74. M-39 ed25519-dalek Double Public Key Signing Function Oracle Attack

Tags: runtime, GHSA ID: GHSA-w5vr-6qhr-36cc

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L1312

```
++++ ①
[[package]]
name = "ed25519-dalek"
version = "1.0.1"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"c762bae6dcaf24c4c84667b8579785430908723d5c889f469d76a41d59cc7a9d"
++++
```

Versions of ed25519-dalek prior to v2.0 model private and public keys as separate types which can be assembled into a Keypair, and also provide APIs for serializing and deserializing 64-byte private/public keypairs.

Such APIs and serializations are inherently unsafe as the public key is one of the inputs used in the deterministic computation of the S part of the signature, but not in the R value. An adversary could somehow use the signing function as an oracle that allows arbitrary public keys as input can obtain two signatures for the same message sharing the same R and only differ on the S part.

Unfortunately, when this happens, one can easily extract the private key.

Revised public APIs in v2.0 of ed25519-dalek do NOT allow a decoupled private/public keypair as signing input, except as part of specially labeled "hazmat" APIs which are clearly labeled as being dangerous if misused.

75. M-40 Resource exhaustion vulnerability in h2 may lead to Denial of Service (DoS)

Tags: `runtime`, GHSA ID: GHSA-8r5v-vm4m-4g25

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L2051

```
++++ ①
[[package]]
name = "h2"
version = "0.1.26"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"a5b34c246847f938a410a03c5458c7fee2274436675e76d8b903c08efc29c462"
++++
```

An attacker with an HTTP/2 connection to an affected endpoint can send a steady stream of invalid frames to force the generation of reset frames on the victim endpoint. By closing their recv window, the attacker could then force these resets to be queued in an unbounded fashion, resulting in Out Of Memory (OOM) and high CPU usage.

This fix is corrected in hyperium/h2#737, which limits the total number of internal error resets emitted by default before the connection is closed.

76. M-41 Unauthenticated Nonce Increment in snow

Tags: `runtime`, Weakness: CWE-440, GHSA ID: GHSA-7g9j-g5jg-3vv3

File https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L6744

```
++++ ①
[[package]]
name = "snow"
version = "0.7.2"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"795dd7aeeee24468e5a32661f6d27f7b5cbed802031b2d7640c7b10f8fb2dd50"
++++
```

76.1. Impact

There was a logic bug where unauthenticated payloads could still cause a nonce increment in snow's internal state. For an attacker with the ability to inject packets into the channel Noise is talking over, this allows a denial-of-service type attack which could prevent communication as it causes the sending and receiving side to be expecting different nonce values than would arrive.

Note that this only affects those who are using the stateful `TransportState`, not those using `StatelessTransportState`.

76.2. Patches

This has been patched in version 0.9.5, and all users are recommended to update.

76.3. References

There will be a more formal report of this in the near future.

77. M-42 Miscompilation of `i8x16.swizzle` and `select` with v128 inputs

Tags: `runtime`, Weakness: CWE-682, CVE ID: CVE-2022-31104, GHSA ID: GHSA-jqwc-c49r-4w2x

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L901

```
++++ ①
[[package]]
name = "cranelift-codegen"
version = "0.69.0"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"1a54e4beb833a3c873a18a8fe735d73d732044004c7539a072c8faa35ccb0c60"
++++
```

77.1. CVSS Score: 4.8/10

Table 61. CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:L/A:L

CVSS base metrics	
Attack vector	Network
Attack complexity	High
Privileges required	None
User interaction	None
Scope	Unchange
Confidentiality	None
Integrity	Low
Availability	Low

77.2. Impact

Wasmtime's implementation of the SIMD proposal for WebAssembly on x86_64 contained two distinct bugs in the instruction lowerings implemented in Cranelift. The aarch64 implementation of the simd proposal is not affected. The bugs were presented in the `i8x16.swizzle` and `select` WebAssembly instructions. The `select` instruction is only affected when the inputs are of `v128` type. The correspondingly affected Cranelift instructions were `swizzle` and `select`.

The `swizzle` instruction lowering in Cranelift erroneously overwrote the mask input register which could corrupt a constant value, for example. This means that future uses of the same constant may see a different value than the constant itself.

