Polymath Polymesh 5.0.0 Feature Review

Security Assessment Report

POLYMATH

Prepared for Polymath Inc.
July 26, 2022 (version 1.1)

Project Team:

Technical Testing	Bryan Geraghty and Sean Bradly
Technical Editing	Shawn Moyer
Project Management	Molly Vukusich

Table of Contents

Engagement Overview

Assessment Components and Objectives

Polymath Inc. ("Polymath") recently engaged Atredis Partners ("Atredis") to perform a security assessment of the new features that are going to be released in Polymesh version 5.0.0. Objectives included validation that the new features were developed with security best practices in mind, and to obtain third party validation that any significant vulnerabilities present in Polymath's environment were identified for remediation.

Testing was performed from May 25, 2022 through June 17, 2022, by Bryan Geraghty and Sean Bradly of the Atredis Partners team, with Molly Vukusich providing project management and delivery oversight. For Atredis Partners' assessment methodology, please see Appendix I of this document, and for team biographies, please see Appendix II. Specific testing components and testing tasks are included below.

COMPONENT	ENGAGEMENT TASKS
Polymath Polymesh 5.0.0 Feature Review	

Assessment Tasks	• Source-assisted penetration testing of version 5.0.0 of the Polymesh node client, with emphasis on new features
	  • Review of source code related to changes
	  • Analyze each feature workflow from bootstrap through execution to identify code branches, associated threats, and potential vulnerabilities
	  • Where code architecture allows, development of unit tests to cover abuse cases
	  • Active testing and proof-of-concept development on local and testnet chains

Reporting and Analysis	
Analysis and Deliverables	• Status Reporting and Realtime Communication • Comprehensive Engagement Deliverable • Engagement Outbrief and Remediation Review

The ultimate goal of the assessment was to provide a clear picture of risks, vulnerabilities, and exposures as they relate to accepted security best practices, such as those created by the National Institute of Standards and Technology (NIST), Open Web Application Security Project (OWASP), or the Center for Internet Security (CIS). Augmenting these, Atredis Partners also draws on its extensive experience in secure development and in testing high-criticality applications and advanced exploitation.

Engagement Tasks

Atredis Partners performed the following tasks, at a high level, for in-scope targets during the engagement.

Application Penetration Testing

For relevant web applications, APIs, and web services, Atredis performed automated and manual application penetration testing of these components, applying generally accepted testing best practices as derived from OWASP and the Web Application Security Consortium (WASC).

Testing was performed from the perspective of an anonymous intruder, identifying scenarios from the perspective of an opportunistic, Internet-based threat actor with no knowledge of the environment, as well as, from the perspective a user working to laterally move through the environment to bypass security restrictions and user access levels.

Where relevant, Atredis Partners utilized both automated fuzzing and fault injection frameworks as well as purpose-built, task-specific testing tools tailored to the application and platforms under review.

Binary and Runtime Analysis

For relevant software targets identified during the course of this engagement, Atredis performed binary and runtime analysis, using debugging and decompilation tools to analyze application flow to aid in software security analysis. Where relevant, purpose-built tools such as fuzzers and customized network clients were utilized to aid in vulnerability identification.

Source Code Analysis

Atredis reviewed the in-scope application source code, with an eye for security-relevant software defects. To aid in vulnerability discovery, application components were mapped out and modeled until a thorough understanding of execution flow, code paths, and application design and architecture was obtained. To aid in this process, the assessment team engaged key stakeholders and members of the development team where possible to provide structured walkthroughs and interviews, helping the team rapidly gain an understanding of the application’s design and development lifecycle.

Status Reporting and Realtime Communication

As described in the methodology section below, Atredis Partners scheduled regular status meetings with client representatives during the project as requested and reported findings in realtime as soon as they were confirmed, via secure communication channels.

Executive Summary

Polymesh is a Substrate[1]-based node client that provides asset trading functionality. To support the trading functionality, Polymesh contains complex Identity[2], Customer Due Diligence (CDD)[3], and permissions mechanisms.

In 2021, Atredis performed a targeted-feature assessment of the Polymesh `settlement`, `external-agent`, `relayer`, and `rewards` pallets, as well as the overall bootstrapping, identity, CDD, and permissions system.

For this assessment, Atredis assessed the differences between versions v4.1.2 and 5.0.0-rc1. In order to identify the differences, Polymath provided Atredis with release notes containing links to pull-requests that were targeted for release. After reviewing these, and finding that some were no longer implemented, Atredis performed a `diff` between the two branches and used that output to finalize the test plan. The `staging` branch was used to review the new smart contracts functionality. The changed components are listed below.

The components in bold contained what Atredis considered material changes and are discussed in detail under the Feature Overview section, below. The other components contained minor changes like updated annotations, calling conventions, whitespace, constants, and versions. These were covered mainly through code review.

Changed Pallets

- **asset**
- balances
- base
- bridge
- committee
- common
- **contracts**
- **corporate-actions**
- **external-agents**
- group
- **identity**
- **multisig**
- permissions
- pips
- portfolio
- protocol-fee
- relayer
- rewards
- runtime
- **settlement**
- staking
- **statistics**
- sto
- sudo
- test-utils
- transaction-payment
- **treasury**
- utility
- weights

[1] https://substrate.io
[2] https://developers.polymesh.network/introduction/identity
[3] https://developers.polymesh.network/polymesh-docs/primitives/cdd

Other Changed Directories

- bin/executor
- node-rpc
- primitives
- rpc
- src

In order to explore new functionality and monitor chain activity, Atredis utilized https://polkadot.js.org and https://staging-apps.polymath.workers.dev, but these web clients were not in-scope for the assessment.

