Learning CBC Bit-flipping Through Gamification

GIAC (GXPN) Gold Certification

Author: Jeremy Druin, jdruin@gmail.com
Advisor: Hamed Khiabani, Ph.D.

Accepted: April 10th, 2018

Abstract

Cryptanalysis concepts like CBC Bit-flipping can be difficult to grasp through study alone. Working through "hands-on" exercises is a common teaching technique intended to assist, but freely available training tools may not be readily available for advanced web application penetration testing practice. To this end, this paper will describe CBC bit-flipping and offer instruction on trying this cryptanalysis technique. Also, a CBC bit-flipping game will be provided within the OWASP Mutillidae II web application. Mutillidae is a large collection of deliberately vulnerable web application challenges designed to teach web security in a stand-alone, local environment.

1. Introduction

Cryptographic systems or cryptosystems may be categorized into three types: Asymmetric, symmetric and hash functions (BEHRENS, 2014). Unlike hash functions, the other two allow the data to be easily recovered as long as the recipient has the proper key (Mehmood, 2017). Asymmetric or "public key" systems utilize two encryption keys, one private and one public, to encrypt and decrypt respectively (Microsoft, 2017). In symmetric cryptosystems, the same key is used for encryption and decryption (***Figure 1***) (TYSON, 2018).

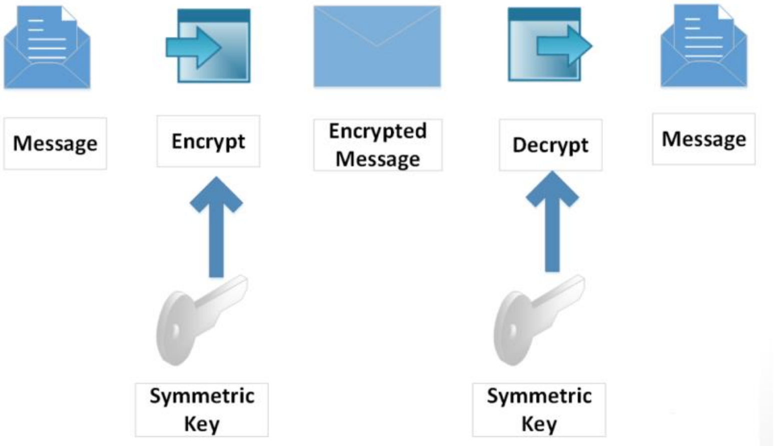


Figure 1: Symmetric key encryption

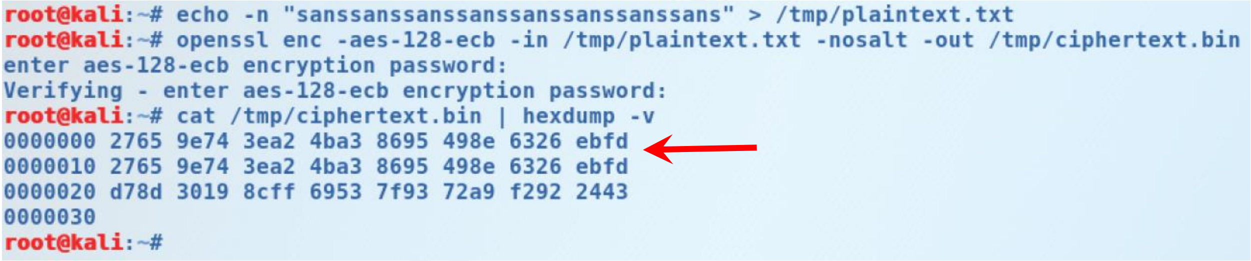
Symmetric encryption systems may use "block" or "stream" ciphers (Young, 2018) . Block ciphers accept, encipher, decipher and output a fixed number of bits. Stream ciphers encrypt or decrypt one bit at a time. Symmetric block ciphers offer various modes of operation that decide how the data is encrypted. The same block cipher can encrypt the data using different techniques. Some modes can even transform a block cipher into a stream cipher. These modes provide flexibility in how block ciphers transform the data (Dworkin, 2001).

1.1. What is cipher block chaining?

Cipher Block Chaining (CBC) is one mode available to block ciphers. Other modes include Electronic Codebook (ECB), Cipher Feedback (CFB), Output Feedback (OFB) and Counter Mode (CTR) (Kowalczyk, 2017).

1.1.1. Modes of Operation

Electronic Codebook mode might be thought of as a null mode. Besides encryption with the symmetric key, no additional transformation occurs (Thijssen, 2010). Each block of plaintext is encrypted independently. Also, the plaintext is not mixed with any additional data. This simplistic mode encrypts each block of plaintext with the key and outputs the ciphertext as-is. In ***Figure 2***, the phrase "sans" is repeated 8 times then encrypted in 128-bit[1] blocks using Advanced Encryption Standard (AES). The two lines of ciphertext output are identical because they were produced by encrypting the same input with the same key.



```
root@kali:~# echo -n "sanssanssanssanssanssanssanssans" > /tmp/plaintext.txt
root@kali:~# openssl enc -aes-128-ecb -in /tmp/plaintext.txt -nosalt -out /tmp/ciphertext.bin
enter aes-128-ecb encryption password:
Verifying - enter aes-128-ecb encryption password:
root@kali:~# cat /tmp/ciphertext.bin | hexdump -v
0000000 2765 9e74 3ea2 4ba3 8695 498e 6326 ebfd
0000010 2765 9e74 3ea2 4ba3 8695 498e 6326 ebfd
0000020 d78d 3019 8cff 6953 7f93 72a9 f292 2443
0000030
root@kali:~#
```

Figure 2: Encryption of data with recurring characters using Electronic Codebook (ECB) mode results in a repetitive ciphertext

Other modes incorporate data besides the key to further scramble the plaintext. Cipher Block Chaining (CBC) performs an exclusive-or (XOR) operation on the current plaintext block and the previously encrypted block (Tutorials Point, 2018). This mixing happens before encrypting the current block. The external source of data (from the current block point of view) acts like the salt used in hashing schemes (Morris & Thompson, 1978). The first block of plaintext presents a special case since no previously encrypted block exists. An Initialization Vector (IV) is provided to prime the pre-encryption operation (Barker & Barker, 2016).

Cipher Feedback (CFB) and Output Feedback (OFB) modes are similar to CBC except the characters in the prior block are encrypted rather than the plaintext. The result is then XORed with the current plaintext block to produce the current ciphertext block (Bunzel, 2018). These variants are desirable when it is convenient to have the underlying block cipher encrypt less than a full block of plaintext per encryption cycle (Hudde, 2009).

One mode combines the salting feature of CBC mode while operating on each block separately like ECB mode. Counter Mode (CTR) does not rely on output from previous

[1] The character encoding is ASCII-Extended which requires 8-bits (1 byte) per character. Encryption is occurring in blocks of 16 characters.

operations. Instead, an unpredictable number-used-once or "nonce" acts as the salt for the first block (Gerard, 2018). The nonce is incremented each encryption cycle before it is used. This "counter" provides uniqueness but allows each block to be encrypted and decrypted autonomously. (Rogaway, 2011).

1.1.2. Failings of Electronic Codebook

In Electronic Codebook (ECB) mode, the plaintext block is encrypted to produce the ciphertext block independent of any other block (IBM, 2018). If a plaintext block contains the same plaintext as another, an identical ciphertext block will be created (***Figure 2***). This redundancy provides opportunities for attacks.