The `select` instruction lowering in Cranelift wasn't correctly implemented for vector types that are 128-bits wide. When the condition was 0 the wrong instruction was used to move the correct input to the output of the instruction meaning that only the low 32 bits were moved and the upper 96 bits of the result were left as whatever the register previously contained (instead of the input being moved from). The `select` instruction worked correctly if the condition was nonzero, however.

This bug in Wasmtime's implementation of these instructions on x86_64 represents an incorrect implementation of the specified semantics of these instructions according to the WebAssembly specification. The impact of this is benign for hosts running WebAssembly but represents possible vulnerabilities within the execution of a guest program. For example a WebAssembly program could take unintended branches or materialize incorrect values internally which runs the risk of exposing the program itself to other related vulnerabilities which can occur from miscompilations.

77.3. Patches

We have released Wasmtime 0.38.1 and cranelift-codegen (and other associated cranelift crates) 0.85.1 which contain the corrected implementations of these two instructions in Cranelift.

77.4. Workarounds

If upgrading is not an option for you at this time, you can avoid the vulnerability by disabling the Wasm simd proposal

```
config.wasm_simd(false);
```

Additionally the bug is only present on x86_64 hosts. Other aarch64 hosts are not affected. Note that s390x hosts don't yet implement the simd proposal and are not affected.

77.5. References

- The WebAssembly simd proposal
- Original test case showing the erroneous behavior
- Fix for the `swizzle` instruction
- Fix for the `select` instruction

77.6. For more information

If you have any questions or comments about this advisory:

- Reach out to us on the Bytecode Alliance Zulip chat
- Open an issue in the bytecodealliance/wasmtime repository

Part IV: Low

78. L-1 rootless: `/sys/fs/cgroup` is writable when cgroupns isn't unshared in runc

Tags: `runtime`, Weakness: CWE-281, CVE ID: CVE-2023-25809, GHSA ID: GHSA-m8cg-xc2p-r3fc

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Golang_Cosmos/go.mod#L189

```
++++ ①
    github.com/opencontainers/runc v1.0.3 // indirect
++++
```

78.1. CVSS Score: 2.5/10

Table 62. CVSS:3.1/AV:L/AC:H/PR:H/UI:N/S:C/C:N/I:N/A:L

CVSS base metrics	
Attack vector	Local
Attack complexity	High
Privileges required	High
User interaction	None
Scope	Changed
Confidentiality	None
Integrity	None
Availability	Low

78.2. Impact

It was found that rootless runc makes `/sys/fs/cgroup` writable in following conditons: 1. when runc is executed inside the user namespace, and the `config.json` does not specify the cgroup namespace to be unshared (e.g.., `(docker|podman|nerdctl) run --cgroupns=host`, with Rootless Docker/Podman/nerdctl) 2. or, when runc is executed outside the user namespace, and `/sys` is mounted with `rbind, ro` (e.g., `runc spec --rootless`; this condition is very rare)

A container may gain the write access to user-owned cgroup hierarchy `/sys/fs/cgroup/user.slice/…` on the host . Other users's cgroup hierarchies are not affected.

78.3. Patches

v1.1.5 (planned)

78.4. Workarounds

- Condition 1: Unshare the cgroup namespace (`(docker|podman|nerdctl) run`

`--cgroupns=private`). This is the default behavior of Docker/Podman/nerdctl on cgroup v2 hosts.

- Condition 2 (very rare): add `/sys/fs/cgroup` to `maskedPaths`

79. L-2 Go package github.com/cosmos/cosmos-sdk module x/crisis does NOT cause chain halt

Tags: `runtime`, GHSA ID: GHSA-qfc5-6r3j-jj22

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Golang_Cosmos/go.mod#L6

```
++++ ①
    github.com/cosmos/cosmos-sdk v0.45.4
++++
```

79.1. x/crisis does NOT cause chain halt

79.2. Impact

If an invariant check fails on a Cosmos SDK network and a transaction is sent to the `x/crisis` module to halt the chain, the chain does not halt. All versions of the `x/crisis` module is affected on all versions of the Cosmos SDK.

