Vulnerable Web Application

Odyssey IthacaLabs

JavaScript and Obfuscation

JavaScript (JS)

- Client Side Scripting Language
- Source code is processed by the client's web browser rather than on the web server

Obfuscation

- source code or machine code that is difficult for humans to understand
- similar to encryption
- machine can understand the code and is able to execute it.

Available online JS Deobfuscators and Beautifiers:

- **Javascriptbeautifier.com**
- **Jsbeautifier.org**
- **Jspretty.com**
- **JS::Beautify**
- **Javascriptformat.com**

Obfuscated Code Sample

JavaScript Clear Code:

```
// Paste your JavaScript code here
function hi() {
  console.log("Hello World!");
}
hi();
```

JavaScript Obfuscated Code:

```
var _0x5e83=['Hello\x20World!'];(function(_0x1f12f2,_0x396141){var _0x3800fe=function(_0x3e46ba){while(--_0x3e46ba){_0x1f12f2['push']
(_0x1f12f2['shift']());}};_0x3800fe(++_0x396141);}(_0x5e83,0x11a));var _0x57bf=function(_0x2b575f,_0x11ae9e){_0x2b575f=_0x2b575f-0x0;var
_0x5475be=_0x5e83[_0x2b575f];return _0x5475be;};function hi(){console['log'](_0x57bf('0x0'));}hi();
```

Available Online Tool: **https://obfuscator.io/**

Authentication

Client Side Authentication

- Authentication checks are performed completely at users' side
- This has never has been secure
  - Malicious user can perform 'white box' testing and look deep into the codes for vulnerabilities.
  - Chances of authentication bypass, sensitive information disclosure and credentials leakage are extremely high

Server Side Authentication

- Solution in the above problem
- BlackBox Testing

OS Command Injection

OS command injection

- Known as shell injection
- Security vulnerability
  - Allows an attacker to execute arbitrary operating system (OS) commands on the server that is running an application
  - Typically fully compromise the host and all its data.

Injection attacks

- Are possible when an application passes unsafe user-supplied data (forms, cookies, HTTP headers, and so on) to a system shell
- The attacker-supplied OS commands are usually executed with the privileges of the vulnerable application

Injection Attacks and Impacts

Code injection
- Full system compromise

CRLF injection
- Cross-site Scripting (XSS)

Cross-site Scripting (XSS)
- Account impersonation
- Defacement
- Run arbitrary JavaScript in the victim's browse

Email Header Injection
- Spam relay
- Information disclosure

Host Header Injection
- Password-reset poisoning
- Cache poisoning

LDAP Injection
- Authentication bypass
- Privilege escalation
- Information disclosure

OS Command Injection
- Full system compromise

SQL Injection (SQLi)
- Authentication bypass
- Information disclosure
- Data loss
- Sensitive data theft
- Loss of data integrity
- Denial of service
- Full system compromise

XPath injection
- Information disclosure
- Authentication bypass

Prevent Command Injection

Do not “exec” out to the Operating System if it can be avoided.

Validate untrusted inputs - “whitelist validation”

- Input Validation:
  - Character set
  - Minimum and maximum length
  - Numeric bounds
  - Date bounds
  - Match to a Regular Expression Pattern
  - Membership in a discrete set (e.g. US States, list of colors, salutations, etc.)