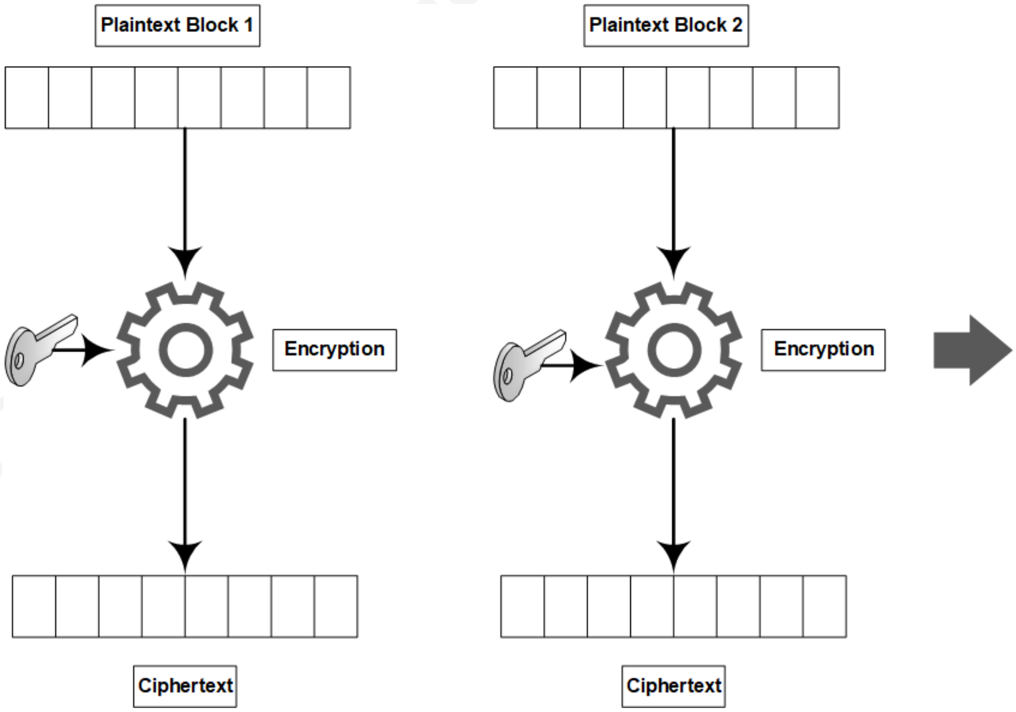


Figure 3: Symmetric Encryption - Electronic Codebook (ECB) Mode

For example, plaintext blocks can be substituted or replayed without knowing the contents (Sharma, 2007). Also, any previously cracked block betrays all identical ciphertext blocks (Crowley, 2013). In some cases, such as encrypting images, the image pattern can leak into the ciphertext (abpolym, 2015).

Identical plaintext is common in network communications. Many messages have well-known headers opening ECB-mode encrypted ciphertext to chosen plaintext attack (Chaudhary, 2014). Networking protocols such as Address Resolution Protocol and Hypertext Transport Protocol contain predictable headers found at the start of each network packet (Plummer, 1982) (Fielding & Reschke, 2014). These packets and their associated headers are transmitted many times until the entire message is sent.

1.1.3. Improvement over ECB

Cipher block chaining (CBC) offers advantages of Electronic Codebook mode. Each block of plaintext is mixed with the previous block of ciphertext (***Figure 4***). The pre-encryption XOR operation alleviates identical ciphertext blocks; even when the same plaintext is encrypted with the same key[2].

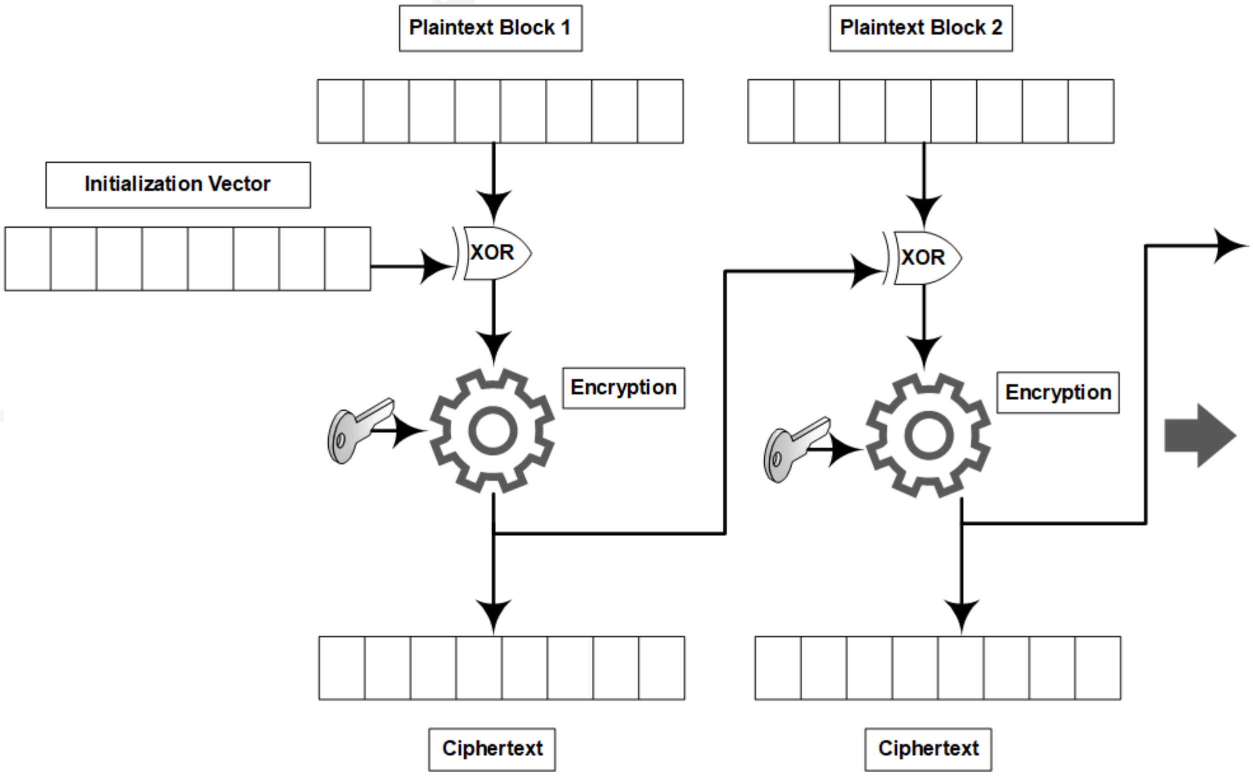


Figure 4: Symmetric Encryption - Cipher Block Chaining (CBC) Mode

[2] This assumes best-practice is followed. An initialization vector must never be reused with a given key (Gordon, 2015).

Since the first ciphertext block has no predecessor, an initial random value stands in for the first ciphertext block. This "*initialization vector*" (IV) provides the randomness needed to ensure identical plaintexts do not encrypt to repetitive ciphertexts. While it is not necessary to keep the IV secret since its only function is to introduce distinctiveness into the ciphertext, it is critical the IV be as unpredictable as possible (Diana-Lynn Contesti, 2007). The IV is provided to the recipient along with the ciphertext so the first block can be decrypted (D.I. Management Services Pty Ltd, 2001).

1.2. What is CBC bit-flipping?

CBC improves the integrity of the ciphertext by doing away with the repeated patterns in the plaintext. Also, exchanging, replicating or deleting ciphertext will corrupt at least one block of the message (Focardi, 2014). However, the exclusive-or (XOR)[3] operation is linear. Changing one bit of ciphertext in a previous block has a predictable consequence on the respective bit in the current block of plaintext (Regalado, 2013).

