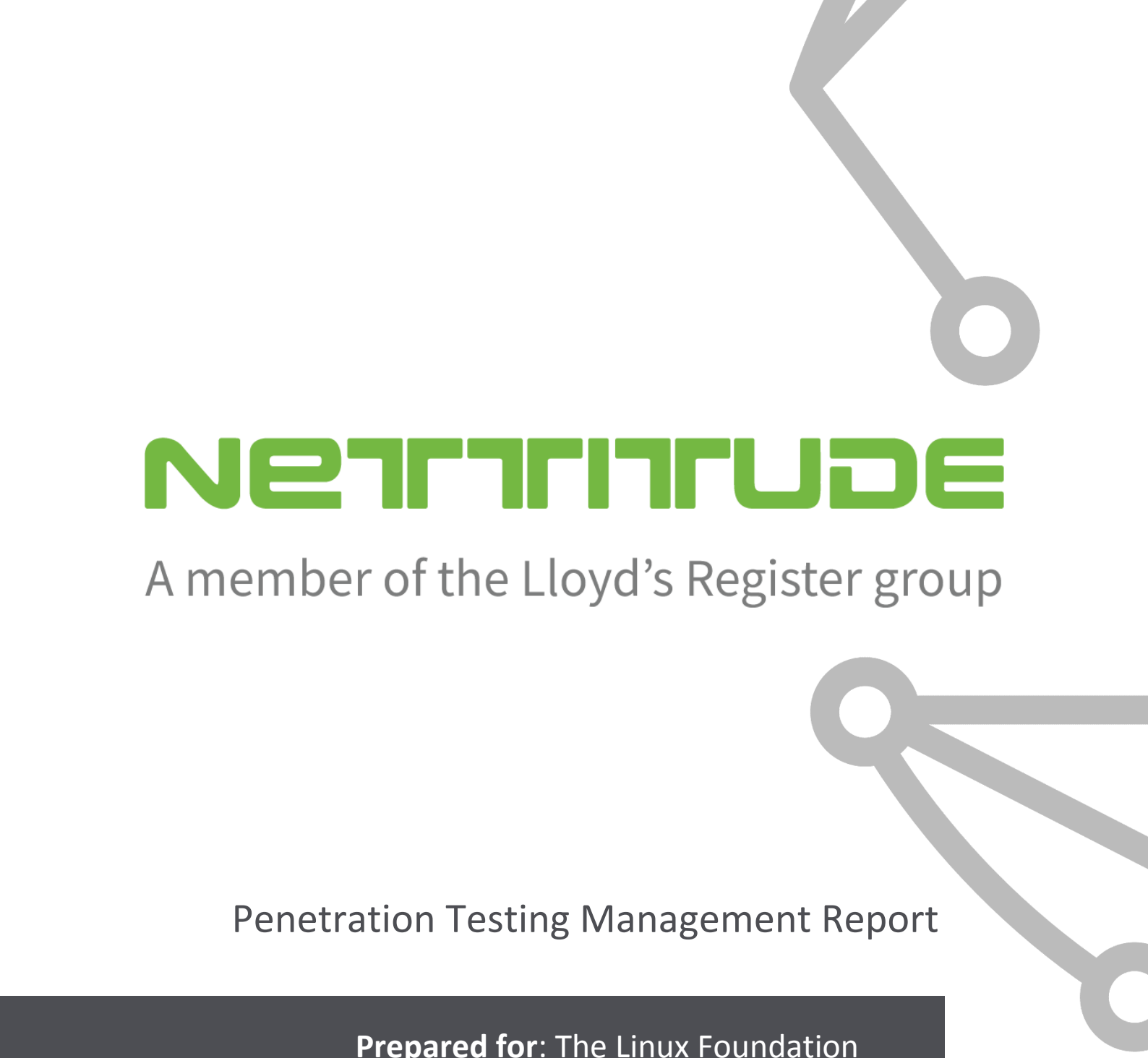


NETITUDE

A member of the Lloyd's Register group

Penetration Testing Management Report

Prepared for: The Linux Foundation
System: Hyperledger Indy
Type: Security Assessment

Author: Graham Shaw
Date: 13 November 2018
Version: 1.0

PCI Security Standards Council

CREST

CHECK
IT Health Check Service

Report Contents

High Level Assessment ____ 3
Overall Security Posture ____ 3
System Analysis ____ 4
Next Steps ____ 6
Distribution List ____ 7
Revision History ____ 7

High Level Assessment

The Linux Foundation engaged with Nettitude in September 2018 in order to assess the overall security posture of their environment.

