Solana Foundation – ELF Parser

Solana Program Security Audit

Prepared by: **Halborn**

Date of Engagement: **July 13th, 2022 - August 4th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	7/28/2022	Guillermo Álvarez
0.2	Document Updates	07/29/2022	Guillermo Álvarez
0.3	Final Draft	08/04/2022	Guillermo Álvarez
0.4	Draft Review	08/04/2022	Gabi Urrutia
1.0	Remediation Plan	08/15/2022	Guillermo Álvarez
1.1	Remediation Plan Review	08/16/2022	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Guillermo Álvarez	Halborn	Guillermo.Alvarez@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Solana Foundation implemented a new dependency-less ELF parser, which is replacing the goblin crate previously used.

Halborn conducted a security audit on the new ELF parser, beginning on July 13th, 2022 and ending on August 4th, 2022 . The security assessment was scoped to the new ELF parser implemented in the rbpf GitHub repository. Commit hashes and further details can be found in the Scope section of this report.

1.2 AUDIT SUMMARY

The team at Halborn was provided 3 weeks for the engagement and assigned 1 full-time security engineer to audit the security of the code in scope. The security engineer is a blockchain and smart contract security expert with advanced penetration testing and smart contract hacking skills, and deep knowledge of multiple blockchain protocols.

The purpose of this audit is to:

- Identify potential security issues within the programs

In summary, Halborn did not identify any vulnerability affecting the newly implemented ELF parser, only two informational recommendations were presented, which have been partially addressed.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual review of the code and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of the program audit. While manual testing is recommended to uncover flaws in logic, process, and implementation;

automated testing techniques help enhance coverage of programs and can quickly identify items that do not follow security best practices.

The following phases and associated tools were used throughout the term of the audit:

- Research into the architecture, purpose, and use of the platform.
- Manual program code review and walkthrough to identify logic issues.
- Mapping out possible attack vectors.
- Thorough assessment of safety and usage of critical Rust variables and functions in scope that could led to arithmetic vulnerabilities.
- Finding unsafe Rust code usage (cargo-geiger).
- Active Fuzz testing (cargo-fuzz, honggfuzz).
- Scanning dependencies for known vulnerabilities (cargo audit).

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.

2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.
3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW
3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

Code repositories:

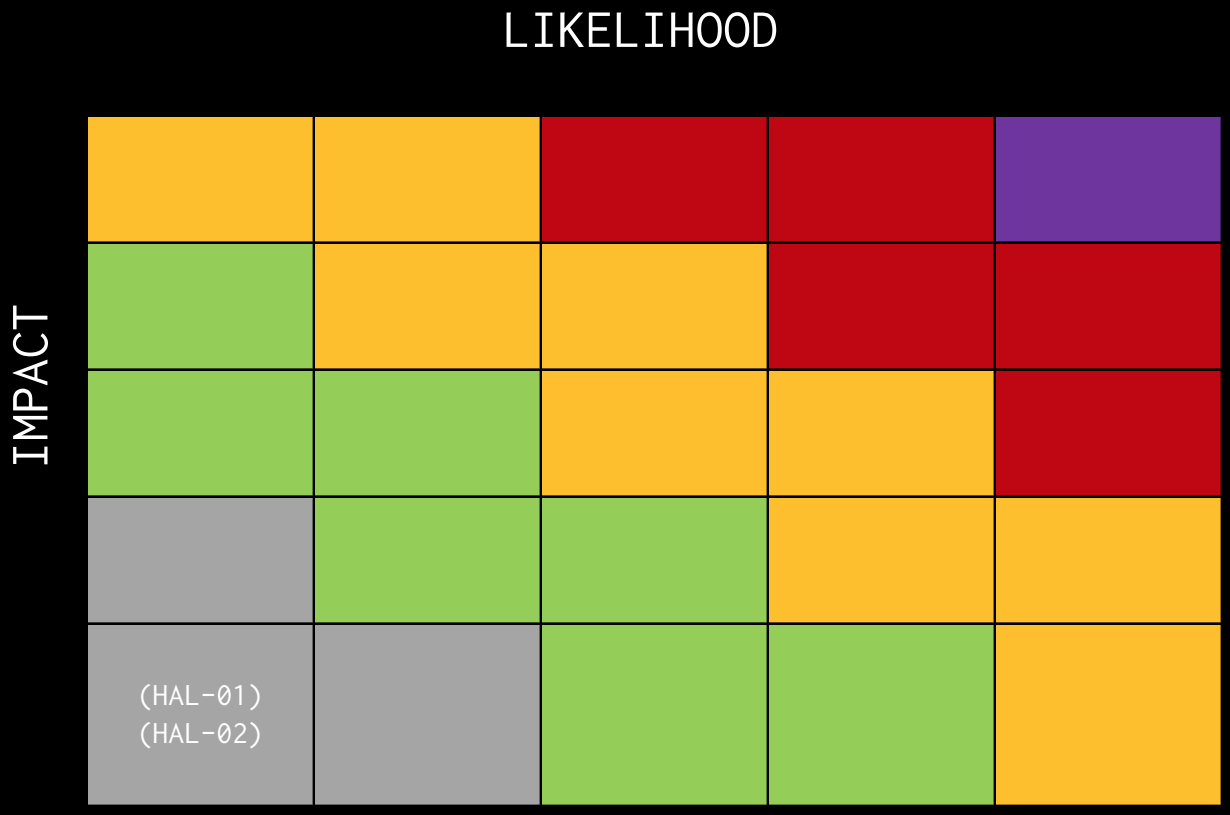
1. Repository: rbpf
2. Pull Request: Introduce new ELF parser
3. Commit IDs:
    - faa80cbc65d2004a4be93b4b9567c3698f8db2f4
    - 93ca9059549aa55677315979e205d7e7a83cddde
    - f39b0358dc295b8265ca90d8d99721057b4447d3
    - dbd7b7a6180f01ebc4328f577acd7e6b3bdd72c8
    - 1174f72bf5a953d1840d7f5e7e41e91fd7776ea2
    - 8cfba85823cb7401621628ea1d74c18426b67e11
    - dbee7d650e8a7988c11d74ad77f405a9f33d34b8
    - 4fde0c40b2e651ed73d2b94d94d317a66e6c0005
    - f26c59ec3e7592466c77bce1dcb18c14bfd2b8c2
    - 287a8f5bbb17884f9970a53c923f1507925c67d4
    - 68d8a91a5dfa789c0ace41091f25bd8a58b4594f
    - e8140d1a84d7f988acd50e7a345b79e6b430e87e
    - 1b640fa8a6cb9cb645f1670185cb71cd8867778a
    - ed4099085b448d8f6be907a6a6d800018efea26b
    - e1495f5cfbf79db0e11db735cf64ef906fdd87af
    - dd189d421ec8fc4de8a5346ddbcabb4d0a99f931
    - 6cf63d990071f1e3254df1283f37ea8d19e792c5
    - ba52d2897eb0cb58d05d3cf9d7f077d3cce31955
    - 8374d8405fcdf49d976981b3b7c3e92e4722c0a7
    - ba30e0c80ecc3fa9d9ac1794b318050525ed6fb2
    - ba30e0c80ecc3fa9d9ac1794b318050525ed6fb2
    - ec84335e76b9087e6f1e65375c2ff51cfdca752a

- eeb9f4613363bacf05db0f0c32a1b79e23bdc355
- 036937d821158af69919805eaec74b96d7dac7e8
- eceac86df2ed985244b13e27f840cac438aec8c5
- 5745ca80b225910d32f81ca163489b136bf113e5
- ebfc58cc372fd159e10399c7e8da4b7202c3bc82
- c096b6629cfdb3d764476fed3a91d1571f3e8bce
- eae90d13d0098fe61ee02189b05968da893682b5
- c2dd6a7a5ecaf2a905f7e7ccb8d574427efe21ef
- 7f801c2199aa03743ae9f27f07fa1ee247422464

Out-of-scope: External libraries and financial related attacks.