79.3. Details

The `x/crisis` module is supposed to allow anyone to halt a chain in the event of a violated invariant by sending a `MsgVerifyInvariant` with the name of the invariant. Processing this message is supposed to cause the nodes to panic. However, because the panic is within a transaction, it is caught by the SDK's built-in panic-recovery machinery and just treated as a normal "invalid" transaction (ie. it returns a non-zero abci Code). Thus the `x/crisis` transactions don't actually cause chains to halt. If there is an invariant violation, it can be confirmed with an `x/crisis` transaction, but it won't cause any nodes to halt, they will just continue processing blocks.

That said, any node running with `start --inv-check-period X` will actually panic when it runs the periodic check (though it will still not panic just by processing an `x/crisis` transaction). Since this panic is located in EndBlock, it is not caught by the panic-recovery machinery and does actually crash the node. Presumably few if any nodes actually run with this in production because of how long the invariant checks take, and this runs all of them every `X` blocks.

79.4. Patches

No patches will be released.

The `x/crisis` module was originally intended to allow chains to halt rather than continue with some unknown behaviour in the case of an invariant violation (safety over liveness). However, as chains mature, and especially as the potential cost of halting increases, chains should consider carefully what invariants they really want to halt for, and what invariants are just sort of helpful sanity checks, but may not be worth halting for.

In some cases, chains have already broken the invariant calculations but have dealt with the consequences off-chain or during development. Halting these chains would be counter-productive.

The SDK team is working on new modules that allow chain developers to fine-tune the chain invariants and the necessary actions.

Hence, the decision was made that the `x/crisis` module will not be patched for chain halts. The module will be deprecated when new modules take over its responsibilities.

79.5. Workarounds

In case of a valid invariant check failure that requires a chain halt, the network validators are encouraged to coordinate off-chain for network halts. This has been an already established process for security patches.

79.6. References

SDK developer epic about invariant checking: https://github.com/cosmos/cosmos-sdk/issues/15706 Public report: https://github.com/cosmos/cosmos-sdk/issues/15325

80. L-3 github.com/cosmos/cosmos-sdk's x/crisis does not charge ConstantFee

Tags: `runtime`, GHSA ID: GHSA-w5w5-2882-47pc

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Golang_Cosmos/go.mod#L6

```
++++ ①
    github.com/cosmos/cosmos-sdk v0.45.4
++++
```

80.1. x/crisis does not charge ConstantFee

80.2. Impact

If a transaction is sent to the `x/crisis` module to check an invariant, the ConstantFee parameter of the chain is NOT charged. All versions of the `x/crisis` module are affected on all versions of the Cosmos SDK.

80.3. Details

The `x/crisis` module is supposed to allow anyone to halt a chain in the event of a violated invariant by sending a `MsgVerifyInvariant` with the name of the invariant. Processing this message takes extra processing power hence a `ConstantFee` was introduced on the chain that is charged as extra from the reporter for the extra computational work. This is supposed to avert spammers on the chain making nodes do extra computations using this transaction. By not charging the `ConstantFee`, the transactions related to invariant checking are relatively cheaper compared to the computational need and other transactions.

That said, the submitter still has to pay the transaction fee to put the transaction on the network, hence using this weakness for spamming is limited by the usual mechanisms.

Synthetic testing showed up to a 20% increase in CPU usage on a validator node that is spammed by hundreds of `MsgVerifyInvariant` messages which still makes this an expensive operation to carry out on a live blockchain network.

80.4. Patches

The `ConstantFee` charge of the `x/crisis` module will either be fixed or disabled in an upcoming regular release of the Cosmos SDK.

The `x/crisis` module was originally intended to allow chains to halt rather than continue with some unknown behavior in the case of an invariant violation (safety over liveness). However, as chains mature, and especially as the potential cost of halting increases, chains should consider carefully what invariants they really want to halt for, and what invariants are just sort of helpful sanity checks.

The SDK team is working on new modules that allow chain developers to fine-tune the chain invariants and the necessary actions.

Hence, the decision was made that the `x/crisis` module will be deprecated when new modules take over its responsibilities.

80.5. Workarounds

There is no workaround posted. Validators are advised to leave some extra computing room on their servers for possible spamming scenarios. (This is a good measure in any case.)

