Sri Lanka Institute of Information Technology

Web Security - IE2062

Bug Bounty Report 4

PERERA A.P.J

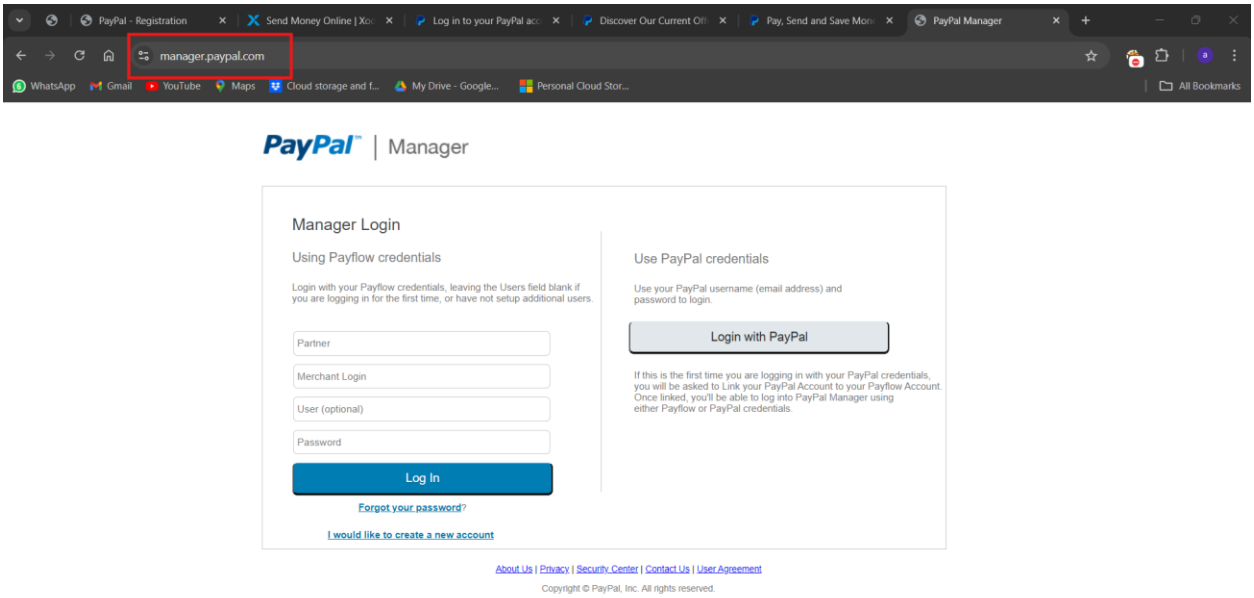
IT22280992

Group Y2S2.CS

SLIIT
Discover Your Future

Table of Contents

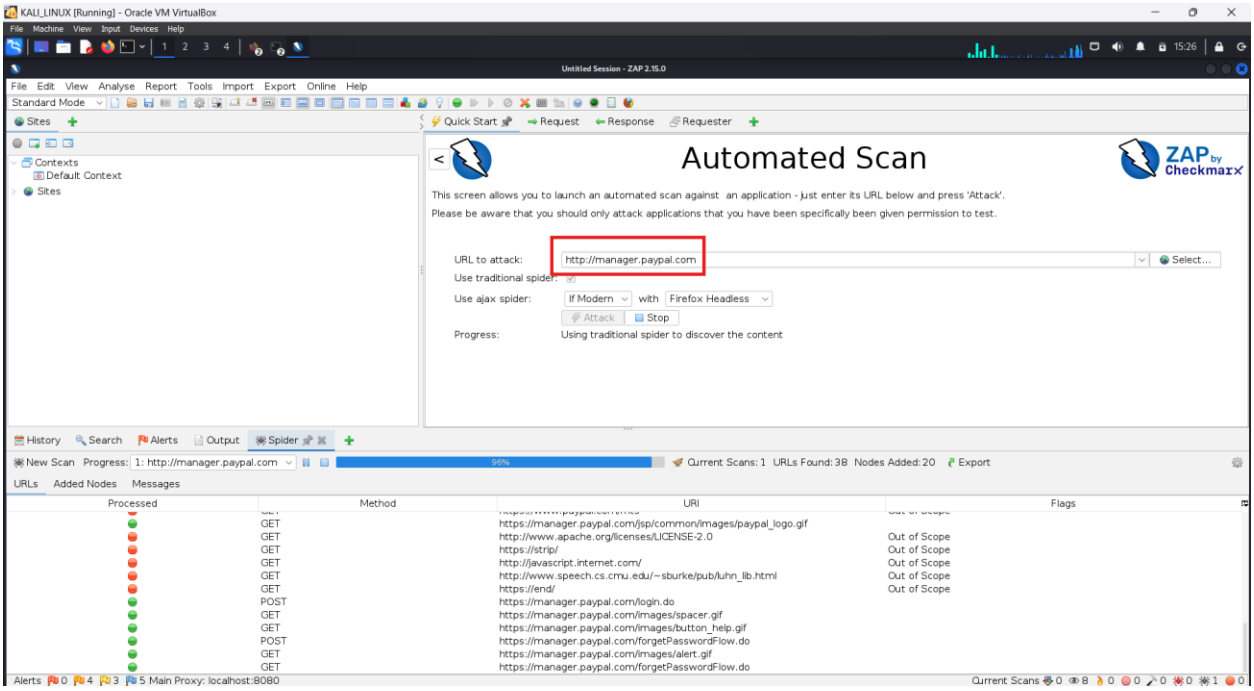
1. Target: http://manager.paypal.com

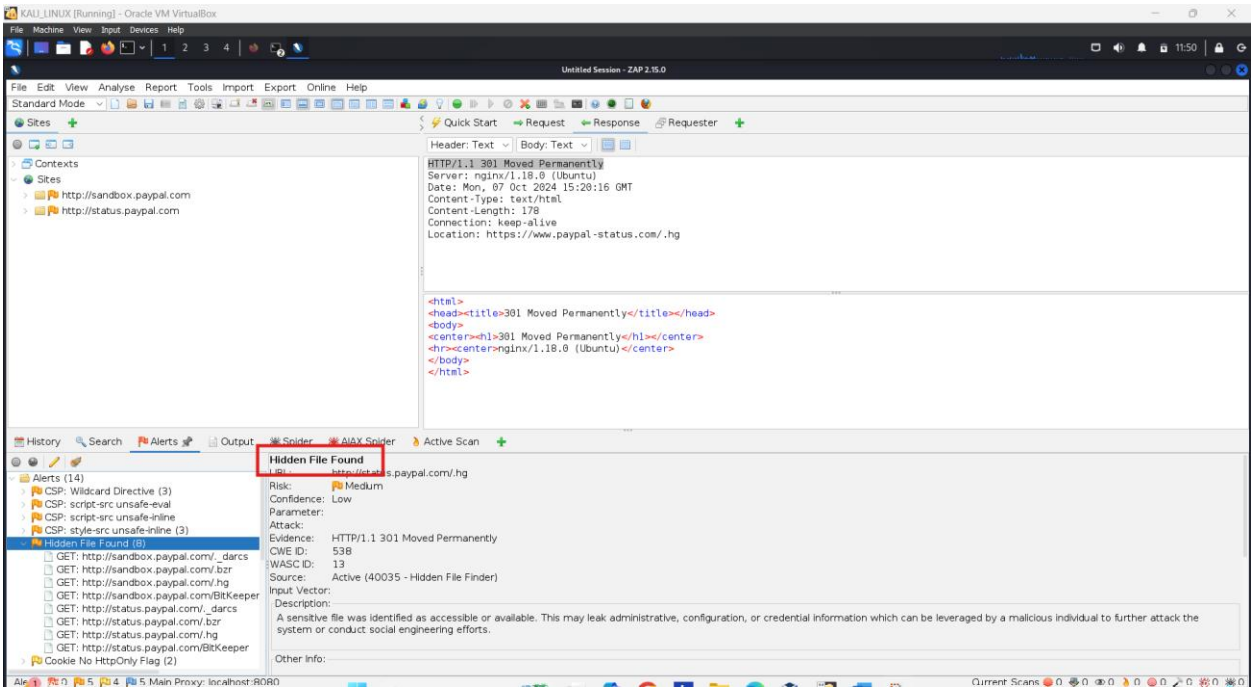


1.1 Vulnerability

1.2 Vulnerability title

- Hidden File Found




KALI_LINUX [Running] - Oracle VM VirtualBox
Untitled Session - ZAP 2.15.0
File Edit View Analyse Report Tools Import Export Online Help
Standard Mode
Sites
Quick Start
Request
Response
Requester
Header: Text
Body: Text
Contexts
Sites
http://sandbox.paypal.com
http://status.paypal.com
HTTP/1.1 301 Moved Permanently
Server: nginx/1.18.0 (Ubuntu)
Date: Mon, 07 Oct 2024 15:20:16 GMT
Content-Type: text/html
Content-Length: 178
Connection: keep-alive
Location: https://www.paypal-status.com/.hg
<html>
<head><title>301 Moved Permanently</title></head>
<body>
<h1>301 Moved Permanently</h1>
<hr> nginx/1.18.0 (Ubuntu)
</body>
</html>
History
Search
Alerts
Output
Spider
AJAX Spider
Active Scan
Alerts (14)
CSP: Wildcard Directive (3)
CSP: script-src unsafe-eval
CSP: script-src unsafe-inline
CSP: style-src unsafe-inline (3)
Hidden File Found (8)
GET: http://sandbox.paypal.com/._darcs
GET: http://sandbox.paypal.com/.bzr
GET: http://sandbox.paypal.com/.hg
GET: http://sandbox.paypal.com/BitKeeper
GET: http://status.paypal.com/._darcs
GET: http://status.paypal.com/.bzr
GET: http://status.paypal.com/.hg
GET: http://status.paypal.com/BitKeeper
Cookie No HttpOnly Flag (2)
Hidden File Found
Risk: Medium
Confidence: Low
Parameter:
Attack:
Evidence: HTTP/1.1 301 Moved Permanently
CWE ID: 538
WASC ID: 13
Source: Active (40035 - Hidden File Finder)
Input Vector:
Description:
A sensitive file was identified as accessible or available. This may leak administrative, configuration, or credential information which can be leveraged by a malicious individual to further attack the system or conduct social engineering efforts.
Other Info:

1.3 Vulnerability description