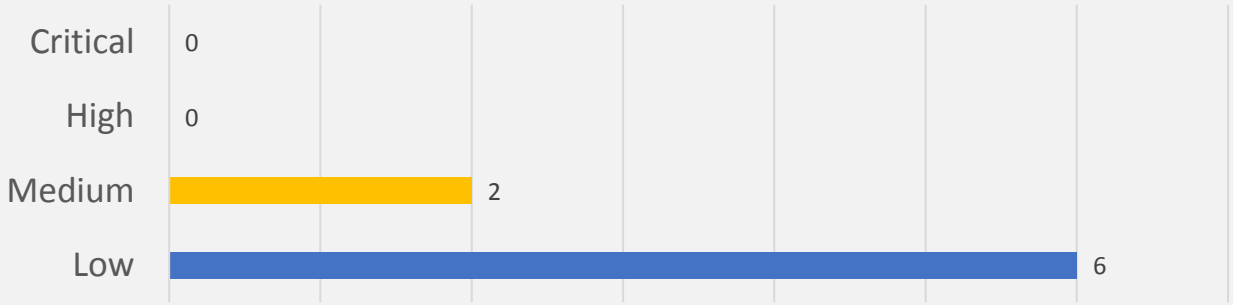
Based on The Linux Foundation's risk profile, primary security concerns and the vulnerabilities identified at the point of the engagement, Nettitude have found the overall security posture to **require moderate attention**.

Nettitude were able to:

- Find cryptographic material generated by a non-cryptographic random number generator
- Find sensitive data not zeroed after use
- Demonstrate that cryptographic operations were not executing in constant time

Overall Security Posture
STRONG
MODERATE
REQUIRES ATTENTION
WEAK

Vulnerabilities by Severity



Limitations and Constraints

No limitations were encountered during the engagement.

System Analysis

Not all random number generators are suitable for use in cryptographic applications, because it is possible for a sequence to have good statistical properties yet still be predictable by a sophisticated attacker. For the most part, the generators used by Hyperledger Indy are fit for purpose, however one instance was found in the function *stp_core.crypto.util.randomSeed* where a non-cryptographic generator had been used inappropriately.

It is good practice in cryptographic code to overwrite sensitive data with zeros when it is no longer needed. Failure to do so is not by itself a vulnerability, since the operating system should ensure the privacy of data in memory. However, if another vulnerability causes that privacy to be compromised, it is better if the sensitive data is not there to be found. Examples of historical vulnerabilities which have allowed read access to memory include Meltdown and Spectre (exploiting the hardware) and Heartbleed (exploiting library code).

This countermeasure is employed to some extent in connection with the encrypted wallet feature of indy-sdk, however its effectiveness will be limited since there are calls to other functions which do not zero sensitive data. It therefore seems likely that the intended level of protection is not being achieved. For other parts of the codebase there is a policy decision to be made, however the ideal would be to perform zeroing throughout.

Also, good practice is for cryptographic code to execute in constant time, in order to prevent timing attacks. For the specific instance analyzed in the technical report, it appears likely that using a more recent version of the amcl library would improve matters. However, this is an issue that would affect many parts of the codebase, and may not always be straightforward to address, so again there is a policy decision to be made.

Other issues found were:

- A potential path-traversal issue in the *DirectoryStore* class
- A potential denial of service issue in the functions *count_bits_set* and *highest_bit_set*

- A race condition in the function *_create_file_with_mode*, which might conceivably allow an attacker to gain access to a private key
- A method by which the *CompactSerializer* class can be induced to throw an unexpected exception
- Unsafe deserialization of *plenum.client.Wallet* (but in code which is already scheduled for removal)
- An off-by-one error in the function *randomString* which causes the result to be slightly less random than it should be

All of these are either low impact, and/or difficult or impossible to exploit as the code in question is used currently. However, they are also relatively straightforward to address, so this is recommended in order to improve the robustness of the code.

Next Steps

Nettitude recommends that The Linux Foundation perform the following post engagement activities in the order of priority indicated.

	Activity	Description	Priority
1	Debrief from Nettitude	Nettitude will deliver a formal debrief to The Linux Foundation in order to ensure that the findings of this engagement have been fully comprehended and to help assist in the formulation of a remediation plan.	++++
2	Cryptographic RNG	Replace use of *random.choice* with a cryptographic random number generator	++
3	Zero memory	Overwrite sensitive data with zeros when it no longer needs to be kept	++
4	Constant-time cryptographic operations	Use cryptographic primitives which execute in constant time	+

Distribution List

Nettitude	Name	Title
	Graham Shaw	Security Consultant
	Jose Lopes	Security Consultant
	Miles Corn	Account Manager

The Linux Foundation	Name	Title
	David Huseby	Security Maven

Revision History

Version	Issue Date	Issued by	Comments
0.1	04 November 2018	Graham Shaw	Initial Draft
0.2	08 November 2018	Jose Lopes	Quality Assurance
0.3	12 November 2018	Miles Corn	Quality Assurance
1.0	13 November 2018	Graham Shaw	Final