Since the node client primarily provides API functionality, Atredis developed a test harness with the Polkadot.js SDK[4] in order to perform active testing of material changes. This consisted of active testing of business logic, authentication, authorization, and validation/encoding of values. The SDK was out of scope for the assessment.

Key Conclusions

Overall, Atredis found the Polymesh node client to be well-designed and developed. In the significant amount of testing performed during this assessment, only two informational issues were identified. One was a case of inverted logic which did not result in a security issue but caused a business logic error. The other was a design issue that may present a concern for how smart contracts are implemented in the future but does not pose a current threat.

Feature Overview

The sections below describe the material changes between versions 4.1.2 and 5.0.0 of Polymesh and how they were tested during this engagement.

Assets

Atredis reviewed the new asset features related to custom asset metadata, metadata locking and expiration. Freeform metadata can be attached to a given asset. New metadata items can be registered for a given asset by providing a name as well as a URL, description and a freeform `typeDef` parameter. This registration will provide an integer identifier that can be later used to refer to the metadata and set values.

[4] https://polkadot.js.org/docs/api/start

Additional data known as details can also be attached to metadata in order to provide locking functionality to prevent further updates to the metadata values. Locking modes include "`Unlocked`", "`Locked`", or "`LockedUntil`" which will eventually expire. A potential issue with the "`LockedUntil`" methodology was identified and documented as an informational issue in Metadata `LockedUntil` Implementation Inverted Logic.

Atredis verified that all metadata API inputs and outputs were properly sanitized or encoded while processing via a mix of automated fuzzing and targeted manual testing used to assist in a static source code review. Similarly, Atredis ensured each API call validated proper access permissions before allowing any sensitive operations to continue.

Contracts

Version 5.0.0 of Polymesh utilizes the smart contract mechanism provided by the Substrate framework. It allows uploading Ink! WASM blobs and has been customized to support Polymesh's Identity, CDD, and permissions mechanisms.

Polymesh contracts can make calls to Polymesh extrinsics[5] that are explicitly exposed through the `contracts` pallet. When contracts are added, they are given their own identity that is used to perform the extrinsic calls. Contract identities are the same as user identities and must be granted permissions and funded before any extrinsics can be called with them.

However, by design, once a contract has been added, any funded user can call a contract. Atredis determined that this may present a concern for future smart contract development, as described in, Any Funded CDD User Can Call Smart Contracts.

At the time of this assessment, extrinsics that were exposed to smart contracts were:

- `asset::register_ticker`
- `asset::accept_ticker_transfer`
- `asset::accept_asset_ownership_transfer`
- `asset::create_asset`
- `asset::register_custom_asset_type`
- `contracts::instantiate_with_hash_perms`

[5] https://docs.substrate.io/v3/concepts/extrinsics

The contract-to-extrinsic lookup was handled through numeric identifiers that were assigned to each function. On the contract side, these were specified through an underscore-separated string of hex-encoded numbers that represent the scheme, pallet, extrinsic, and version, respectively. For example, `0x00_1A_00_00` specifies `scheme` 0, `pallet` 26 (`asset`), `extrinsic` 0 (`register_ticker`), and `version` 0. At the time of this assessment, the `scheme` and `version` fields were not used. The example below shows how some of these calls are defined in the contract source code.

```
55      // V5.0.0-rc1
56      #[ink(extension = 0x00_1A_00_00, returns_result = false)]
57      fn register_ticker(
58        ticker: Ticker
59      );
60
61      // V5.0.0-rc1
62      #[ink(extension = 0x00_1A_04_00, returns_result = false)]
63      fn issue(
64        ticker: Ticker,
65        amount: Amount
66      );
```