1.2.1. Exclusive-Or

As a logical operation, XOR follows the rules expressed in ***Table 1***. If the two inputs are different, the result is *True* but otherwise *False* (Electronics Tutorials, 2018). Assuming False is encoded as 0 and True as 1, the operation outputs bits as prescribed in ***Table 2***.

Table 1: Truth Table for Exclusive-Or

A	**B**	**A ⊕ B**
False	False	False
False	True	True
True	False	True
True	True	False

[3] The exclusive-or (XOR) operation is represented by the symbol ⊕ (Addition symbol "plus" embedded in a circle)

Table 2: Result of Exclusive-Or operation on bits

A	B	A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

Exclusive-Or can be implemented as addition modulo 2 (Germundsson, 2002). For the first 3 rows of ***Table 2***, addition works regardless of modulo (i.e. 0 + 0 = 0, 0 + 1 = 1…). In row 4, (1 + 1) mod 2 = 2 mod 2 = 0. Alternatively, one of the two input bits can be thought of as a switch that decides whether the second bit should change. For 0 ⊕ 1, the first bit has a value of zero which indicates we should leave the second bit alone. "1" is output. For 1 ⊕ 1, the first bit has a value of one telling us to change the second bit. The answer is "0".

1.2.2. Bit-flipping

When considering CBC, it may be more convenient to think in terms of the switch analogy. In the first two rows of ***Table 2***, the first bit "A" is 0 so the second bit "B" is output as-is. In the third and fourth rows of ***Table 2***, the first bit "A" is 1 so the opposite of bit "B" is output. Regardless, note that changing bit "A" no matter the value of bit "B" always causes the output to alternate or *flip*[4] with respect to "B". Viewing the first bit as a switch dovetails nicely with the attack. The attacker decides which bit(s) in the previous block to change. These "flips" consistently cause the corresponding bits in the next block to change their value.

In CBC, the mixing of the plaintext and previous ciphertext block (or initialization vector) occurs prior to encryption (***Figure 4***). Exclusive-Or respects bit order when operating on blocks. The first bit of prior ciphertext is XORed with the first bit of plaintext, etc. If bit 3 of the previous ciphertext (or IV) is switched, bit 3 of the plaintext will be flipped prior to encryption (***Figure 5***).

[4] The term "flip" may refer to turning over a coin so head becomes tails or vice versa

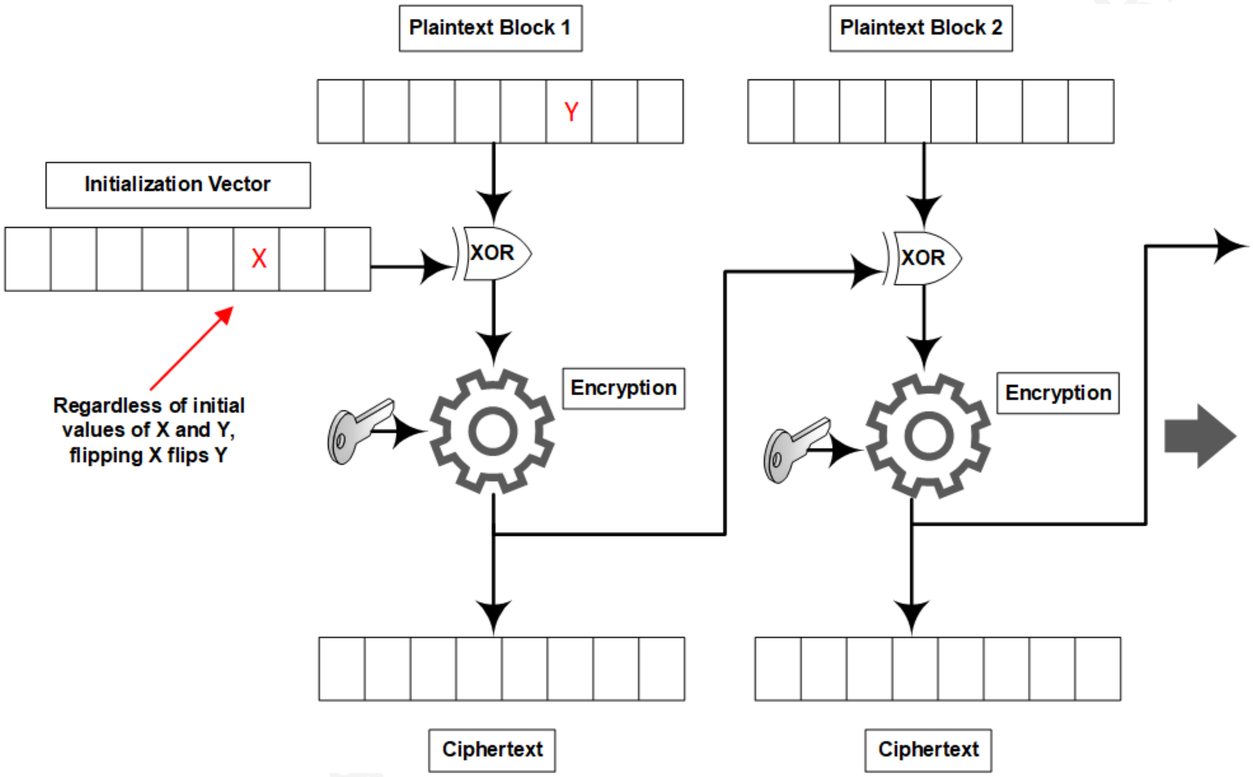


Figure 5: CBC Bit-flipping

The consequence of bit-flipping on an actual application is highly contextual. It depends on how the application decodes and interprets the affected bits, but the consequences can be significant (GOODIN, 2016) (Marclass, 2015).

1.3. How might we learn about CBC bit-flipping?

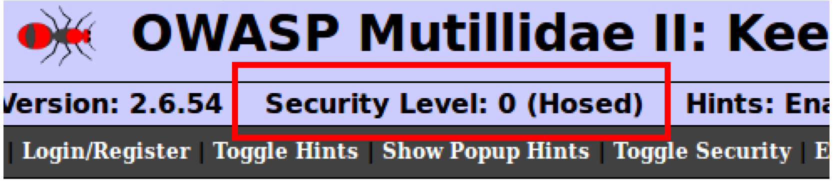
Complex information security concepts may be easier to learn through experience (Griffiths & Guile, 2004). "Gamification" is an interactive simulation in the form of a goal-oriented challenge. Gamification is shown to increase awareness and interest in difficult notions (Gjertsen, Gjære, Bartnes, & Flores, 2017). OWASP Mutillidae II is a deliberately vulnerable web-application that uses gamification to teach web application security attacks and defenses (Druin, OWASP Mutillidae II, 2018). A CBC bit-flipping challenge is included (Druin, Listing of Vulnerabilities, 2018) .

2. The Game

2.1. OWASP Mutillidae II Web Application

OWASP Mutillidae II is a free, open source web application that implements over 40 web application vulnerabilities including multiple defects from each of the OWASP Top Ten list (Druin, 2018). It is a training system that guides the user with hints that pop-up when the user hovers over a vulnerable input parameter and context-sensitive tutorials built into each page. Relevant instructional videos are linked from the bottom of each tutorial.