80.6. References

SDK developer epic about invariant checking: https://github.com/cosmos/cosmos-sdk/issues/15706

81. L-4 ASA-2024-003: Missing `BlockedAddressed` Validation in Vesting Module

Tags: `runtime`, Weakness: CWE-20, GHSA ID: GHSA-4j93-fm92-rp4m

File https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Golang_Cosmos/go.mod#L6

```
++++ ①
    github.com/cosmos/cosmos-sdk v0.45.4
++++
```

81.1. CVSS Score: 3.5/10

Table 63. CVSS:3.1/AV:A/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:L

CVSS base metrics	
Attack vector	Adjacent
Attack complexity	Low
Privileges required	Low
User interaction	None
Scope	Unchange
Confidentiality	None
Integrity	None
Availability	Low

81.2. ASA-2024-003: Missing `BlockedAddressed` Validation in Vesting Module

Component: Cosmos SDK **Criticality**: Low **Affected Versions**: Cosmos SDK versions ¬ 0.50.3; ¬ 0.47.8 **Affected Users**: Chain developers, Validator and Node operators **Impact**: Denial of Service

81.3. Description

A vulnerability was identified in the `x/auth/vesting` module, which can allow a user to create a periodic vesting account on a blocked address, for example a non-initialized module account. Additional validation was added to prevent creation of a periodic vesting account in this scenario.

If this case is triggered, there is the potential for a chain halt if the uninitialized account in question is called by `GetModuleAccount` in `Begin`/`EndBlock` of a module. This combination of an uninitialized blocked module account is not common.

81.4. Next Steps for Impacted Parties

If your chain has uninitialized blocked module accounts, it is recommended to proactively initialize them, as they are often initialized during a chain migration or during init genesis.

If you are a chain developer on an affected version of the Cosmos SDK, it is advised to update to the latest available version of the Cosmos SDK for your project. Once a patched version is available, it is recommended that network operators upgrade.

A Github Security Advisory for this issue is available in the Cosmos-SDK repository. For more information about Cosmos SDK, see https://docs.cosmos.network/.

This issue was found by Dongsam who reported it to the Cosmos Bug Bounty Program on HackerOne on January 30, 2024. If you believe you have found a bug in the Interchain Stack or would like to contribute to the program by reporting a bug, please see https://hackerone.com/cosmos.

82. L-5 ASA-2024-005: Potential slashing evasion during re-delegation

Tags: `runtime`, Weakness: CWE-372, GHSA ID: GHSA-86h5-xcpx-cfqc

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Golang_Cosmos/go.mod#L6

```
++++ ①
    github.com/cosmos/cosmos-sdk v0.45.4
++++
```

82.1. ASA-2024-005: Potential slashing evasion during re-delegation

Component: Cosmos SDK **Criticality**: Low **Affected Versions**: Cosmos SDK versions ¬ 0.50.4; ¬ 0.47.9 **Affected Users**: Chain developers, Validator and Node operators **Impact**: Slashing Evasion

82.2. Summary

An issue was identified in the slashing mechanism that may allow for the evasion of slashing penalties during a slashing event. If a delegation contributed to byzantine behavior of a validator, and the validator has not yet been slashed, it may be possible for that delegation to evade a pending slashing penalty through re-delegation behavior. Additional validation logic was added to restrict this behavior.

82.3. Next Steps for Impacted Parties

If you are a chain developer on an affected version of the Cosmos SDK, it is advised to update to the latest available version of the Cosmos SDK for your project. Once a patched version is available, it is recommended that network operators upgrade.

A Github Security Advisory for this issue is available in the Cosmos-SDK repository. For more information about Cosmos SDK, see https://docs.cosmos.network/.

This issue was found by cat shark (Khanh) who reported it to the Cosmos Bug Bounty Program on HackerOne on December 6, 2024. If you believe you have found a bug in the Interchain Stack or would like to contribute to the program by reporting a bug, please see https://hackerone.com/cosmos.