Example Contract Definitions for `register_ticker` and `issue` Extrinsics

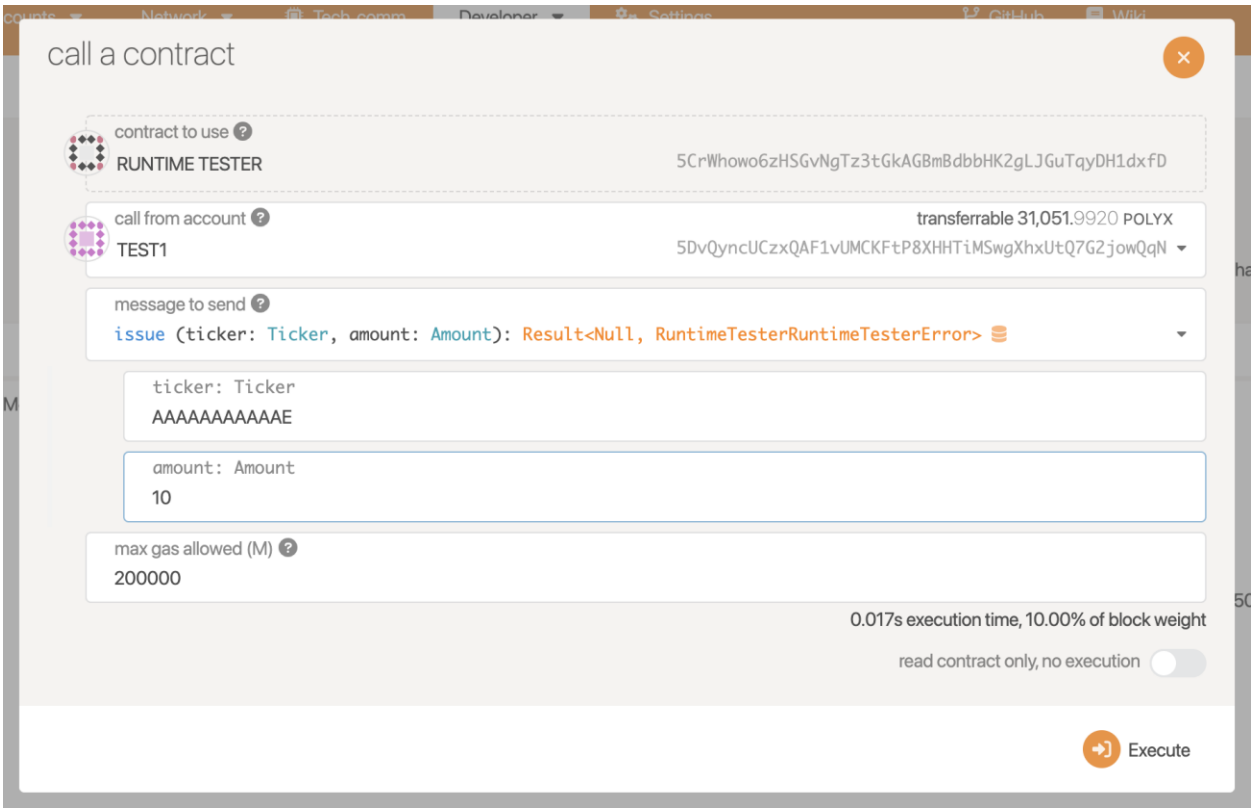
The example below shows how the extrinsic calls are mapped from the specifier string.

```
⚑ 532      /// Pattern match on functions `0x00_pp_ee_00`.
  533      macro_rules! on {
⚑ 534          ($p:pat, $e:pat) => {
  535              FuncId {
  536                  scheme: 0,
  537                  pallet: $p,
  538                  extrinsic: $e,
  539                  version: 0,
  540              }
  541          };
  542      }
  543
  544      let func_id = split_func_id(func_id);
⚑ 545      Ok(match func_id {
⚑ 546          on!(26, 0) => CommonCall::Asset(pallet_asset::Call::register_ticker { ticker:
decode!() }),
  547          on!(26, 1) => {
  548              CommonCall::Asset(pallet_asset::Call::accept_ticker_transfer { auth_id:
decode!() })
  549          }
  550          on!(26, 2) =>
CommonCall::Asset(pallet_asset::Call::accept_asset_ownership_transfer {
  551              auth_id: decode!(),
  552          }),
```

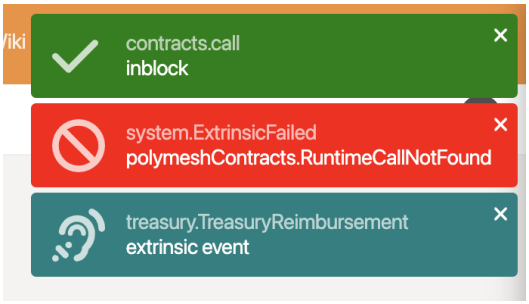
Contract extrinsics call lookup

Because of this strict numeric lookup of the call, combined with the SCALE[6] encoding of parameters, the ability for a contract to escape this process is effectively mitigated.

The example below shows how a contract that tries to call an extrinsic that is not implemented simply fails the lookup.



WASM Representation of Extrinsic Not Implemented in Contracts Pallet



Failed Extrinsic Lookup

[6] https://docs.substrate.io/reference/scale-codec/

Corporate Actions

The main change to the `corporate_actions` pallet is the addition of the `initiate_corporate_action_and_distribute` extrinsic. It is a simple transactional wrapper around calls to `unsafe_initiate_corporate_action` and `unverified_distribute` with a `ensure_agent_asset_perms` permissions check.

Atredis reviewed the code changes and determined that the business logic is upheld and no adverse behavior was introduced. However, Atredis notes that while the operation is wrapped in a transaction, no explicit rollback triggered during in the case of a failure, as is seen in other scenarios like settlement leg failures.

External Agents

Polymesh 5.0.0 added the `create_group_and_add_auth` and `create_and_change_custom_group` extrinsics.

The `create_group_and_add_auth` extrinsic is a simple wrapper around `external_agents::base_create_group` and `identity::add_auth`. The `create_and_change_custom_group` extrinsic is a wrapper around `external_agents::base_create_group` and `external_agents::unsafe_change_group`. In both cases, `base_create_group` performs the authorization checks before the secondary function is called.

Atredis reviewed the source code, reviewed and verified unit tests, and developed test cases with a custom JavaScript test harness that exercises the functionality through external extrinsic calls. Atredis verified that the permissions behave the same as was seen in the previous engagement and that the new extrinsics did not introduce new issues.

Identity

In the Identity pallet, Polymesh 5.0.0 adds transaction weight based on secondary key complexity and limits on when Multisig secondary keys can be removed. Neither of these changes introduced new attack surface, so active testing focused on business logic and authorization enforcement.

