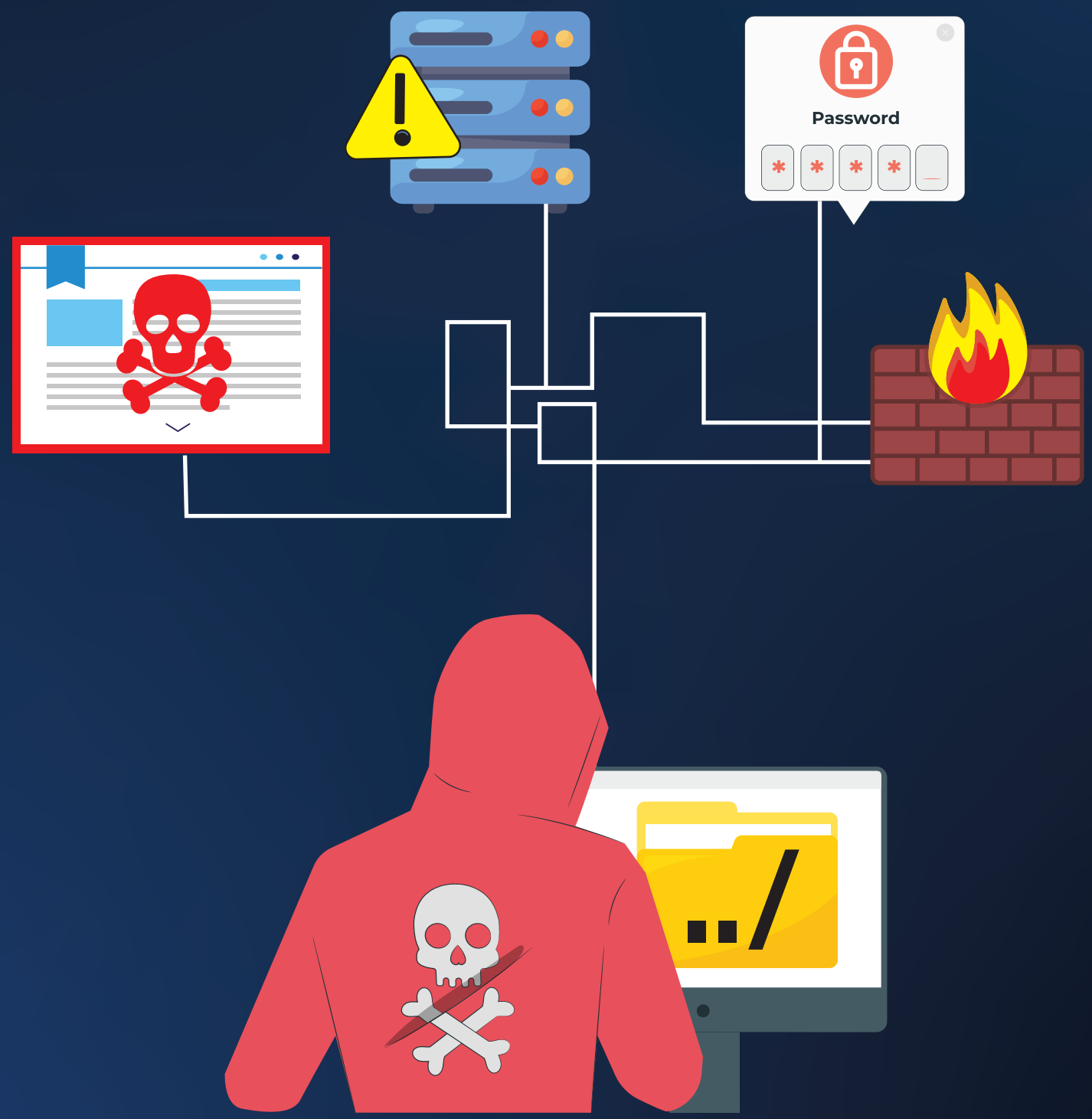
SECURITYBOAT
Frontline Of Your Business

PATH TRAVERSAL
HANDBOOK



www.securityboat.net

SECURITYBOAT
Frontline Of Your Business

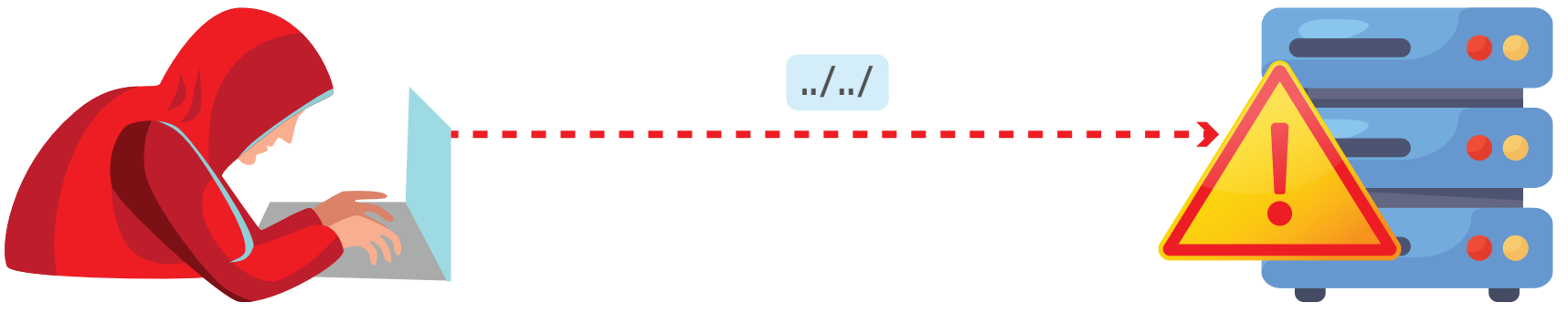
Table of Contents

1. What is path traversal?

A path traversal attack, also known as dot-dot-slash, directory traversal, directory climbing, and backtracking, aims to access files and directories outside the web root folder by manipulating variables referencing files with “dot-dot-slash (../)” sequences or using absolute file paths. This vulnerability enables an attacker to read arbitrary files on the server running an application, including application code and data, credentials for back-end systems, and sensitive operating system files. In some cases, attackers might also write to arbitrary files, modify application data or behavior, and ultimately take full control of the server. However, access to files is limited by system operational access controls, such as locked or in-use files on the Microsoft Windows operating system.

2. How does the path traversal vulnerability occur?

Path traversal vulnerability occurs when web applications, which use locally stored resources like images, scripts, and text files, fail to properly sanitize user parameters and lack sufficient access control. This allows attackers to manipulate request parameters to access and retrieve unauthorized resources. The impact of this vulnerability is typically severe, as attackers can read sensitive files, including configuration files with credentials and keys, access sensitive operating system files, retrieve and analyze the application's source code and server organization, and in some cases, write to the server, potentially altering the application's behavior or gaining full control over the server.



3. Common methods to perform path traversal attacks

3.1 URL Encoding and Decoding

Attackers manipulate URLs by encoding characters such as `../` to `%2e%2e%2f` to bypass input validation mechanisms. By submitting payloads like `%2e%2e%2f%2e%2e%2fetc%2fpasswd`, attackers can trick the server into decoding the string back to its original form, `../../etc/passwd`, allowing them to access restricted directories and files.

3.2 Null Byte Injection