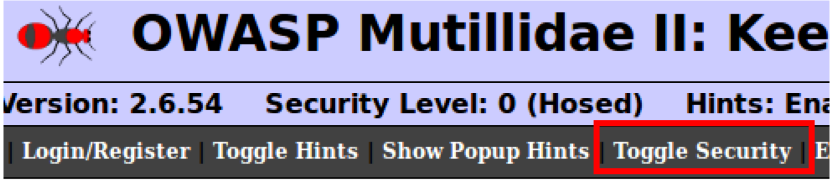
There are two learning modes: vulnerable ("Level 0"/"Level 1") and secure ("Level 5"). "Level 0" and "Level 1" implement identical vulnerable PHP source code which can be hacked[5]. "Level 5" implements a different set of PHP source code that has been patched against the respective vulnerability. The system is in "Level 0" by default (***Figure 6***).


OWASP Mutillidae II: Kee
| Login/Register | Toggle Hints | Show Popup Hints | Toggle Security | E

Version: 2.6.54 Security Level: 0 (Hosed) Hints: Ena

Figure 6: Security Level

The following demos use "Level 0" exclusively. If needed, the level can be changed by clicking "***Toggle Security***" (***Figure 7***).


OWASP Mutillidae II: Kee
| Login/Register | Toggle Hints | Show Popup Hints | Toggle Security | E

Version: 2.6.54 Security Level: 0 (Hosed) Hints: Ena

Figure 7: Toggle Security button

2.2. Installation

A version of Mutillidae is pre-installed on the Samurai Web Testing Framework virtual machine[6]. Mutillidae can be updated to the latest version by copying the latest scripts over the

[5] "Level 1" layers on several client-side "security controls"
[6] https://sourceforge.net/projects/samurai/files/SamuraiWTF%203.0%20Branch/

existing[7]. Also, the source is available on SourceForge[8]. The project may be installed on Linux Ubuntu using LAMP[9] or Windows on XAMPP[10].

2.3. Solving the CBC bit-flipping Challenge

2.3.1. Understanding the goal

The challenge is implemented within the "**View User Privilege Level**" page. On the left menu, click "***OWASP 2017***" → "***A2 - Broken Authentication and Session Management***" → "***Privilege Escalation***" → "***Via CBC bit-flipping***" (***Figure 8***). The page will load (***Figure 9***).

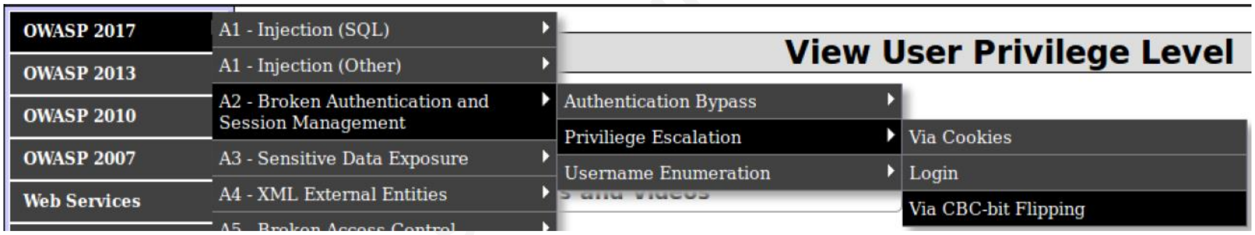
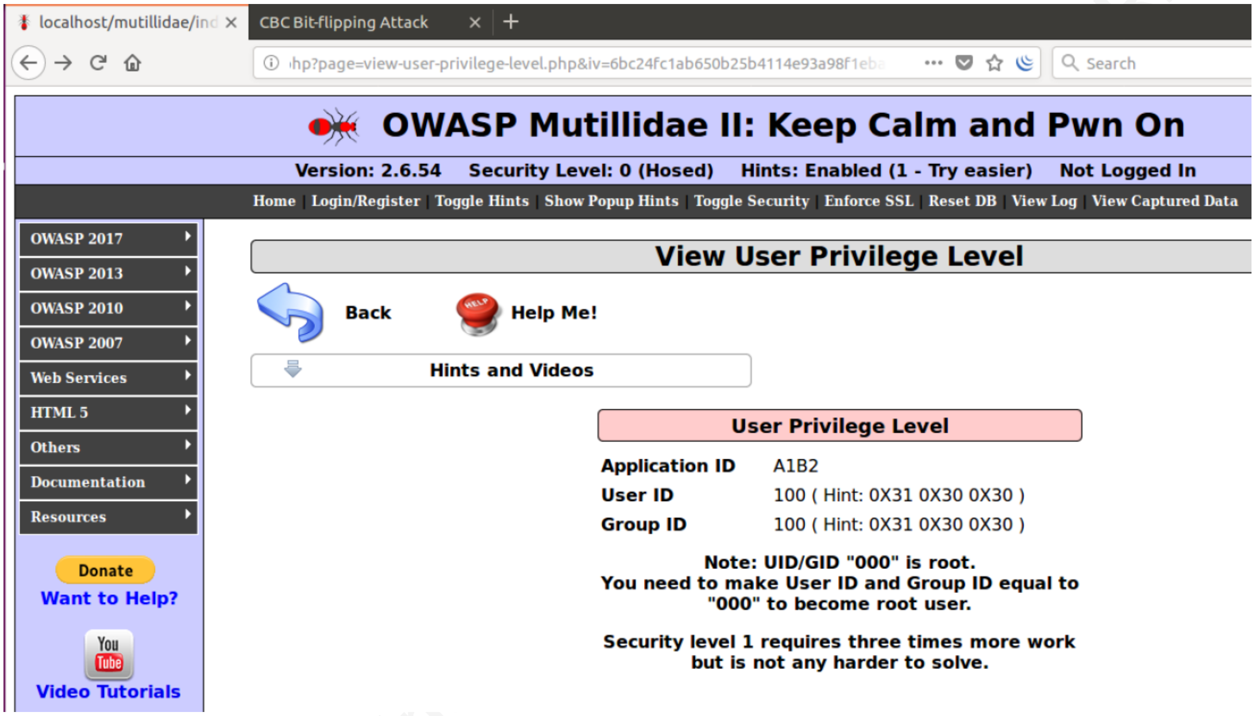


Figure 8: Menu to View User Privilege Level page

[7] https://www.youtube.com/watch?v=uQ2lr0TIiqE
[8] https://sourceforge.net/projects/mutillidae/files/mutillidae-project/
[9] https://sourceforge.net/projects/mutillidae/files/documentation/mutillidae-installation-on-ubuntu.txt/download
[10] https://sourceforge.net/projects/mutillidae/files/documentation/mutillidae-installation-on-xampp-win7.pdf/download


localhost/mutillidae/ind
CBC Bit-flipping Attack
hp?page=view-user-privilege-level.php&iv=6bc24fc1ab650b25b4114e93a98f1eba
Search
Home | Login/Register | Toggle Hints | Show Popup Hints | Toggle Security | Enforce SSL | Reset DB | View Log | View Captured Data
OWASP 2017
OWASP 2013
OWASP 2010
OWASP 2007
Web Services
HTML 5
Others
Documentation
Resources
Donate
Want to Help?
Video Tutorials
Back
Help Me!
Hints and Videos

OWASP Mutillidae II: Keep Calm and Pwn On

Version: 2.6.54 Security Level: 0 (Hosed) Hints: Enabled (1 - Try easier) Not Logged In