When secondary keys were added, complexity of the key's permissions was factored into the weight of the operation in order to combat denial-of-service attacks. Atredis verified this behavior through a JavaScript test harness that intentionally created large permissions objects and observing the weight associated with the transactions.

Multisigs that have been added to an identity as secondary keys can no longer be removed if they have a balance. Atredis traced and reviewed the source code for this feature and verified that it is well-covered by the `identity_test::remove_secondary_keys_test_with_externalities` unit test.

Multisig

In version 5.0.0 of Polymesh the material changes to the `multisig` pallet were primarily simplifying how Multisigs are added as secondary keys to identities. This involved renaming the `make_multisig_signer` extrinsic to `make_multisig_secondary` and a change across the pallet that replaced `KeyToMultiSig` with storage and lookup of keys in the `identity` pallet.

The `proposal_to_id` parameter was also added to the `create_proposal` extrinsic which is a boolean parameter which specifies whether a mapping between the Multisig and proposal should be created in storage.

Atredis reviewed the code changes and determined that the business logic is upheld and no adverse behavior was introduced.

Settlement

The main change made to the `settlement` pallet for Polymesh version 5.0.0 is the addition of a rollback of the entire transaction if processing of a leg fails in `execute_instruction`.

This is a simple change that did not introduce any new attack surface, so testing was limited to code review. Atredis determined that the change did not introduce any security weaknesses.

Statistics

The statistics pallet was entirely rewritten for Polymesh version 5.0.0. It allows asset managers to set various limits on their assets during corporate action and settlement operations.

Ultimately, the `check_transfer_condition` function enforces the limits. It is eventually called as part of the call chain within these functions:

- `settlement::execute_instruction`
- `corporate_actions::transfer_benefit`

If transfer rules are not paused, it validates the following conditions:

- Maximum number of investors
- Maximum percentage of investor ownership
- Minimum and maximum number of investor claims
- Minimum and maximum percentage of claim ownership

Atredis reviewed the Polymesh source code and unit tests in order to understand the changes and identify any potential security issues related to authorization and data handling. Atredis also developed a custom test harness to interact with this behavior for active testing. No issues were identified in the statistics implementation.

Treasury

The main change to the treasury pallet is the way disbursements are handled. Previously, beneficiary accounts were looked up and the disbursement paid to them. In version 5.0.0, the beneficiary's DID is used to look up their primary key and the disbursement is paid to the primary key.

This change did not introduce any new attack surface, so testing focused on potential logic flaws through source review. Ultimately, the changed code was minimal and Atredis did not identify any issues with it.

Findings Summary

In performing testing for this assessment, Atredis Partners identified **two (2) informational** findings. No low, medium, high or critical severity findings were noted.

Atredis defines vulnerability severity ranking as follows:

- **Critical:** These vulnerabilities expose systems and applications to immediate threat of compromise by a dedicated or opportunistic attacker.
- **High:** These vulnerabilities entail greater effort for attackers to exploit and may result in successful network compromise within a relatively short time.
- **Medium:** These vulnerabilities may not lead to network compromise but could be leveraged by attackers to attack other systems or applications components or be chained together with multiple medium findings to constitute a successful compromise.
- **Low:** These vulnerabilities are largely concerned with improper disclosure of information and should be resolved. They may provide attackers with important information that could lead to additional attack vectors or lower the level of effort necessary to exploit a system.

Findings and Recommendations

The following section outlines findings identified via manual and automated testing over the course of this engagement. Where necessary, specific artifacts to validate or replicate issues are included, as well as Atredis Partners' views on finding severity and recommended remediation.

Findings Summary

The below tables summarize the number and severity of the unique issues identified throughout the engagement.

CRITICAL	HIGH	MEDIUM	LOW	INFO
0	0	0	0	2

Findings Detail

FINDING NAME	SEVERITY
Any Funded CDD User Can Call Smart Contracts	Info
Metadata `LockedUntil` Implementation Inverted Logic	Info