- The term "Hidden File Found" vulnerability pertains to the misplaced existence of files that were meant to be concealed from users because of various factors, such as configuration errors, poor programming, or oversight. This weakness could expose sensitive files like configuration files, backups, or any files with invaluable information to unauthorized individuals. Such documents can be used by attackers to infiltrate or gather intelligence making the subsequent attacks easy. It frequently occurs that programmers fail to delete files with extensions .bak or .old and .backup which often contain some configuration settings or passwords. Furthermore, due to inadequate file control and security procedures, .git files, .env files, or even edge case temporary log files can be left open. Directory listing permission enabled, the attacker can also directly look into the directory structure that contains those files. Also, these files are more vulnerable to being accessed without permission due to poor access control mechanisms in place, and are therefore more susceptible to being compromised.

1.4 Affected components

The "Hidden File Found" vulnerability can affect various components within a web application or server environment. Here are the main affected components

1. **Web Server Configuration**: Misconfigurations in web server settings can lead to unauthorized access to hidden or backup files. This includes incorrect permissions or exposing directories that should be restricted.
2. **File Systems**: The underlying file system can contain sensitive files (e.g., .bak, .old, .env, or logs) that are inadvertently left accessible to the public.
3. **Source Code Repositories**: If version control directories (like .git) are left exposed, they can reveal the entire source code, including sensitive configuration files and credentials.
4. **Application Logic**: Poorly designed application logic may allow users to access hidden or administrative files unintentionally, especially if proper validation is not enforced.
5. **Directory Listings**: When directory listing is enabled, attackers can view all files in a directory, making it easy to find and exploit hidden files.
6. **Backup Systems**: Backup systems or services that retain older versions of files may inadvertently expose sensitive data if not configured properly.
7. **Third-Party Libraries**: Some third-party libraries or frameworks may create temporary files or backups that could become exposed due to inadequate security measures.