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	0	2



SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL-01) WEAK DEFAULT VM CONFIGURATION	Informational	ACKNOWLEDGED
(HAL-02) POSSIBLE RUST PANICS DUE TO UNSAFE UNWRAP USAGE	Informational	FUTURE RELEASE

FINDINGS & TECH DETAILS

3.1 (HAL-01) WEAK DEFAULT VM CONFIGURATION - INFORMATIONAL

Description:

In order to validate and load programs, it is necessary to provide a valid `vm::Config`. The `Config` struct implements the `Default` trait, used in debugging tools such as `rbpf_cli`, or `solana_cli::program`. Default settings are not configured with the most strict options, thus `Config::default()` can be inadvertently used, allowing corrupt ELF files. It is important to note that when `Config::default()` was used in `solana_cli::program`, `reject_broken_elfs` was explicitly set to `true` when it was not necessary for backwards compatibility.

Code Location:

Listing 1: src/vm.rs (Line 245)

```
235 impl Default for Config {
236     fn default() -> Self {
237         Self {
238             max_call_depth: 20,
239             stack_frame_size: 4_096,
240             enable_stack_frame_gaps: true,
241             instruction_meter_checkpoint_distance: 10000,
242             enable_instruction_meter: true,
243             enable_instruction_tracing: false,
244             enable_symbol_and_section_labels: false,
245             reject_broken_elfs: false,
246             noop_instruction_rate: 256,
247             sanitize_user_provided_values: true,
248             encrypt_environment_registers: true,
249             syscall_bpf_function_hash_collision: true,
250             reject_callx_r10: true,
251             dynamic_stack_frames: true,
252             enable_sdiv: true,
253             optimize_rodata: true,
254             static_syscalls: true,
255             enable_elf_vaddr: true,
256             new_elf_parser: true,
```

```
257             }
258         }
259 }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Use default configurations with the most restrictive settings, which can be later overwritten if required.

Remediation Plan:

ACKNOWLEDGED: `Config::Default()` will not be changed since it is only used as a development API and the `runtime` uses hardcoded values and includes a warning against using `Config::Default()`.

3.2 (HAL-02) POSSIBLE RUST PANICS DUE TO UNSAFE UNWRAP USAGE - INFORMATIONAL

Description:

The use of helper methods in Rust, such as unwrap, is allowed in dev and testing environment because those methods are supposed to throw an error (also known as panic!) when called on Option::None or a Result which is not Ok. However, keeping unwrap functions in production environment is considered bad practice because they may lead to program crashes, which are usually accompanied by insufficient or misleading error messages.

Code Location:

Listing 2

```
./jit.rs:947:        let mut diversification_rng = SmallRng::
↳ from_rng(rand::thread_rng()).unwrap();
./static_analysis.rs:191:                self.cfg_nodes.get_mut(&
↳ source).unwrap().destinations = destinations.clone();
./static_analysis.rs:196:                    .unwrap()
./static_analysis.rs:341:                    self.instructions.
↳ last().unwrap().ptr
./static_analysis.rs:383:                                    [self.
↳ cfg_nodes.get(destination).unwrap().instructions.end]
./static_analysis.rs:458:        let mut function_start = *
↳ function_iter.next().unwrap();
./static_analysis.rs:463:                    function_start = *
↳ function_iter.next().unwrap();
./static_analysis.rs:468:                    self.instructions.
↳ last().unwrap().ptr + 1
./static_analysis.rs:561:                self.instructions.last()
↳ .unwrap().ptr + 1
./static_analysis.rs:630:                let dynamic_analysis =
↳ dynamic_analysis.unwrap();
./static_analysis.rs:700:            let cfg_node = self.cfg_nodes
↳ .get(&node.cfg_node).unwrap();
./static_analysis.rs:705:                    .unwrap()
```