Any Funded CDD User Can Call Smart Contracts

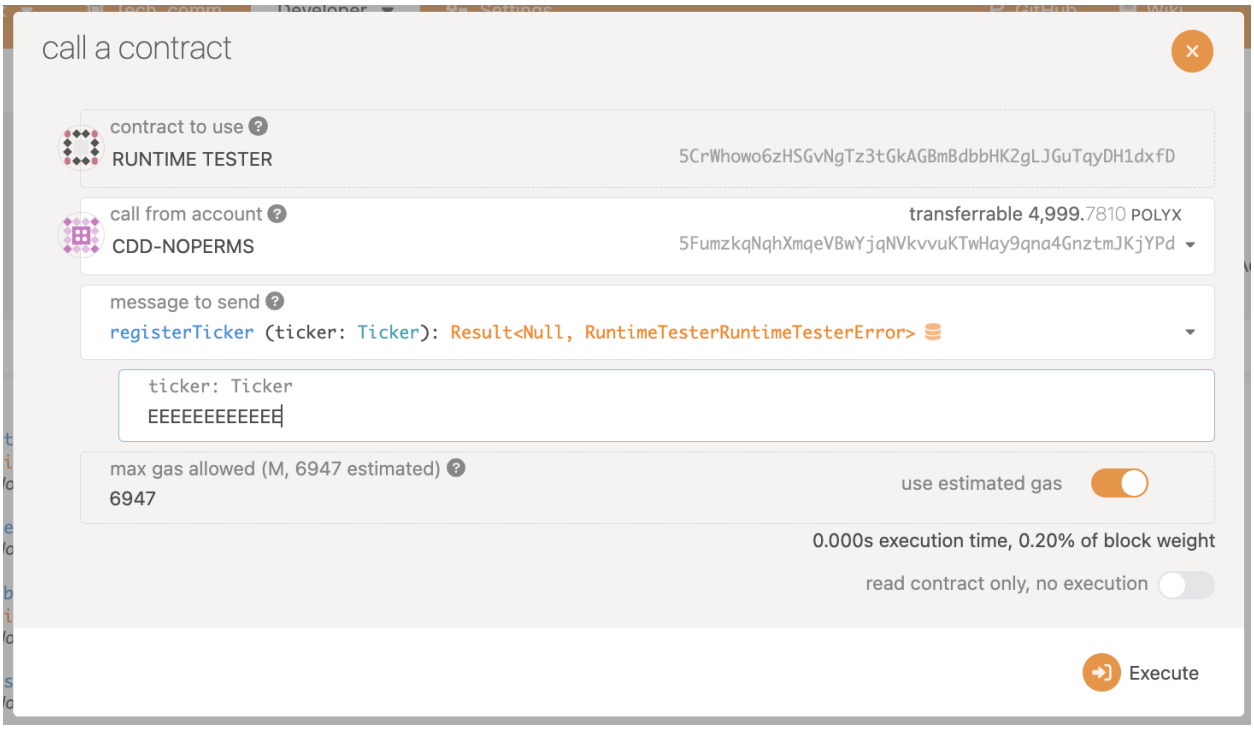
Severity: Info

Finding Overview

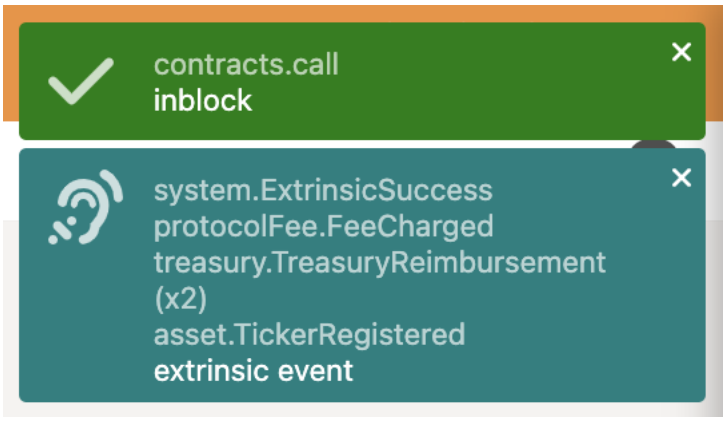
Polymesh smart contracts must be granted permissions to call extrinsics, but no permissions are required for a user to call a contract. This design decision presents a potential compromise to the integrity of the Polymesh CDD system.

Finding Detail

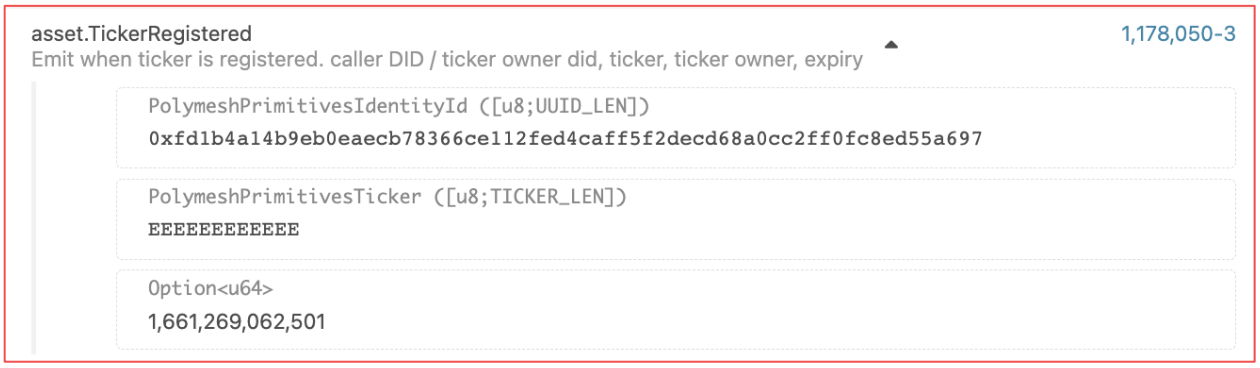
Polymesh is built with a complex Customer Due Diligence and permission system that requires users to be CDD members and have explicit permissions to call extrinsics for any given asset. Polymesh contracts have their own identities that must be granted permissions to call extrinsics. However, once a contract has been uploaded, any CDD user with funding can call the contract, as shown below.



Contract Called By Unpermissioned Caller



Extrinsic Success



Ticker Registered and Owned by Contract

In the above example, an asset is created and owned by the contract. If any other functionality to manage that asset were added, any Polymesh CDD user could perform the calls to manage that asset.

The current Polymesh contract implementation leaves this facet of contract implementation to contract creators. Each contract that creates assets must also track its callers and manage permissions to perform various functions on those assets.

Recommendation(s)

Polymesh smart contracts provide a means to package custom functionality. In some cases, it might make sense to allow any user to call the contract. However, because of the restricted nature of Polymesh operations, there are many cases where contract execution should also be restricted.