8. **Cloud Storage and Services**: Misconfigured cloud storage settings can also lead to hidden files being accessible if appropriate access controls are not implemented.

1.5 Impact assessment

The impact assessment of the "Hidden File Found" vulnerability can vary depending on the specific context and the sensitivity of the exposed files. Here are the potential impacts:

1. **Data Breach**: Unauthorized access to sensitive files can lead to data breaches, exposing personally identifiable information (PII), financial data, or proprietary information, which can have severe legal and financial consequences.
2. **System Compromise**: Attackers may gain access to configuration files or credentials, allowing them to compromise the entire system, escalate privileges, or move laterally within the network.
3. **Reputation Damage**: A data breach or exposure of sensitive information can harm an organization's reputation, leading to loss of customer trust and potentially impacting business relationships.
4. **Regulatory Fines and Legal Consequences**: Organizations may face regulatory scrutiny and fines for failing to protect sensitive data, particularly in industries governed by strict regulations like healthcare (HIPAA) or finance (PCI DSS).
5. **Operational Disruption**: Exposed files can lead to service disruptions, whether through direct attacks or as part of a broader compromise, affecting business continuity and productivity.
6. **Financial Loss**: The costs associated with remediation, legal fees, and potential fines can be significant. Additionally, organizations may incur costs related to customer notification and credit monitoring services.
7. **Intellectual Property Theft**: If proprietary code or designs are exposed, it could lead to theft of intellectual property, giving competitors an unfair advantage and undermining innovation.
8. **Increased Attack Surface**: The presence of hidden files can increase the attack surface of an application, providing attackers with more potential points of exploitation.
9. **Long-Term Security Risks**: Even if immediate impacts are mitigated, the presence of hidden files may indicate broader security misconfigurations, leading to long-term vulnerabilities in the system.