```
   ./static_analysis.rs:747:            let cfg_node = self.cfg_nodes
↳ .get_mut(&node.cfg_node).unwrap();
   ./static_analysis.rs:780:                    super_root.
↳ destinations.push(first_node.unwrap());
   ./static_analysis.rs:787:            let cfg_node = self.cfg_nodes
↳ .get_mut(v).unwrap();
   ./static_analysis.rs:825:            .unwrap()
   ./static_analysis.rs:843:                    let mut cfg_node =
↳ self.cfg_nodes.get_mut(b).unwrap();
   ./static_analysis.rs:859:            let dominator_cfg_node = self
↳ .cfg_nodes.get_mut(&p).unwrap();
   ./static_analysis.rs:1084:                       .unwrap(),
   ./static_analysis.rs:1116:                    .unwrap();
   ./call_frames.rs:37:        stack.resize(stack_len, 0).unwrap();
   ./elf_parser_glue.rs:266:               ei_mag: h.e_ident[0..4].
↳ try_into().unwrap(),
   ./elf_parser/mod.rs:495:                   self.
↳ section_names_section_header.unwrap(),
   ./elf_parser/mod.rs:499:                .unwrap();
   ./elf_parser/mod.rs:504:           let symbol_table = self.
↳ get_symbol_table_of_section(section_header).unwrap();
   ./elf_parser/mod.rs:510:                            self.
↳ symbol_names_section_header.unwrap(),
   ./elf_parser/mod.rs:514:                       .unwrap();
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended not to use the unwrap function in the production environment because its use causes panic! and may crash the contract without verbose error messages. Crashing the system will result in a loss of availability and, in some cases, even private information stored in the state. Some alternatives are possible, such as propagating the error with ? instead of unwrapping, or using the error-chain crate for errors.

Remediation Plan:

PENDING: Solana Foundation stated that most unwrap functions were present in debug and off-chain code, one call is in the JIT code and will be fixed in a future release.

AUTOMATED TESTING

4.1 AUTOMATED ANALYSIS

Description:

Halborn used automated security scanners to assist with detection of well-known security issues and vulnerabilities. Among the tools used was cargo-audit, a security scanner for vulnerabilities reported to the RustSec Advisory Database. All vulnerabilities published in https://crates.io are stored in a repository named The RustSec Advisory Database. cargo audit is a human-readable version of the advisory database which performs a scanning on Cargo.lock. Security Detections are only in scope. All vulnerabilities shown here were already disclosed in the above report. However, to better assist the developers maintaining this code, the auditors are including the output with the dependencies tree, and this is included in the cargo audit output to better know the dependencies affected by unmaintained and vulnerable crates.

Results:

No vulnerabilities were identified.

4.2 UNSAFE RUST CODE DETECTION

Description:

Halborn used automated security scanners to assist with the detection of well-known security issues and vulnerabilities. Among the tools used was cargo-geiger, a security tool that lists statistics related to the usage of unsafe Rust code in a core Rust codebase and all its dependencies.