83. L-6 Lenient Parsing of Content-Length Header When Prefixed with Plus Sign

Tags: `runtime`, Weakness: CWE-444, CVE ID: CVE-2021-32715, GHSA ID: GHSA-f3pg-qwvg-p99c

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L2259

```
++++ ①
[[package]]
name = "hyper"
version = "0.12.36"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"5c843caf6296fc1f93444735205af9ed4e109a539005abb2564ae1d6fad34c52"
++++
```

83.1. CVSS Score: 3.1/10

Table 64. CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N

CVSS base metrics	
Attack vector	Network
Attack complexity	High
Privileges required	None
User interaction	Required
Scope	Unchange
Confidentiality	Low
Integrity	None
Availability	None

83.2. Summary

hyper's HTTP/1 server code had a flaw that incorrectly parses and accepts requests with a `Content-Length` header with a prefixed plus sign, when it should have been rejected as illegal. This combined with an upstream HTTP proxy that doesn't parse such `Content-Length` headers, but forwards them, can result in "request smuggling" or "desync attacks".

83.3. Vulnerability

The flaw exists in all prior versions of hyper, if built with `rustc` v1.5.0 or newer.

Example:

```
GET / HTTP/1.1
Host: example.com
Content-Length: +3

abc
```

This request gets accepted and hyper reads the body as abc. The request *should* be rejected, according to RFC 7230, since the ABNF for `Content-Length` only allows for ``DIGIT`s. This is due to using the `FromStr`` implementation for `u64` in the standard library. By differing from the spec, it is possible to send requests like these to endpoints that have different HTTP implementations, with different interpretations of the payload semantics, and cause "desync attacks".

In this particular case, an upstream proxy would need to error when parsing the `Content-Length`, but *not* reject the request (swallowing its own error), and forwarding the request as-is with the `Content-Length` still included. *Then* the upstream proxy and hyper would disagree on the length of the request body. The combination of these factors would be extremely rare.

Read more about desync attacks: https://portswigger.net/research/http-desync-attacks-request-smuggling-reborn

83.4. Impact

To determine if vulnerable, all these things must be true:

- **Using hyper as an HTTP server**. While the lenient decoder also exists in the client, a vulnerability does not exist around *responses*.
- **Using HTTP/1**. The HTTP/2 code uses a stricter parser.
- **Using a vulnerable HTTP proxy upstream to hyper**. If an upstream proxy correctly rejects the illegal `Content-Length` header, **OR** can parse the length with the plus sign, the desync attack cannot succeed.

83.5. Patches

We have released the following patch versions:

- v0.14.10 (to be released when this advisor is published)

83.6. Workarounds

Besides upgrading hyper, you can take the following options:

- Reject requests manually that contain a plus sign prefix in the `Content-Length` header.
- Ensure any upstream proxy handles `Content-Length` headers with a plus sign prefix.

83.7. Credits

This issue was initially reported by Mattias Grenfeldt and Asta Olofsson.

84. L-7 `tokio::io::ReadHalf<T>::unsplit` is Unsound

Tags: `runtime`, GHSA ID: GHSA-4q83-7cq4-p6wg

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L7837

```
++++ ①
[[package]]
name = "tokio"
version = "0.1.22"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"5a09c0b5bb588872ab2f09afa13ee6e9dac11e10a0ec9e8e3ba39a5a5d530af6"
++++
```

`tokio::io::ReadHalf<T>::unsplit` can violate the `Pin` contract

The soundness issue is described in the tokio/issues#5372

Specific set of conditions needed to trigger an issue (a !Unpin type in ReadHalf) is unusual, combined with the difficulty of making any arbitrary use-after-free exploitable in Rust without doing a lot of careful alignment of data types in the surrounding code.

The `tokio` feature `io-util` is also required to be enabled to trigger this soundness issue.

Thanks to zachs18 reporting the issue to Tokio team responsibly and taiki-e and carllerche appropriately responding and fixing the soundness bug.

Tokio before 0.2.0 used `futures` 0.1 that did not have `Pin`, so it is not affected by this issue.

85. L-8 Race Condition Enabling Link Following and Time-of-check Time-of-use (TOCTOU) Race Condition in remove_dir_all

Tags: `runtime`, Weakness: CWE-366, CWE-367, GHSA ID: GHSA-mc8h-8q98-g5hr

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L5300

```
++++ ①
[[package]]
name = "remove_dir_all"
version = "0.5.3"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"3acd125665422973a33ac9d3dd2df85edad0f4ae9b00dafb1a05e43a9f5ef8e7"
++++
```

The `remove_dir_all` crate is a Rust library that offers additional features over the Rust standard library `fs::remove_dir_all` function. It suffers the same class of failure as the code it was layering over: TOCTOU race conditions, with the ability to cause arbitrary paths to be deleted by substituting a symlink for a path after the type of the path was checked.