1.6 Steps to reproduce

1. **Set Up a Web Server**:
   - Deploy a web server (e.g., Apache, Nginx, or IIS) on a local machine or virtual environment.
   - Ensure it's configured to serve files from a specific directory.

2. **Create Sensitive Files**:
   - Generate files with sensitive information, such as:
     - Configuration files (e.g., config.php, settings.json)
     - Backup files (e.g., backup.zip, database.bak)
     - Log files (e.g., debug.log, error.log)
   - Place these files in the web server's document root or a subdirectory.

3. **Modify Web Server Configuration**:
   - Adjust the web server configuration to allow access to hidden or backup files. This could involve:
     - Removing restrictions on file extensions or directory access.
     - Enabling directory listing, allowing users to view all files in a directory.

4. **Access Files via URL**:
   - Using a web browser or a tool like curl or wget, attempt to access the sensitive files directly via their URLs (e.g., http://localhost/config.php or http://localhost/backup.zip).
   - Check if the files can be accessed without authentication or authorization.

5. **Verify Directory Listing** (if applicable):

- If directory listing is enabled, navigate to the directory in the web browser (e.g., http://localhost/backups/) and confirm that the sensitive files are visible and accessible.

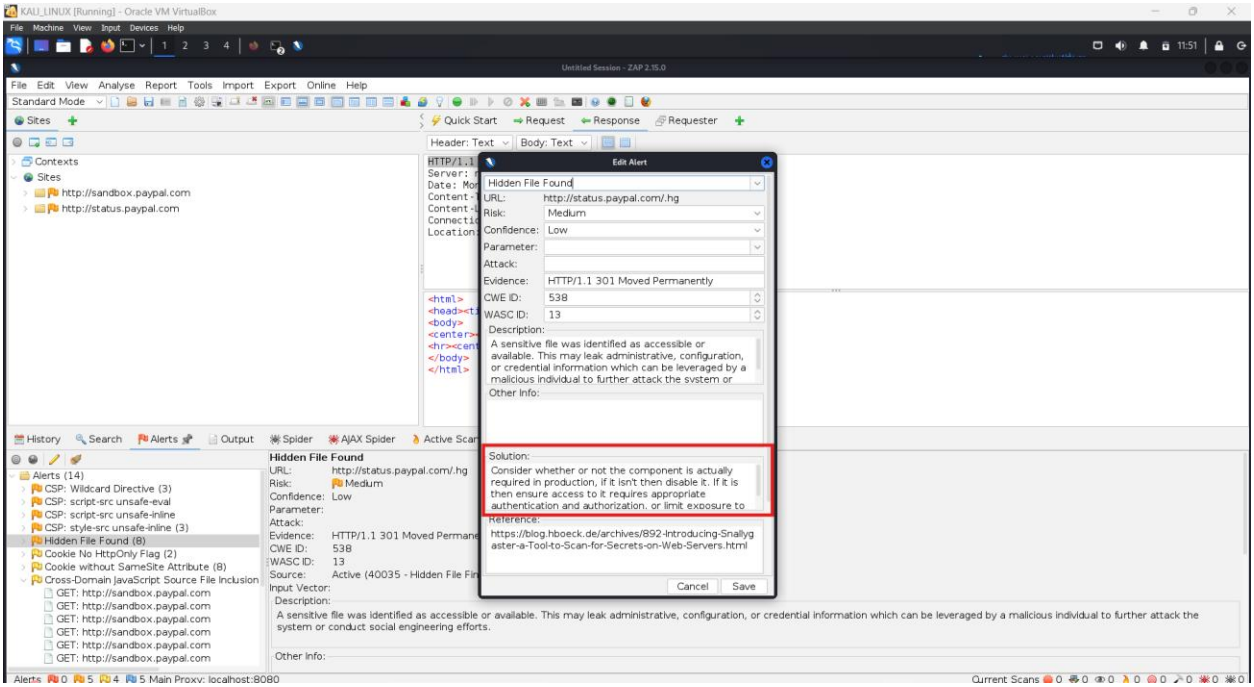
6. **Assess Access Controls**:

   - Analyze the web server's access controls to determine if any sensitive files are unintentionally exposed. This can include checking for improperly set permissions or exposed directories.

7. **Document Findings**:

   - Record the steps taken, the files exposed, and any sensitive information accessed during the test.
   - Consider taking screenshots or logs for documentation purposes.

1.7 Proof of concept (if applicable)



1.8 Proposed mitigation or fix

- Consider the Whether or not the components is actually required in production ,if it isn't then disable. If it is then ensure access, it it requires appropriate authentication and authorization.
- Secure Serer configuration.
- Implement Access Coontrol.
- File naming and Location Practices.
- Regular Audits and Scan.
- Monitor logs.
- Data Encryption.
- Web application Firewall.
- User training awareness.