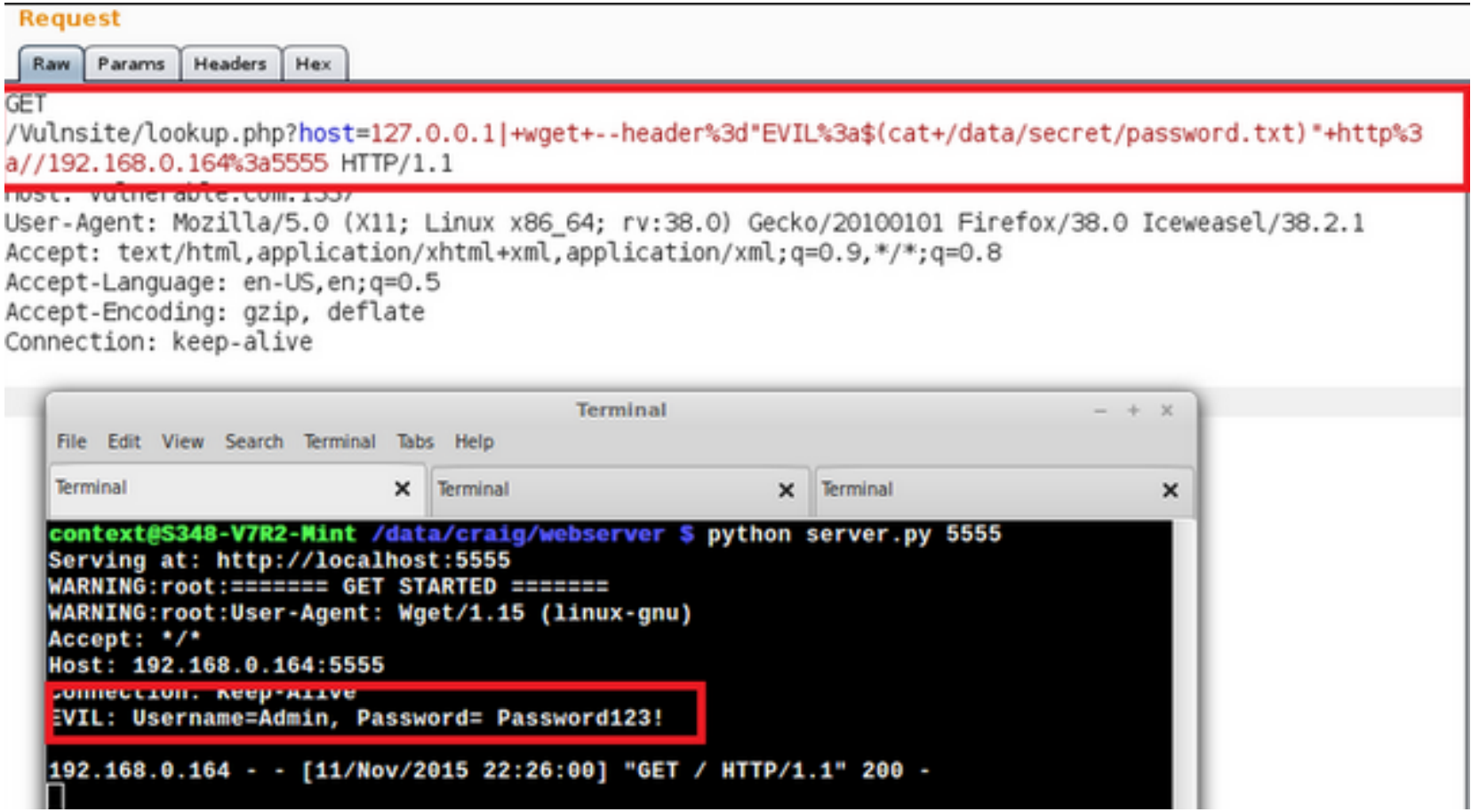
Neutralize meta-characters that have meaning in the target OS command-line:

- For Windows: neutralize its special meaning to the command-line interpreter: ( ) < > & * ‘ | = ? ; [ ] ^ ~ ! . ” % @ / \ : + , `
- For Linux and Unix: neutralize its special meaning to the command-line interpreter: { } ( ) < > & * ‘ | = ? ; [ ] $ – # ~ ! . ” % / \ : + , `

Implement “Least Privilege”

- Not prevent or avoid Command Injection vulnerabilities
- Restrict the power (permissions) of the account used to execute OS commands
- Mitigate the potential damage

Injection Attack Sample



Types and Prevention of Linux Privilege Escalation

Types:

- Kernel exploits
- Exploiting services which are running as root
- Exploiting SUID Executables
- Exploiting SUDO rights/user
- Exploiting badly configured cron jobs
- Exploiting users with ‘.’ in their PATH

Prevention:

- Patching and Updating System
- Right Use of the above exploitable points.

Linux Privilege Escalation with SUDO Rights

Linux Privilege escalation can be performed by running bash through SUDO processes

TTY

- Short for teletype, known as terminal.
- A device (implemented in software nowadays)
- Allows interaction with the system by passing on the data (input) to the system, and displaying the output produced by the system.

Types

- Python
- Echo
- /bin/sh –i
- Perl
- Ruby
- Interactive Ruby
- Lua
- Vi
- Nmap
- .SO Injection
- Symlinks

For more info

- https://blog.g0tmi1k.com/2011/08/basic-linux-privilege-escalation/

Linux Privilege Escalation - Available SUDO Commands

```
lubuntu@lubuntu:~$ sudo -l
[sudo] password for lubuntu:
Matching Defaults entries for lubuntu on lubuntu:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User lubuntu may run the following commands on lubuntu:
    (ALL : ALL) ALL
```