Results:

```
Symbols:
    🔒  = No `unsafe` usage found, declares #![forbid(unsafe_code)]
    ❓  = No `unsafe` usage found, missing #![forbid(unsafe_code)]
    ☢  = `unsafe` usage found

Functions  Expressions  Impls  Traits  Methods  Dependency

0/0        221/221      0/0    0/0     0/0      ☢ solana_rbpf 0.2.31
1/1        193/193      0/0    0/0     0/0      ☢ ├── byteorder 1.4.3
0/0        13/14        0/0    0/0     1/1      ☢ ├── combine 3.8.1
0/0        156/156      0/0    0/0     28/28    ☢ │   ├── ascii 0.9.3
1/1        193/193      0/0    0/0     0/0      ☢ │   ├── byteorder 1.4.3
0/0        0/0          0/0    0/0     0/0      ❓ │   ├── either 1.6.1
2/37       361/2140     0/0    0/0     4/16     ☢ │   ├── memchr 2.4.1
1/20       10/353       0/2    0/0     5/38     ☢ │   │   └── libc 0.2.122
1/1        12/12        0/0    0/0     4/4      ☢ │   └── unreachable 1.0.0
0/0        0/0          0/0    0/0     0/0      ❓ │       └── void 1.0.2
0/0        2/2          6/6    0/0     1/1      ☢ ├── goblin 0.5.1
1/1        16/16        1/1    0/0     0/0      ☢ │   ├── log 0.4.16
0/0        0/0          0/0    0/0     0/0      ❓ │   │   └── cfg-if 1.0.0
2/2        29/31        11/13  1/1     0/0      ☢ │   ├── plain 0.2.3
0/0        12/12        0/0    0/0     0/0      ☢ │   └── scroll 0.11.0
0/0        0/0          0/0    0/0     0/0      ❓ │       └── scroll_derive 0.11.0
0/0        12/12        0/0    0/0     3/3      ☢ │           ├── proc-macro2 1.0.37
0/0        0/0          0/0    0/0     0/0      🔒 │           │   └── unicode-xid 0.2.2
0/0        0/0          0/0    0/0     0/0      ❓ │           ├── quote 1.0.17
0/0        12/12        0/0    0/0     3/3      ☢ │           │   └── proc-macro2 1.0.37
0/0        47/47        3/3    0/0     2/2      ☢ │           └── syn 1.0.91
0/0        12/12        0/0    0/0     3/3      ☢ │               ├── proc-macro2 1.0.37
0/0        0/0          0/0    0/0     0/0      ❓ │               ├── quote 1.0.17
0/0        0/0          0/0    0/0     0/0      🔒 │               └── unicode-xid 0.2.2
0/0        29/29        0/0    0/0     0/0      ☢ ├── hash32 0.2.1
1/1        193/193      0/0    0/0     0/0      ☢ │   └── byteorder 1.4.3
1/20       10/353       0/2    0/0     5/38     ☢ ├── libc 0.2.122
1/1        16/16        1/1    0/0     0/0      ☢ ├── log 0.4.16
0/0        32/32        0/0    0/0     0/0      ☢ ├── rand 0.8.5
1/20       10/353       0/2    0/0     5/38     ☢ │   ├── libc 0.2.122
1/1        16/16        1/1    0/0     0/0      ☢ │   ├── log 0.4.16
0/0        0/0          0/0    0/0     0/0      ❓ │   ├── rand_chacha 0.3.1
0/2        165/712      0/0    0/0     16/25    ☢ │   │   ├── ppv-lite86 0.2.16
0/0        15/15        0/0    0/0     0/0      ☢ │   │   └── rand_core 0.6.3
1/4        53/173       1/1    0/0     3/3      ☢ │   │       └── getrandom 0.2.6
0/0        0/0          0/0    0/0     0/0      ❓ │   │           ├── cfg-if 1.0.0
1/20       10/353       0/2    0/0     5/38     ☢ │   │           └── libc 0.2.122
0/0        15/15        0/0    0/0     0/0      ☢ │   └── rand_core 0.6.3
0/0        0/0          0/0    0/0     0/0      ❓ ├── rustc-demangle 0.1.21
0/0        12/12        0/0    0/0     0/0      ☢ ├── scroll 0.11.0
0/0        0/0          0/0    0/0     0/0      ❓ └── thiserror 1.0.30
0/0        0/0          0/0    0/0     0/0      ❓     └── thiserror-impl 1.0.30
0/0        12/12        0/0    0/0     3/3      ☢         ├── proc-macro2 1.0.37
0/0        0/0          0/0    0/0     0/0      ❓         ├── quote 1.0.17
0/0        47/47        3/3    0/0     2/2      ☢         └── syn 1.0.91

9/68       1378/4170    22/26  1/1     67/121
```

THANK YOU FOR CHOOSING

// HALBORN