July 16, 2025

Contents

Chapter 1

Git Leak, RCE and Privilege Escalation

Git leak vulnerabilities occur when a web server exposes the `.git` directory of a project. Since `.git` contains the entire repository, including source code and history, unauthorized access can lead to disclosure of sensitive information such as API keys, credentials, or proprietary code.

1.1 Detecting Git Leak Vulnerabilities

Common techniques to detect an exposed `.git` directory include:

- Accessing URLs like `http://example.com/.git/` or `http://example.com/.git/config`.
- Checking for HTTP responses indicating directory or file access.
- Using automated tools like `GitTools` to verify exposure.

1.2 Ethical Exploitation of Git Leak Vulnerabilities

If a `.git` directory is accessible, the following steps can be performed to ethically exploit and analyze the leak:

1. **Download the .git directory contents**: Tools such as `GitTools` can automate downloading files recursively from the exposed `.git` directory.
2. **Rebuild the repository**: Using the downloaded `.git` folder, reconstruct the repository by running:
   ```
   git checkout .
   ```
3. **Analyze the codebase**: Search for sensitive data such as passwords, API tokens, or private keys within the source files.

1.3 Using GitTools

`GitTools` is a popular Python tool to download and reconstruct exposed Git repositories.

1.3.1 Installation

Clone the repository and install dependencies:

```
git clone https://github.com/internetwache/GitTools.git
cd GitTools
pip install -r requirements.txt
```

or (recommended)

```
mkdir dump
pip3 install git-dumper
git-dumper http://example.com/.git/ dump
git restore .
```

1.3.2 Downloading an Exposed .git Directory

Run the `gitdumper.py` script:

```
python gitdumper.py http://example.com/.git/ ./local_dump/
```

This downloads the entire exposed Git repository into `./dump`.

1.3.3 Reconstructing the Repository

After downloading:

```
cd local_dump
git checkout .
```

1.4 Preventing Git Leak Vulnerabilities

To avoid accidental exposure of Git data:

- Do **not** deploy the `.git` directory to publicly accessible web roots.
- Configure the web server to block access to `.git` files and directories:
  - **Apache** (`.htaccess`):

    ```
    RedirectMatch 404 /\.git
    ```

  - **Nginx**:

    ```
    location ~ /\.git {
        deny all;
        return 404;
    }
    ```

- Regularly audit deployed files and directories.

1.5 Web Fuzzing with FFUF

When a target doesn't openly expose the `.git` folder in known paths, directory fuzzing can help discover hidden files or misconfigurations.
Web fuzzing is a technique used to discover vulnerabilities in web applications by sending a large number of random or unexpected inputs to the application and observing its behavior. This process helps identify potential security issues such as SQL injection, cross-site scripting (XSS), and buffer overflow attacks.

1.5.1 Using FFUF

`ffuf` (Fuzz Faster U Fool) is a powerful tool to fuzz web directories, parameters, and APIs.
To fuzz for hidden '.git' files:

```
ffuf -w /path/to/SecLists/Discovery/Web-Content/common.txt -u http://
    example.com/FUZZ -mc 200
```

This command tests each word in the wordlist by replacing `FUZZ` in the URL. You can also replace the code returned by the browser in order to search for specific inputs; for example, using 403 (Forbidden) code, it is possible to find users with admin privileges.
You can also explicitly check for '.git' files:

```
ffuf -w /path/to/SecLists/Discovery/Web-Content/git-files.txt -u http
    ://example.com/.git/FUZZ -mc 200
```

If you receive an HTTP 200 status code for paths like '.git/config' or '.git/index', the Git directory is likely exposed.

1.5.2 Common Git Targets

Some important Git files to fuzz for:

- `.git/config`
- `.git/HEAD`
- `.git/index`
- `.git/logs/HEAD`

1.5.3 Using SecLists Wordlists

`SecLists` is a comprehensive collection of wordlists maintained by security professionals and widely used in web fuzzing and enumeration tasks.
To install:

```
git clone https://github.com/danielmiessler/SecLists.git
```

Useful paths include:

- `SecLists/Discovery/Web-Content/common.txt`
- `SecLists/Discovery/Web-Content/raft-medium-directories.txt`
- `SecLists/Discovery/Web-Content/git-files.txt`

These can be used with fuzzers like `ffuf`, `dirsearch`, or `gobuster`.