This technique involves appending a null byte (`%00`) to the input string. Many languages use the null byte as a string terminator, which can truncate the rest of the string during processing. For example, a payload like `/../../../../etc/passwd%00.jpg` might bypass filename restrictions by being interpreted as `/../../../../etc/passwd`.

3.3 Double URL Encoding

By double encoding payloads, attackers can bypass some filtering mechanisms that only decode inputs once. For example, `..%252e%252e%252f` is double encoded to first decode to `%2e%2e%2f` and then to `../../`, effectively bypassing filters that do not decode twice.

3.4 Using Encoded Dots and Slashes

Some applications inadequately handle or fail to decode characters like `.` and `/` when they are URL-encoded multiple times. Attackers can exploit this by submitting strings like `%252e%252e%252f` which decode to `../../` in two stages, allowing directory traversal to access sensitive files.

3.5 Path Truncation

When servers impose length restrictions on file paths, attackers can use overly long strings to truncate valid paths. Combining this with directory traversal sequences, they can bypass the intended directory restrictions. For instance, a path like `../` repeated multiple times followed by a long sequence of characters might get truncated to a meaningful traversal sequence.

4. OS specific path traversal payloads

▪ UNIX:

Root directory: `/`

Directory separator: `/`

▪ WINDOWS:

Root directory: `<partition letter>:\`

Directory separator: `/` or `\`

Note: Windows allows filenames to be followed by extra `.`, `\`, or `/` characters.

In many operating systems, null bytes (`%00`) can be injected to terminate the filename. For example, sending a parameter like `?file=secret.doc%00.pdf`

This will result in the Java application seeing a string that ends with .pdf while the operating system will see a file that ends in .doc. Attackers may use this trick to bypass validation routines.

5. Example of a path traversal vulnerability

Request:

```
GET /download?file=../../../../etc/passwd HTTP/1.1
Host: example.com
```

Response:

```
HTTP/1.1 200 OK
Content-Type: text/plain

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
...
```

6. CVEs with path traversal vulnerability

CVE-2021-41773

A directory traversal vulnerability discovered in 2021 in Apache HTTP Server 2.4.49 that could allow remote code execution.