In order to maintain the integrity of Polymesh's CDD and permissions system, contracts should have assignable permissions similar to Polymesh Assets. For example, Polymesh could provide a contract permission set that would be attached to a contract and deny access by default, but could be set to allow anyone to call it if needed or modified to allow specific accounts. Contract permissions would provide a framework that contract developers could use as-needed rather than having to invent something on their own.

References

CWE-862: Missing Authorization
https://cwe.mitre.org/data/definitions/862.html

Metadata `LockedUntil` Implementation Inverted Logic

Severity: Info

Finding Overview

One of the mechanisms by which an owner or an agent can lock the metadata for an asset is to set a `LockedUntil` detail. This feature, when provided with a future timestamp, should lock metadata out of any changes until sometime after the timestamp appears in the blockchain. Atredis observed that, instead of the intended behavior, it does the opposite. The metadata remains unlocked until the timestamp occurs, at which point it is then locked.

This issue does not directly impact the overall security posture and is included for informational purposes only.

Finding Detail

A user would call `asset.setAssetMetadataDetails` with a `details` argument like the following in order to enable the feature.

```
var locked_until_details = {expire: null, lockStatus: {LockedUntil: [timestamp] }};
```

Example Details to Set for the `LockedUntil` Feature

The `is_locked` expression for the `LockedUntil` case will return an unlocked status before the timestamp expiration and then return a locked status after expiration, as shown below.

```
impl<Moment: PartialOrd> AssetMetadataLockStatus<Moment> {
    /// Check if the lock status is locked.
    pub fn is_locked(&self, now: Moment) -> bool {
        match self {
            Self::Unlocked => false,
            Self::Locked => true,
            Self::LockedUntil(until) => now > *until,
        }
    }
}
```

`is_locked` Implementation in `primitives/src/asset_metadata.rs`

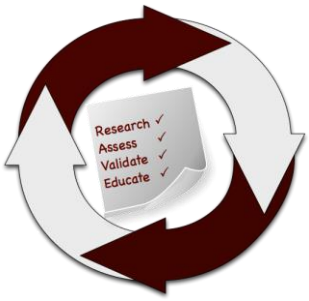
Recommendation(s)

The boolean expression should be inverted, so that it is true when `until` is greater than or equal to `now`.

References

CWE-841: Improper Enforcement of Behavioral Workflow:
https://cwe.mitre.org/data/definitions/841.html

Appendix I: Assessment Methodology



Atredis Partners draws on our extensive experience in penetration testing, reverse engineering, hardware/software exploitation, and embedded systems design to tailor each assessment to the specific targets, attacker profile, and threat scenarios relevant to our client's business drivers and agreed upon rules of engagement.

Where applicable, we also draw on and reference specific industry best practices, regulations, and principles of sound systems and software design to help our clients improve their products while simultaneously making them more stable and secure.

Our team takes guidance from industry-wide standards and practices such as the National Institute of Standards and Technology's (NIST) Special Publications, the Open Web Application Security Project (OWASP), and the Center for Internet Security (CIS).

Throughout the engagement, we communicate findings as they are identified and validated, and schedule ongoing engagement meetings and touchpoints, keeping our process open and transparent and working closely with our clients to focus testing efforts where they provide the most value.

In most engagements, our primary focus is on creating purpose-built test suites and toolchains to evaluate the target, but we do utilize off-the-shelf tools where applicable as well, both for general patch audit and best practice validation as well as to ensure a comprehensive and consistent baseline is obtained.

Research and Profiling Phase

Our research-driven approach to testing begins with a detailed examination of the target, where we model the behavior of the application, network, and software components in their default state. We map out hosts and network services, patch levels, and application versions. We frequently use a number of private and public data sources to collect Open Source Intelligence about the target, and collaborate with client personnel to further inform our testing objectives.

For network and web application assessments, we perform network and host discovery as well as map out all available application interfaces and inputs. For hardware assessments, we study the design and implementation, down to a circuit-debugging level. In reviewing source code or compiled application code, we map out application flow and call trees and develop a solid working understand of how the application behaves, thus helping focus our validation and testing efforts on areas where vulnerabilities might have the highest impact to the application's security or integrity.

Analysis and Instrumentation Phase

Once we have developed a thorough understanding of the target, we use a number of specialized and custom-developed tools to perform vulnerability discovery as well as binary, protocol, and runtime analysis, frequently creating engagement-specific software tools which we share with our clients at the close of any engagement.

We identify and implement means to monitor and instrument the behavior of the target, utilizing debugging, decompilation and runtime analysis, as well as making use of memory and filesystem forensics analysis to create a comprehensive attack modeling testbed. Where they exist, we also use common off-the-shelf, open-source and any extant vendor-proprietary tools to aid in testing and evaluation.

Validation and Attack Phase

Using our understanding of the target, our team creates a series of highly-specific attack and fault injection test cases and scenarios. Our selection of test cases and testing viewpoints are based on our understanding of which approaches are most relevant to the target and will gain results in the most efficient manner, and built in collaboration with our client during the engagement.

Once our test cases are validated and specific attacks are confirmed, we create proof-of-concept artifacts and pursue confirmed attacks to identify extent of potential damage, risk to the environment, and reliability of each attack scenario. We also gather all the necessary data to confirm vulnerabilities identified and work to identify and document specific root causes and all relevant instances in software, hardware, or firmware where a given issue exists.