View User Privilege Level

User Privilege Level

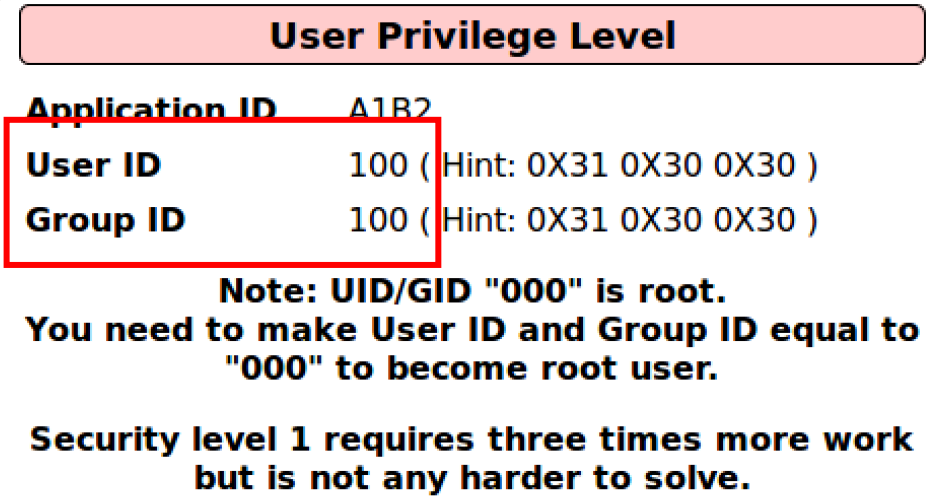
Application ID	A1B2
User ID	100 (Hint: 0X31 0X30 0X30)
Group ID	100 (Hint: 0X31 0X30 0X30)

Note: UID/GID "000" is root.
You need to make User ID and Group ID equal to "000" to become root user.

Security level 1 requires three times more work but is not any harder to solve.

Figure 9: View User Privilege Level page

By default, the "***User ID***" and "***Group ID***" displayed will be "***100***" (***Figure 9***)[11]. The challenge is to set both the "***User ID***" and "***Group ID***" to "***000***" using a CBC bit-flipping attack. Both fields are close to the target value. Only the leading "1" needs to be changed to a "0".


User Privilege Level

Application ID	A1B2
User ID	100 (Hint: 0X31 0X30 0X30)
Group ID	100 (Hint: 0X31 0X30 0X30)

Note: UID/GID "000" is root.
You need to make User ID and Group ID equal to "000" to become root user.

Security level 1 requires three times more work but is not any harder to solve.

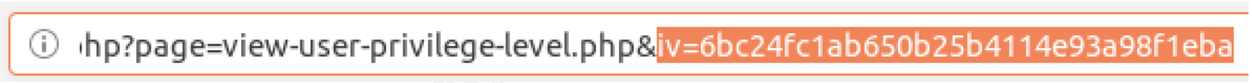
Figure 10: User and Group IDs

[11] If the player would like to attempt a more difficult scenario, security "Level 1" implements "User ID" of "174" and "Group ID" of "235"

2.3.2. How to locate the vulnerable bytes

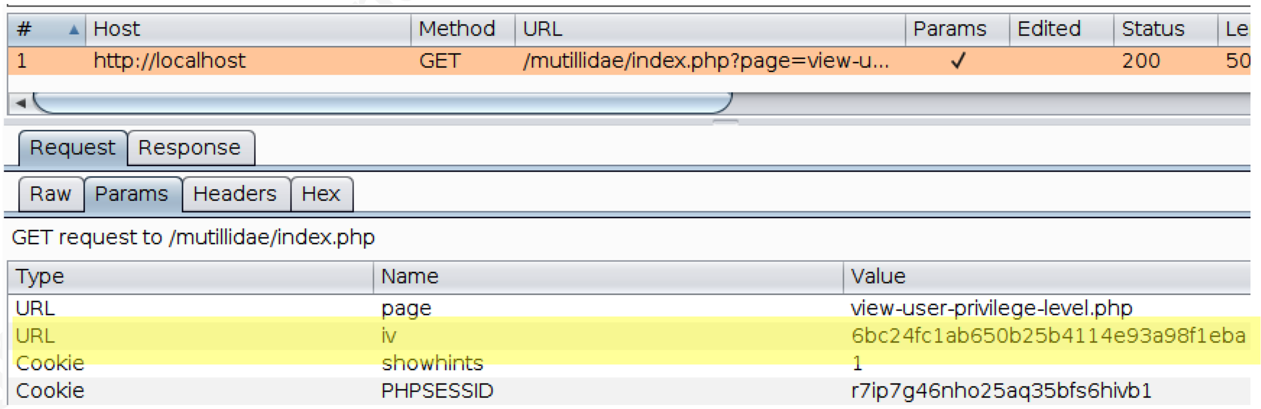
To attempt a CBC bit-flip attack on a given block, except for the first block, any previous block of ciphertext can be targeted (***Figure 4***). To manipulate the first block, the initialization vector has to be targeted. Each input on the page needs to be inspected to determine if the parameter provides an opportunity to influence some block.

When first approaching the page, it is not clear if an attack vector is exposed to the client although one parameter is suspicious. Inspection of the URL shows a query parameter named "iv" with a value "6bc24fc1ab650b25b4114e93a98f1eba" (***Figure 11***). The same parameter can be seen in Burp-Suite[12] [13] (***Figure 12***).



hp?page=view-user-privilege-level.php&iv=6bc24fc1ab650b25b4114e93a98f1eba

Figure 11: Parameter "iv" in the URL



#	Host	Method	URL	Params	Edited	Status	Le
1	http://localhost	GET	/mutillidae/index.php?page=view-u...	✓		200	50

GET request to /mutillidae/index.php

Type	Name	Value
URL	page	view-user-privilege-level.php
URL	iv	6bc24fc1ab650b25b4114e93a98f1eba
Cookie	showhints	1
Cookie	PHPSESSID	r7ip7g46nho25aq35bfs6hivb1

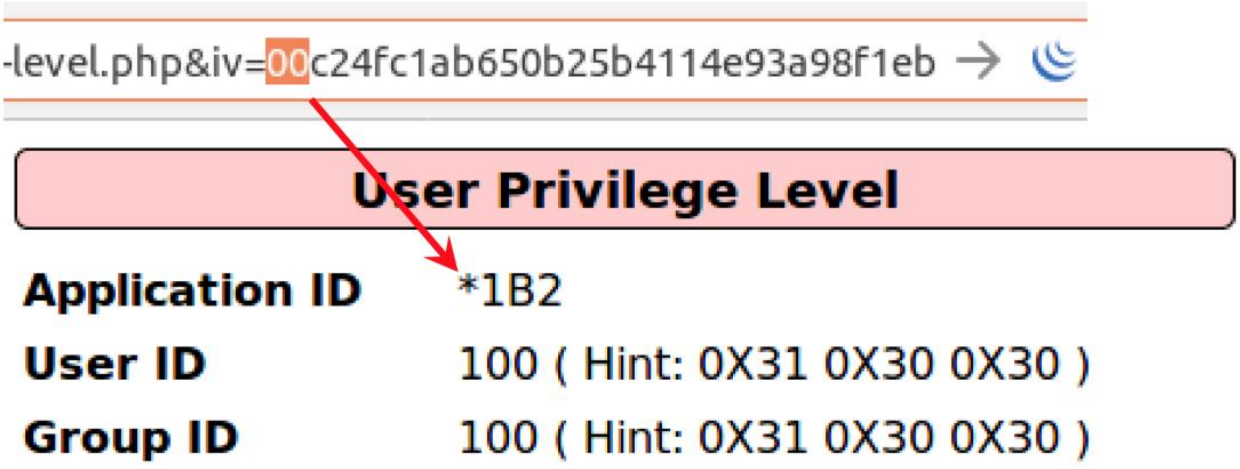
Figure 12: Parameter "iv" shown in Burp-Suite[14]

There does not appear to be any other input parameter that could have an effect on either the IV or a block of ciphertext. Also, the parameter name is dubious. "Pareto tests"[15], trying only the input most likely to confirm the correct variable, are warranted to verify "iv". If these quick

[12] Burp-Suite Community Edition 1.7.30 on Ubuntu 16.04.3 LTS was used for demonstration
[13] Please refer to ***Appendix A*** for video tutorials on Burp-Suite
[14] Proxy → HTTP History, click the request, select Request tab → Params tab
[15] Vilfredo Pareto noted in the context of economics that about 20% of the input leads to 80% of results (Investopedia, 2018). The principle also applies to results relative to effort.

tests are inconclusive, methodical testing can be done. This test can be done formally or informally using manual methods or automated tools.