CVE-2018-13379

A directory traversal vulnerability discovered in 2018 in Fortinet FortiOS, the operating system of FortiGate firewalls. This was listed by CISA in 2021 as one of the top routinely exploited vulnerabilities.

CVE-2024-24919

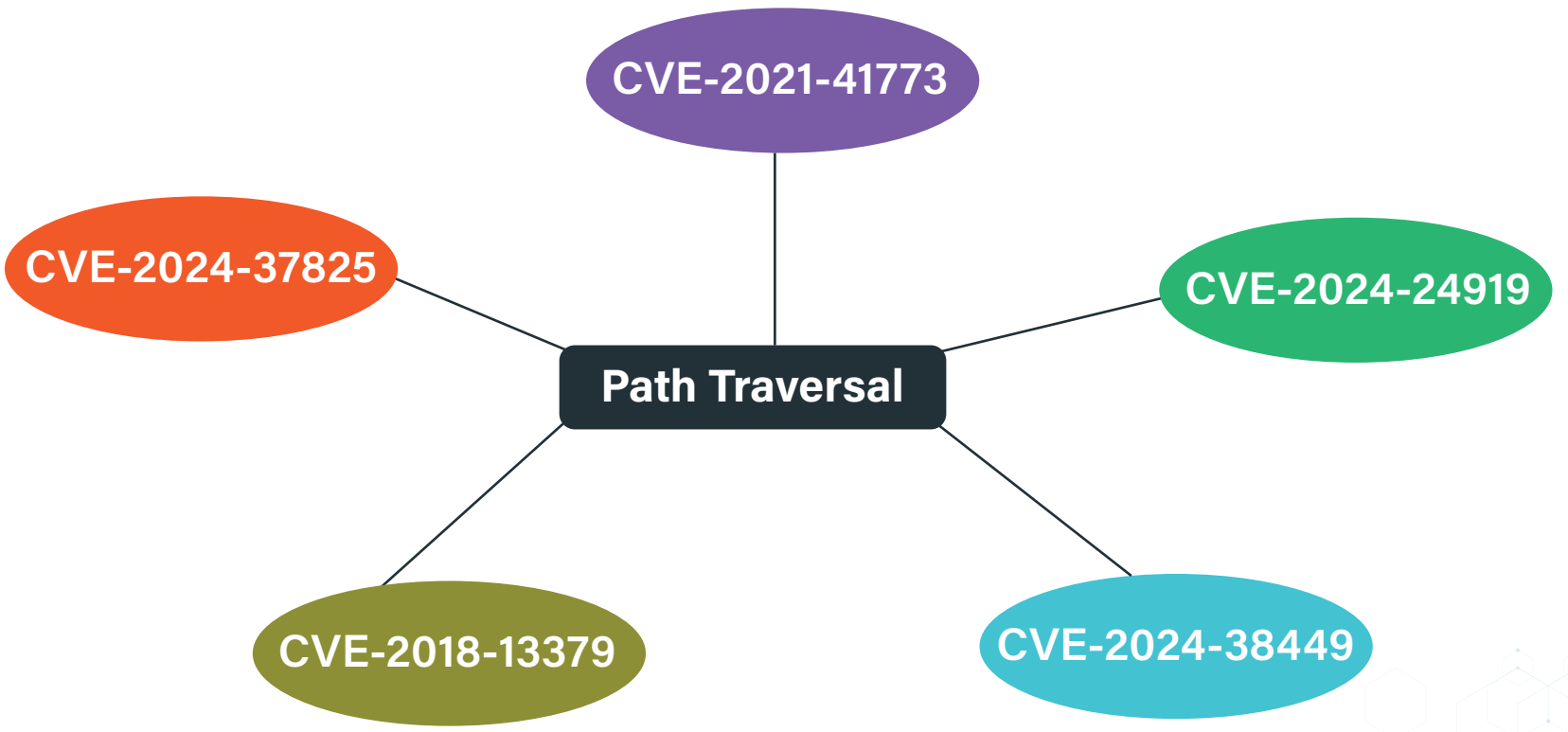
Potentially allowing an attacker to read certain information on Check Point Security Gateways once connected to the internet and enabled with remote Access VPN or Mobile Access Software Blades.

CVE-2024-38449

A Directory Traversal vulnerability in KasmVNC 1.3.1.230e50f7b89663316c70de7b0e3db6f6b9340489 and possibly earlier versions allows remote authenticated attackers to browse parent directories and read the content of files outside the scope of the application.

CVE-2024-37825

An issue in EnvisionWare Computer Access & Reservation Control SelfCheck v1.0 (fixed in OneStop 3.2.0.27184 Hotfix May 2024) allows unauthenticated attackers on the same network to perform a directory traversal.