Education and Evidentiary Phase

At the conclusion of active testing, our team gathers all raw data, relevant custom toolchains, and applicable testing artifacts, parses and normalizes these results, and presents an initial findings brief to our clients, so that remediation can begin while a more formal document is created. Additionally, our team shares confirmed high-risk findings throughout the engagement so that our clients may begin to address any critical issues as soon as they are identified.

After the outbrief and initial findings review, we develop a detailed research deliverable report that provides not only our findings and recommendations but also an open and transparent narrative about our testing process, observations and specific challenges in developing attacks against our targets, from the real world perspective of a skilled, motivated attacker.

Automation and Off-The-Shelf Tools

Where applicable or useful, our team does utilize licensed and open-source software to aid us throughout the evaluation process. These tools and their output are considered secondary to manual human analysis, but nonetheless provide a valuable secondary source of data, after careful validation and reduction of false positives.

For runtime analysis and debugging, we rely extensively on Hopper, IDA Pro and Hex-Rays, as well as platform-specific runtime debuggers, and develop fuzzing, memory analysis, and other testing tools primarily in Ruby and Python.

In source auditing, we typically work in Visual Studio, Xcode and Eclipse IDE, as well as other markup tools. For automated source code analysis we will typically use the most appropriate toolchain for the target, unless client preference dictates another tool.

Network discovery and exploitation make use of Nessus, Metasploit, and other open-source scanning tools, again deferring to client preference where applicable. Web application runtime analysis relies

extensively on the Burp Suite, Fuzzer and Scanner, as well as purpose-built automation tools built in Go, Ruby and Python.

Engagement Deliverables

Atredis Partners deliverables include a detailed overview of testing steps and testing dates, as well as our understanding of the specific risk profile developed from performing the objectives of the given engagement.

In the engagement summary we focus on "big picture" recommendations and a high-level overview of shared attributes of vulnerabilities identified and organizational-level recommendations that might address these findings.

In the findings section of the document, we provide detailed information about vulnerabilities identified, provide relevant steps and proof-of-concept code to replicate these findings, and our recommended approach to remediate the issues, developing these recommendations collaboratively with our clients before finalization of the document.

Our team typically makes use of both DREAD and NIST CVE for risk scoring and naming, but as part of our charter as a client-driven and collaborative consultancy, we can vary our scoring model to a given client's preferred risk model, and in many cases will create our findings using the client's internal findings templates, if requested.

Sample deliverables can be provided upon request, but due to the highly specific and confidential nature of Atredis Partners' work, these deliverables will be heavily sanitized, and give only a very general sense of the document structure.

Appendix II: Engagement Team Biographies

Shawn Moyer, Founding Partner and CEO

Shawn Moyer scopes, plans, and coordinates security research and consulting projects for the Atredis Partners team, including reverse engineering, binary analysis, advanced penetration testing, and private vulnerability research. As CEO, Shawn works with the Atredis leadership team to build and grow the Atredis culture, making Atredis Partners a home for some of the best minds in information security, and ensuring Atredis continues to deliver research and consulting services that exceed our client's expectations.

Experience

Shawn brings over 25 years of experience in information security, with an extensive background in penetration testing, advanced security research including extensive work in mobile and Smart Grid security, as well as advanced threat modeling and embedded reverse engineering.

Shawn has served as a team lead and consultant in enterprise security for numerous large initiatives in the financial sector and the federal government, including IBM Internet Security Systems' X-Force, MasterCard, a large Federal agency, and Wells Fargo Securities, all focusing on emerging network and application attacks and defenses.

In 2010, Shawn created Accuvant Labs' Applied Research practice, delivering advanced research-driven consulting to numerous clients on mobile platforms, critical infrastructure, medical devices and countless other targets, growing the practice 1800% in its first year.

Prior to Accuvant, Shawn helped develop FishNet Security's penetration testing team as a principal security consultant, growing red team offerings and advanced penetration testing services, while being twice selected as a consulting MVP.

Key Accomplishments

Shawn has written on emerging threats and other topics for Information Security Magazine and ZDNet, and his research has been featured in the Washington Post, BusinessWeek, NPR and the New York Times. Shawn is a twelve-time speaker at the Black Hat Briefings and has been an invited speaker at other notable security conferences around the world.

Shawn is likely best known for delivering the first public research on social network security, pointing out much of the threat landscape still exists on social network platforms today. Shawn also co-authored an analysis of the state of the art in web browser exploit mitigation, creating the first in-depth comparison of browser security models along with Dr. Charlie Miller, Chris Valasek, Ryan Smith, Joshua Drake, and Paul Mehta.

Shawn studied Computer and Network Information Systems at Missouri University and the University of Louisiana at Lafayette, holds numerous information security certifications, and has been a frequent presenter at national and international security industry conferences.

Bryan C. Geraghty, Principal Research Consultant

Bryan leads and executes highly technical application and network security assessments, as well as adversarial simulation assessments. He specializes in cryptography and reverse engineering.

Experience

Bryan has over 20 years of experience building and exploiting networks, software, and hardware systems. His deep background in systems administration, software development, and cryptography has been demonstrably beneficial for security assessments of custom or unique applications in industries such as healthcare, manufacturing, marketing, banking, utilities, and entertainment.