To test informally using a manual method, the first character or two of the "iv" field can be changed. Because the test is informal, the first two characters can be set to an arbitrary value and resubmitted. "00" is used in this example. Any change in output may indicate the "iv" parameter is a viable target. Indeed, the value of Application ID changes from "A1B2" to "*1B2" (***Figure 13***). Since there are only 32 characters in the "iv" parameter, manual analysis works well even if all positions need to be tested.



```
-level.php&iv=00c24fc1ab650b25b4114e93a98f1eb
```

User Privilege Level

Application ID *1B2

User ID 100 (Hint: 0X31 0X30 0X30)

Group ID 100 (Hint: 0X31 0X30 0X30)

Figure 13: Application ID changes when "iv" parameter manipulated

The remaining bytes are mapped by changing pairs of characters in the IV to "00", observing the page output, and recording the results.

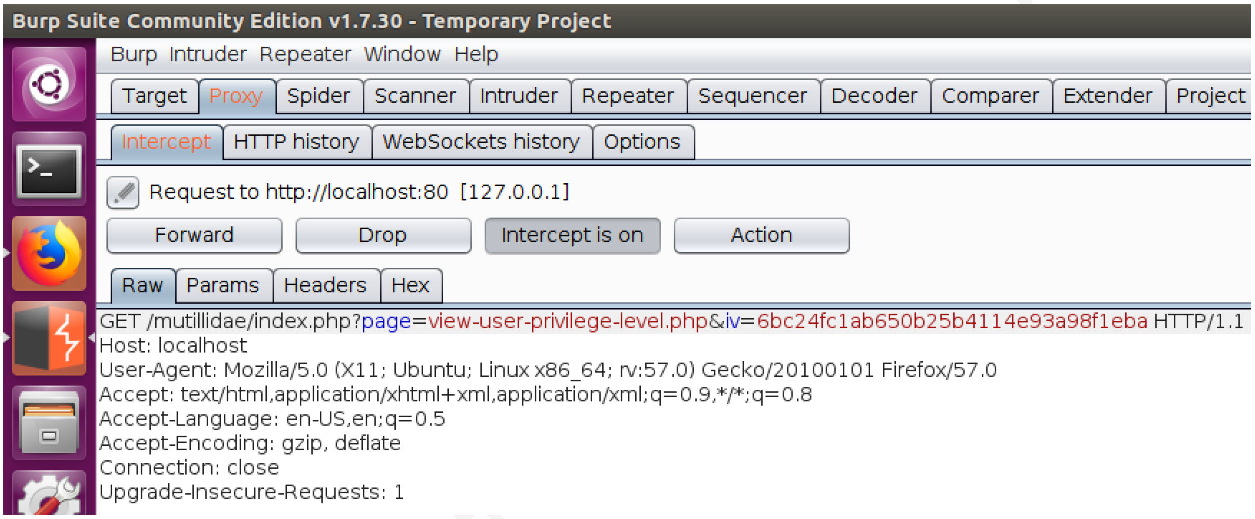
Formal testing can be done with *Burp-Suite Community Edition*[16] [17]. The request is intercepted in *Burp-Suite Proxy* tool (***Figure 14***) then sent to the *Burp-Suite Repeater* tool where it is replayed (***Figure 15***). To ensure that replaying the HTTP request did not result in an unexpected response[18], the response from the intercepted request is compared against the response from the repeated request using the *Burp-Suite Comparer* tool. No significant

[16] https://portswigger.net/burp/communitydownload

[17] Please see ***Appendix A*** for links to video tutorials prepared for each of the Burp-Suite features referenced

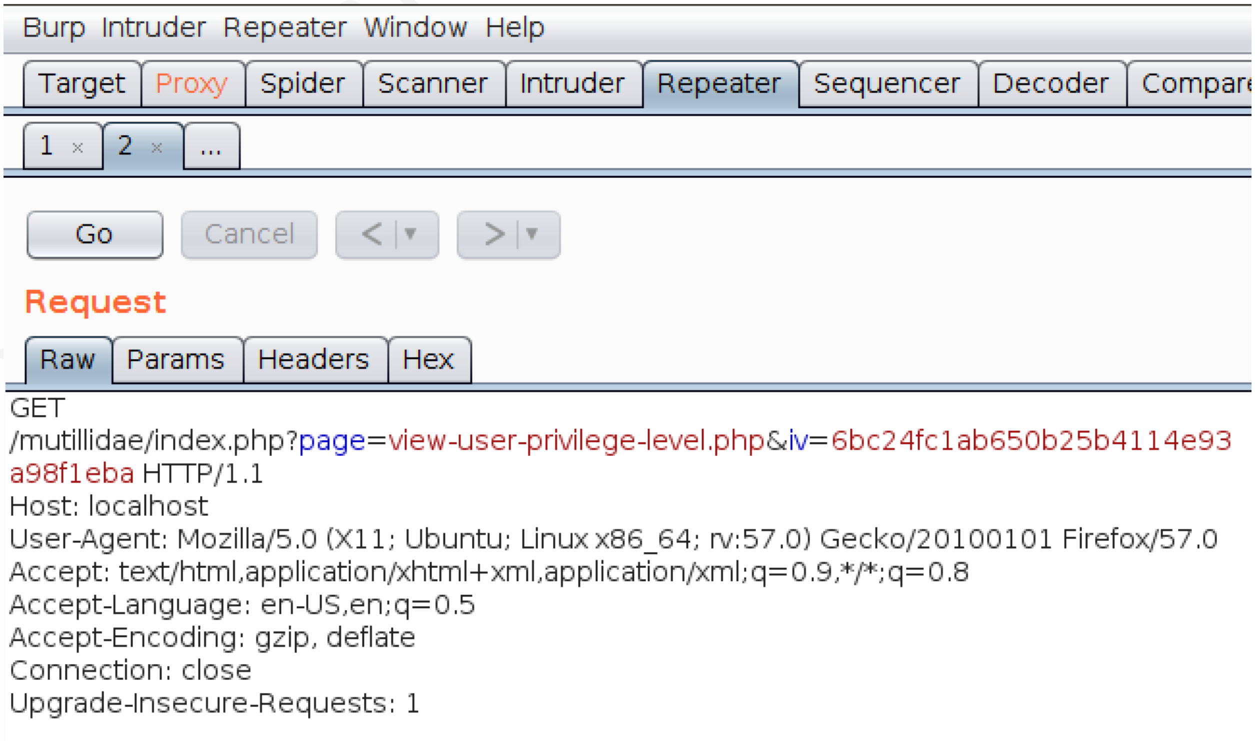
[18] For example, if the site employed cross-site request forgery (CSRF) tokens valid for only one request

difference is found (***Figure 16***). This indicates requests can be repeated with minimal risk of invalid responses.