7. Impact

- **Unauthorized Access to Files:** Attackers can gain access to sensitive files such as configuration files, password files (e.g., /etc/passwd), and other sensitive data stored on the server.
- **Disclosure of Sensitive Information:** Confidential information such as user data, intellectual property, and business-critical information can be exposed.
- **System Compromise:** Accessing system files may allow attackers to gather information necessary to compromise the server, such as credentials or scripts that can be exploited.
- **Log File Access:** Access to log files can help attackers understand the system's operations, aiding in further attacks and making it easier to evade detection.
- **Credential Theft:** Path traversal can be used to access files containing credentials, such as database connection strings or API keys, leading to further compromise.
- **Data Corruption:** If writable files are accessed, attackers can corrupt data, insert malicious code, or change application behavior.
- **Reputation Damage:** Exposure of sensitive information can lead to loss of customer trust and significant damage to the organization's reputation.
- **Intellectual Property Theft:** Confidential business information, proprietary algorithms, and other intellectual properties can be stolen and misused.
- **Compliance Violations:** Unauthorized access to sensitive data can result in violations of regulations and standards, leading to legal penalties and fines.

8. Mitigation

Input Validation and Sanitization:

- **Strict Whitelisting:** Only allow filenames that match a strict whitelist of safe characters and patterns.
- **Reject Suspicious Input:** Reject any input containing sequences like `../`, `..%2f`, and other encodings of `..`.

Canonicalization:

- **Resolve Paths:** Always resolve paths to their canonical form before using them. This process involves converting relative paths to absolute paths and removing any redundant elements.
- **Use Secure Functions:** Utilize secure APIs or libraries that automatically handle path normalization and canonicalization.

Access Control:

- **Restrict Access:** Ensure that the application only has access to the directories and files it needs. Implement least privilege principles.
- **Sandboxing:** Run applications with minimal privileges and in isolated environments to limit the impact of a potential compromise.

File System Permissions:

- **Enforce Strong Permissions:** Set appropriate file permissions to prevent unauthorized access. Ensure sensitive files are not readable by the web server or application user.
- **Separate User Accounts:** Use separate user accounts for different services to limit access scope.

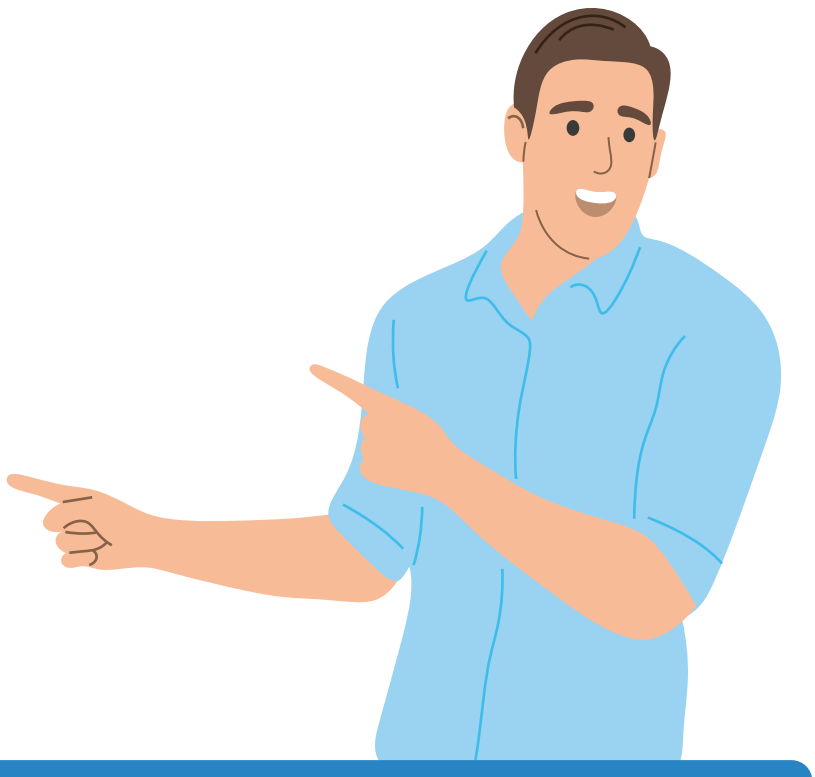
Web Application Firewalls (WAF) and Configurations:

- **Deploy a WAF:** Use a WAF to detect and block malicious requests, including those attempting path traversal attacks.
- **Disable Directory Listings:** Ensure web server configurations do not allow directory listings.

SECURITYBOAT

Frontline Of Your Business

PATH TRAVERSAL HANDBOOK



Scan QR Code to Download Handbook

www.securityboat.net