Thanks to the Rust security team for identifying the problem and alerting us to it.

86. L-9 Undefined Behavior in Rust runtime functions

Tags: `runtime`, Weakness: CWE-758, CVE ID: CVE-2023-30624, GHSA ID: GHSA-ch89-5g45-qwc7

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L8590

```
++++ ①
[[package]]
name = "wasmtime"
version = "0.22.0"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"7426055cb92bd9a1e9469b48154d8d6119cd8c498c8b70284e420342c05dc45d"
++++
```

86.1. CVSS Score: 3.9/10

Table 65. CVSS:3.1/AV:N/AC:H/PR:H/UI:R/S:U/C:L/I:L/A:L

CVSS base metrics	
Attack vector	Network
Attack complexity	High
Privileges required	High
User interaction	Required
Scope	Unchange
Confidentiality	Low
Integrity	Low
Availability	Low

86.2. Impact

Wasmtime's implementation of managing per-instance state, such as tables and memories, contains LLVM-level undefined behavior. This undefined behavior was found to cause runtime-level issues when compiled with LLVM 16 which causes some writes, which are critical for correctness, to be optimized away. Vulnerable versions of Wasmtime compiled with Rust 1.70, which is currently in beta, or later are known to have incorrectly compiled functions. Versions of Wasmtime compiled with the current Rust stable release, 1.69, and prior are not known at this time to have any issues, but can theoretically exhibit potential issues.

The underlying problem is that Wasmtime's runtime state for an instance involves a Rust-defined structure called `Instance` which has a trailing `VMContext` structure after it. This `VMContext` structure has a runtime-defined layout that is unique per-module. This representation cannot be expressed with safe code in Rust so `unsafe` code is required to maintain this state. The code doing this, however, has methods which take `&self` as an argument but modify data in the `VMContext` part of the allocation. This means that pointers derived from `&self` are mutated. This is typically not allowed, except in the presence of `UnsafeCell`, in Rust. When compiled to LLVM these functions have `noalias readonly` parameters which means it's UB to write through the pointers.

Wasmtime's internal representation and management of `VMContext` has been updated to use `&mut self` methods where appropriate. Additionally verification tools for `unsafe` code in Rust, such as `cargo miri`, are planned to be executed on the `main` branch soon to fix any Rust-level issues that may be exploited in future compiler versions.

Precomplied binaries available for Wasmtime from GitHub releases have been compiled with at most LLVM 15 so are not known to be vulnerable. As mentioned above, however, it's still recommended to update.

86.3. Patches

Wasmtime version 6.0.2, 7.0.1, and 8.0.1 have been issued which contain the patch necessary to work correctly on LLVM 16 and have no known UB on LLVM 15 and earlier.

86.4. Workarounds

If Wasmtime is compiled with Rust 1.69 and prior, which use LLVM 15, then there are no known issues. There is a theoretical possibility for UB to exploited, however, so it's recommended that users upgrade to a patched version of Wasmtime. Users using beta Rust (1.70 at this time) or nightly Rust (1.71 at this time) must update to a patched version to work correctly.

86.5. References

- GitHub Advisory
- Mailing list announcement

86.6. For more information

If you have any questions or comments about this advisory:

- Reach out to us on the Bytecode Alliance Zulip chat
- Open an issue in the bytecodealliance/wasmtime repository

87. L-10 atty potential unaligned read

Tags: `runtime`, GHSA ID: GHSA-g98v-hv3f-hcfr

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L390

```
++++ ①
[[package]]
name = "atty"
version = "0.2.14"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"d9b39be18770d11421cdb1b9947a45dd3f37e93092cbf377614828a319d5fee8"
++++
```

On windows, `atty` dereferences a potentially unaligned pointer.

In practice however, the pointer won't be unaligned unless a custom global allocator is used.

In particular, the `System` allocator on windows uses `HeapAlloc`, which guarantees a large enough alignment.

87.1. atty is Unmaintained

A Pull Request with a fix has been provided over a year ago but the maintainer seems to be unreachable.

Last release of `atty` was almost 3 years ago.