Burp Suite Community Edition v1.7.30 - Temporary Project
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project
Intercept HTTP history WebSockets history Options
Request to http://localhost:80 [127.0.0.1]
Forward Drop Intercept is on Action
Raw Params Headers Hex

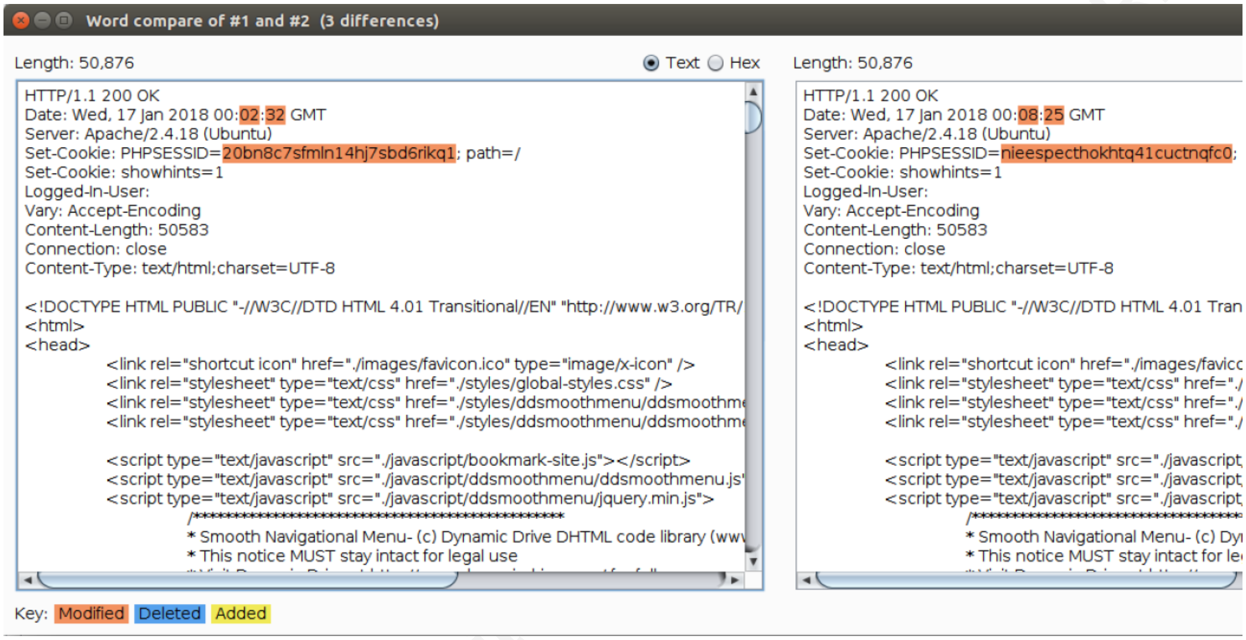
```
GET /mutillidae/index.php?page=view-user-privilege-level.php&iv=6bc24fc1ab650b25b4114e93a98f1eba HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:57.0) Gecko/20100101 Firefox/57.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
```

Figure 14: HTTP request for the page is intercepted in Burp-Suite Proxy tool


Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Compare
1 × 2 × ...
Go Cancel < >
Request
Raw Params Headers Hex

```
GET
/mutillidae/index.php?page=view-user-privilege-level.php&iv=6bc24fc1ab650b25b4114e93
a98f1eba HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:57.0) Gecko/20100101 Firefox/57.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
```

Figure 15: HTTP request for the page is sent to Burp-Suite Repeater tool


Word compare of #1 and #2 (3 differences)
● Text ○ Hex
Key: Modified Deleted Added

Length: 50,876

```
HTTP/1.1 200 OK
Date: Wed, 17 Jan 2018 00:02:32 GMT
Server: Apache/2.4.18 (Ubuntu)
Set-Cookie: PHPSESSID=20bn8c7sfmln14hj7sbd6rikq1; path=/
Set-Cookie: showhints=1
Logged-In-User:
Vary: Accept-Encoding
Content-Length: 50583
Connection: close
Content-Type: text/html;charset=UTF-8

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/
<html>
<head>
        <link rel="shortcut icon" href="./images/favicon.ico" type="image/x-icon" />
        <link rel="stylesheet" type="text/css" href="./styles/global-styles.css" />
        <link rel="stylesheet" type="text/css" href="./styles/ddsmoothmenu/ddsmoothme
        <link rel="stylesheet" type="text/css" href="./styles/ddsmoothmenu/ddsmoothme

        <script type="text/javascript" src="./javascript/bookmark-site.js"></script>
        <script type="text/javascript" src="./javascript/ddsmoothmenu/ddsmoothmenu.js'
        <script type="text/javascript" src="./javascript/ddsmoothmenu/jquery.min.js">
                /*************************************************
                * Smooth Navigational Menu- (c) Dynamic Drive DHTML code library (ww
                * This notice MUST stay intact for legal use
```

Length: 50,876

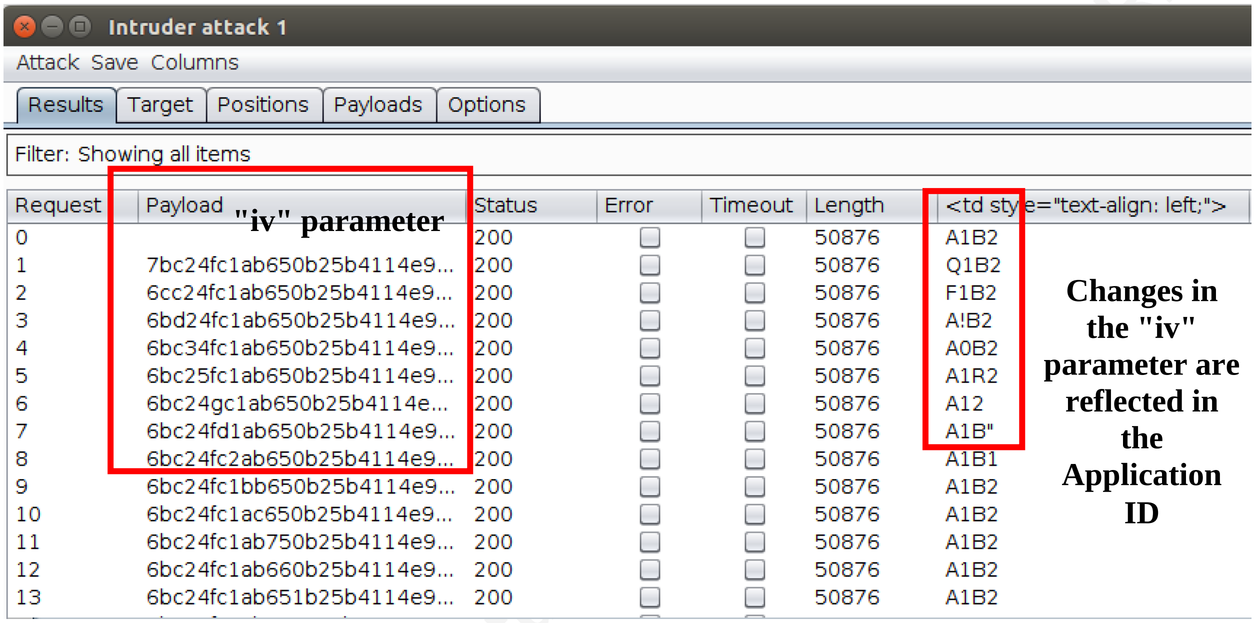
```
HTTP/1.1 200 OK
Date: Wed, 17 Jan 2018 00:08:25 GMT
Server: Apache/2.4.18 (Ubuntu)
Set-Cookie: PHPSESSID=nieespecthokhtq41cuctnqfc0;
Set-Cookie: showhints=1
Logged-In-User:
Vary: Accept-Encoding
Content-Length: 50583
Connection: close
Content-Type: text/html;charset=UTF-8

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Tran
<html>
<head>
        <link rel="shortcut icon" href="./images/favicc
        <link rel="stylesheet" type="text/css" href="./
        <link rel="stylesheet" type="text/css" href="./
        <link rel="stylesheet" type="text/css" href="./

        <script type="text/javascript" src="./javascript
        <script type="text/javascript" src="./javascript
        <script type="text/javascript" src="./javascript
                /************************************
                * Smooth Navigational Menu- (c) Dy
                * This notice MUST stay intact for le
```