1.5.4 Example: Fuzzing with a handmade wordlist.txt

```
ffuf -w wordlist.txt -u http://target.com/?q=accounts/FUZZ -mc 403 -c -
    v
```

This fuzzes for finding possible existing users in a webapp.

1.6 Remote Command Execution (RCE)

Remote Code Execution (RCE) is a type of cyberattack where an attacker can execute arbitrary code on a remote system or server, often without the need for user interaction.
This vulnerability allows attackers to gain unauthorized access, potentially leading to data breaches, malware infections, and complete system takeovers.
RCE vulnerabilities typically arise from issues such as unvalidated user input, insecure deserialization, or memory management flaws. Attackers exploit these vulnerabilities to inject malicious code, which can then be executed remotely, often with the same privileges as the compromised process or user.
The consequences of successful RCE attacks can be severe, including data loss, financial losses, and reputational damage.

1.7 Privilege Escalation

Privilege escalation in cybersecurity refers to a type of cyberattack where an attacker gains unauthorized access to elevated permissions within a system, allowing them to perform actions or access information beyond their normal rights. This can occur through various methods, such as exploiting vulnerabilities, misconfigurations, or stolen credentials.
Privilege escalation is a critical step in the cyberattack chain, as it enables attackers to move deeper into a network, potentially leading to data theft, malware deployment, or full system compromise.

There are two primary types of privilege escalation: **horizontal** and **vertical**. Horizontal privilege escalation involves gaining access to another account with similar privileges, often referred to as ”account takeover”. Vertical privilege escalation, on the other hand, involves increasing the privileges of an account to gain access to higher-level permissions, such as administrator or root access.

Attackers may use various techniques to achieve privilege escalation, including credential theft, exploitation of software vulnerabilities, and social engineering. To prevent privilege escalation, organizations should implement the principle of least privilege, regularly update and patch systems, and provide security awareness training. Additionally, utilizing tools like user and entity behavior analytics (UEBA), vulnerability scanners, and multi-factor authentication (MFA) is crucial.

1.7.1 Reverse Shell Upgrading

A reverse shell is a common post-exploitation technique in which a compromised host connects back to the attacker’s machine, typically using an outbound TCP connection. This gives the attacker remote access to the target system. However, such shells are often limited in functionality and need to be upgraded to enable a more interactive and stable terminal session.

Limitations of a Basic Reverse Shell

Basic reverse shells (e.g., initiated using `bash -i`, `nc`, or similar one-liners) do not provide a fully interactive environment. Common limitations include:

- Lack of job control (e.g., no `Ctrl+C` or `Ctrl+Z` support)
- Malfunctioning arrow keys, tab completion, and backspace
- Inability to run full-screen interactive tools like `nano`, `less`, or `htop`

These issues stem from the absence of a pseudo-terminal (PTY). A PTY simulates a terminal interface, which most interactive shells rely on to function properly.

Spawning a Pseudo-Terminal (PTY)

To upgrade the shell, the attacker can spawn a PTY from within the compromised session. This is commonly achieved using Python:

```
python3 -c 'import pty; pty.spawn("/bin/bash")'
```

If Python 3 is not available, Python 2 may work:

```
python -c 'import pty; pty.spawn("/bin/bash")'
```

This command emulates a proper terminal session, enabling features such as command editing and signal handling.

Terminal Configuration on the Attacker Side

After spawning the PTY, further steps are typically required on the attacker's terminal to complete the upgrade:

1. Suspend the reverse shell using `Ctrl+Z`.
2. Run the following command to configure the local terminal:

   ```
   stty raw -echo
   ```

3. Resume the session with:

   ```
   fg
   ```

4. Inside the upgraded shell, set the terminal type:

   ```
   export TERM=xterm
   ```

These steps allow full line editing, proper display formatting, and better command-line behavior.

Upgrading a reverse shell is a critical post-exploitation technique to move from a minimal command execution interface to a more stable and interactive shell. This upgrade relies on emulating a PTY and configuring the local terminal environment correctly. Once upgraded, tasks such as privilege escalation, file editing, and enumeration become much more manageable and efficient.