87.2. Possible Alternative(s)

The below list has not been vetted in any way and may or may not contain alternatives;

- std::io::IsTerminal - Stable since Rust 1.70.0
- is-terminal - Standalone crate supporting Rust older than 1.70.0"

88. L-11 wasmtime_trap_code C API function has out of bounds write vulnerability

Tags: `runtime`, Weakness: CWE-787, CVE ID: CVE-2022-39394, GHSA ID: GHSA-h84q-m8rr-3v9q

File
https://github.com/HalbornSecurity/CTFs/blob/e0e91e535617f9ed3bfeb5db740e7c9782dca1ee/HalbornCTF_Rust_Substrate/Cargo.lock#L8590

```
++++ ①
[[package]]
name = "wasmtime"
version = "0.22.0"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"7426055cb92bd9a1e9469b48154d8d6119cd8c498c8b70284e420342c05dc45d"
++++
```

88.1. CVSS Score: 3.8/10

Table 66. CVSS:3.1/AV:L/AC:H/PR:H/UI:R/S:U/C:L/I:L/A:L

CVSS base metrics	
Attack vector	Local
Attack complexity	High
Privileges required	High
User interaction	Required
Scope	Unchange
Confidentiality	Low
Integrity	Low
Availability	Low

88.2. Impact

There is a bug in Wasmtime's C API implementation where the definition of the `wasmtime_trap_code` does not match its declared signature in the `wasmtime/trap.h` header file. This discrepancy causes the function implementation to perform a 4-byte write into a 1-byte buffer provided by the caller. This can lead to three zero bytes being written beyond the 1-byte location provided by the caller.

88.3. Patches

This bug has been patched and users should upgrade to Wasmtime 2.0.2.

88.4. Workarounds

This can be worked around by providing a 4-byte buffer casted to a 1-byte buffer when calling `wasmtime_trap_code`. Users of the `wasmtime` crate are not affected by this issue, only users of the C API function `wasmtime_trap_code` are affected.

88.5. References

- Definition of `wasmtime_trap_code`
- Mailing list announcement
- Patch to fix for `main` branch

88.6. For more information

If you have any questions or comments about this advisory:

- Reach out to us on the Bytecode Alliance Zulip chat
- Open an issue in the bytecodealliance/wasmtime repository

Part V: Informational

89. I-1 Malicious `.DS_Store` file

I ran a dsstore parser [2] to extract the metadata from `/.DS_Store` binary file. I found that `/.DS_Store` file is a malicious file and has probably been tampered with. If you want to learn more about parsing the .DS_Store file and its security implications see article [3].

NOTE	`.DS_Store` file can leak information about the structure of a users directory or server. This information can be used by an attacker to identify the directories and could even expose sensitive information like private keys.

Output from parsing `.DS_Store`:

```
Count:  4
HalbornCTF_Rust_Substrate
HalbornCTF_Rust_Substrate
HalbornCTF_Rust_Substrate
HalbornCTF_Solidity_Ethereum
```

89.1. Impact

The content shows 3 directories of the same name `HalbornCTF_Rust_Substrate` followed by `HalbornCTF_Solidity_Ethereum` which contradicts what is currently present in this repository. This indicated with a **HIGH** probability that `.DS_Store` has a bug or a malicious actor could have tampered with this file. A file system is unlikely to have 3 directories with the same name store within the same directory. I ask you to follow my recommendations below to mitigate this vulnerability.

89.2. Recommendation

- Remove the `.DS_Store` file from the repository see stackoverflow example here.
- Create `.gitignore` file at the root of your repository and add `.DS_Store` to prevent it from being added to the repository in the future.
- Additionally use an update hook to prevent certain files from being pushed to your repositories, see stackoverflow example here. This can prevent anyone from force pushing a file even if it has already been included in your `.gitignore`.
- I recommend that anyone working on this project to take pre-cautions by working in an isolated virtual enviroment to prevent exposing your system files to malicious third parties who could also try to comprimise your system when sharing code.

Exhibit A: Tools Used

A.1. Dependabot

Exhibit B: Challenges

B.1. Context

Terminology

References

- [2] https://github.com/gehaxelt/Python-dsstore
- [3] https://0day.work/parsing-the-ds_store-file-format/