Key Accomplishments

Bryan is a creator and maintainer of several open-source security tools. He is also a nationally recognized speaker; often presenting research on topics such as software, hardware, and communications protocol attacks, and participating in offense-oriented panel discussions. Bryan is also an organizing-board member of multiple Kansas City security events, and a staff volunteer & organizer of official events at DEF CON.

Sean Bradly, Principal Research Consultant

Sean Bradly is an expert security researcher with 20 years of experience in general software development and nearly 15 years with a focus on security. He has spent many of these years researching, auditing, reverse engineering, exploiting, designing, implementing, maintaining, and delivering both software and hardware pertaining to all manner of subject matter.

Experience

Sean has held many roles within the industry, starting in the year 2000 as a junior programmer and quickly moving into other realms such as systems automation, embedded development, security engineering, and security consulting.

Sean got his start in computer security while developing an automated network vulnerability scanning service (TrustWatch) in 2006. He then went on to BreakingPoint Systems where he designed network testing software, writing network protocol simulators and exploit traffic generators with focus on supporting both realistic and fuzzed test cases. He also extensively researched security vulnerabilities by hunting for undiscovered bugs, scouring publicly available information, and frequently reverse engineering vendor software update files to craft new exploits for inclusion into the product.

In addition, Sean held the position of Senior Security Consultant with Leviathan Security Group, frequently leading audits on everything from embedded device firmware to web applications as well as building tools to automate analysis and better identify potential security issues.

Most recently before joining Atredis, Sean was a partner at Inverse Limit (InvLim), working on aggressively-paced research and development contracts with clients such as DARPA and Google.

Key Accomplishments

In 2013, Sean authored a custom hypervisor and analysis engine for Project MAIM (part of a DARPA Cyber Fast-Track research grant) to study the differences in CPU instruction sets of different vendors' implementations.

In 2015, Sean designed and implemented an open source, cross platform (ARM/OpenRISC), embedded operating system written from scratch to host sensitive cryptographic applications. In six months, Inverse Limit's team of four were able to deliver custom circuit board with a custom OpenRISC CPU that included accelerated cryptography, the bespoke operating system, and demo applications. This (along with the hardware and other components) was all open sourced as Google's Project Vault and was presented by Pieter Zatko at Google I/O.

Sean also designed and implemented a custom TCP/IP protocol stack for BreakingPoint Systems' Security Engine in order to audit network appliances by realistically simulating attack traffic from tens of thousands of exploits.

Molly Vukusich, Client Operations Associate

Molly Vukusich supports nearly every phase of the project lifecycle at Atredis Partners, from pre-sales, to project planning and management, to project delivery, readout and follow-up. She aims to increase efficiency of project execution and client communication for the benefit of both the consultants and clients.

Experience

Molly has over 11 years of experience in marketing and project management roles in various industries such as Healthcare, Finance, Sports & Recreation, and Non-Profit. Her experience includes copywriting and editing (both technical and promotional), creative strategy development, data analysis, event planning, graphic design, and website management.

Key Accomplishments

Molly earned a bachelor's degree in Mass Communications with an emphasis in Advertising and Public Relations from Oklahoma City University.

Appendix III: About Atredis Partners

Atredis Partners was created in 2013 by a team of security industry veterans who wanted to prioritize offering quality and client needs over the pressure to grow rapidly at the expense of delivery and execution. We wanted to build something better, for the long haul.

In six years, Atredis Partners has doubled in size annually, and has been named three times to the Saint Louis Business Journal's "Fifty Fastest Growing Companies" and "Ten Fastest Growing Tech Companies". Consecutively for the past three years, Atredis Partners has been listed on the Inc. 5,000 list of fastest growing private companies in the United States.

The Atredis team is made up of some of the greatest minds in Information Security research and penetration testing, and we've built our business on a reputation for delivering deeper, more advanced assessments than any other firm in our industry.

Atredis Partners team members have presented research over forty times at the BlackHat Briefings conference in Europe, Japan, and the United States, as well as many other notable security conferences, including RSA, ShmooCon, DerbyCon, BSides, and PacSec/CanSec. Most of our team hold one or more advanced degrees in Computer Science or engineering, as well as many other industry certifications and designations. Atredis team members have authored several books, including *The Android Hacker's Handbook*, *The iOS Hacker's Handbook*, *Wicked Cool Shell Scripts*, *Gray Hat C#*, and *Black Hat Go*.

While our client base is by definition confidential and we often operate under strict nondisclosure agreements, Atredis Partners has delivered notable public security research on improving the security at Google, Microsoft, The Linux Foundation, Motorola, Samsung and HTC products, and were the first security research firm to be named in Qualcomm's Product Security Hall of Fame. We've received four research grants from the Defense Advanced Research Project Agency (DARPA), participated in research for the CNCF (Cloud Native Computing Foundation) to advance the security of Kubernetes, worked with OSTIF (The Open Source Technology Improvement Fund) and The Linux Foundation on the Core Infrastructure Initiative to improve the security and safety of the Linux Kernel, and have identified entirely new classes of vulnerabilities in hardware, software, and the infrastructure of the World Wide Web.

In 2015, we expanded our services portfolio to include a wide range of advanced risk and security program management consulting, expanding our services reach to extend from the technical trenches into the boardroom. The Atredis Risk and Advisory team has extensive experience building mature security programs, performing risk and readiness assessments, and serving as trusted partners to our clients to ensure the right people are making informed decisions about risk and risk management.