Figure 16: HTTP responses are compared to ensure requests are repeatable

The *Character Frobber* attack found in the *Burp-Suite Intruder* tool is designed to swiftly check if characters in a lengthy string impact how an application handles input (Portswigger, 2018). During the test, Burp-Suite sends one HTTP request for each character in the value of the "iv" parameter. The respective character is "incremented" before sending each request so that 6 becomes a 7, "b" becomes "c" and so on (PortSwigger, 2018). There are 32 characters in the value of "iv" so 32 unique requests are sent. The "Grep - Extract" feature can display the Application ID for each request to make changes in the Application ID easier to spot (***Figure 17***).



Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	<td style="text-align: left;">
0		200	☐	☐	50876	A1B2
1	7bc24fc1ab650b25b4114e9...	200	☐	☐	50876	Q1B2
2	6cc24fc1ab650b25b4114e9...	200	☐	☐	50876	F1B2
3	6bd24fc1ab650b25b4114e9...	200	☐	☐	50876	A!B2
4	6bc34fc1ab650b25b4114e9...	200	☐	☐	50876	A0B2
5	6bc25fc1ab650b25b4114e9...	200	☐	☐	50876	A1R2
6	6bc24gc1ab650b25b4114e...	200	☐	☐	50876	A12
7	6bc24fd1ab650b25b4114e9...	200	☐	☐	50876	A1B"
8	6bc24fc2ab650b25b4114e9...	200	☐	☐	50876	A1B1
9	6bc24fc1bb650b25b4114e9...	200	☐	☐	50876	A1B2
10	6bc24fc1ac650b25b4114e9...	200	☐	☐	50876	A1B2
11	6bc24fc1ab750b25b4114e9...	200	☐	☐	50876	A1B2
12	6bc24fc1ab660b25b4114e9...	200	☐	☐	50876	A1B2
13	6bc24fc1ab651b25b4114e9...	200	☐	☐	50876	A1B2

Figure 17: The Grep -Extract feature show the impact of changing the "iv" input parameter on the "Application ID" field

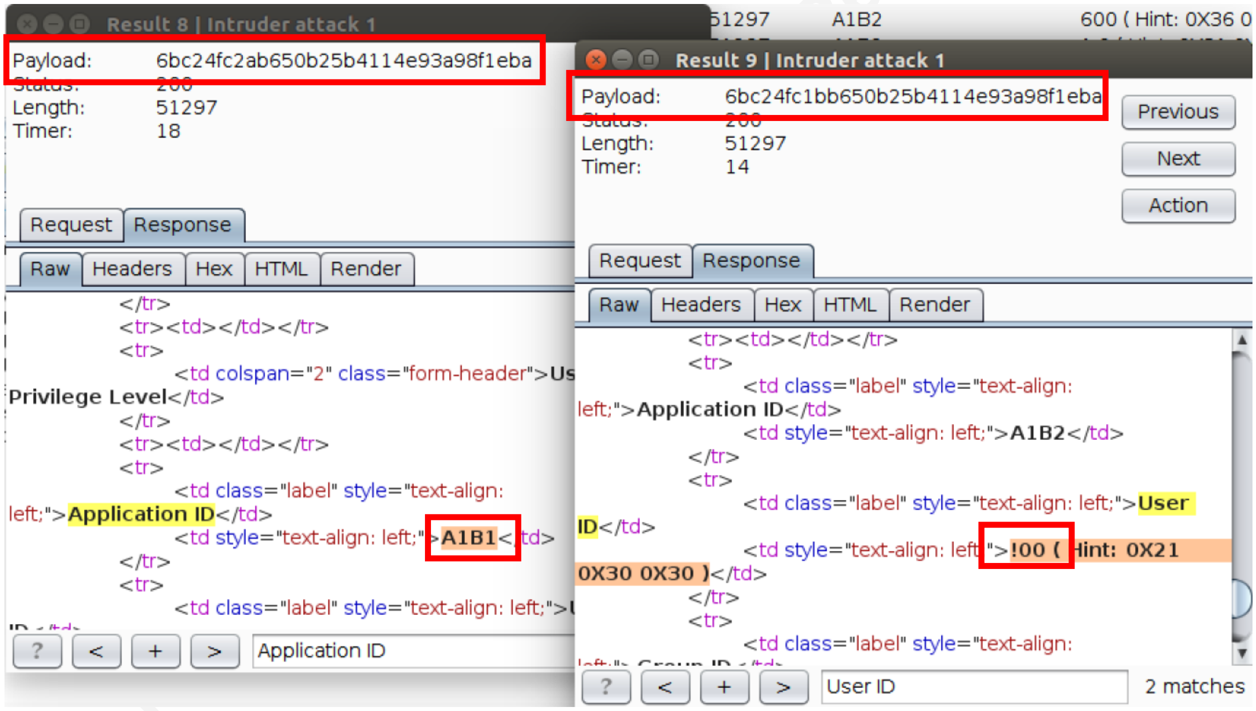
In the first request, the value of the Application ID is "A1B2"; the default value. In the second request, Burp changed the first character of the IV from 6 to 7. The Application ID changed to "Q1B2". This implies the "iv" parameter is the initialization vector sought. Requests 2 - 7 provide additional evidence.

The test also suggests the IV is encoded as hexadecimal digits. The effects of character frobbing wear off once Burp moves on to characters 9 - 32 (***Figure 17***). The Application ID is only influenced if the first 8 characters are altered and the IV is made of 0-9/A-F. Assuming each pair of letters in the IV is a byte, the first letter of the Application ID corresponds to the first byte of the IV, the second letter to the second byte, etc. The information gathered so far is summarized in ***Table 3***.

Table 3: Bytes of Initialization Vector by Position and Effect

Position	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Value	6b	c2	4f	c1	ab	65	0b	25	b4	11	4e	93	a9	8f	1e	ba
Effect	A	1	B	2	?	?	?	?	?	?	?	?	?	?	?	?
Affected Field Name	Application ID				?	?	?	?	?	?	?	?	?	?	?	?

Manipulating characters 9 - 32 did not impact the Application ID but may have affected other output. The boundary of character 8 and 9 can be tested with the Comparer tool[19] or by looking at each HTTP response in a new window[20]. The comparison shows IV-character 8 affects the end of the Application ID and the IV-character 9 does the same for the first number in the User ID (***Figure 18***).



Result 8 | Intruder attack 1

```
Payload:    6bc24fc2ab650b25b4114e93a98f1eba
Status:     200
Length:     51297
Timer:      18
```

```
            </tr>
            <tr><td></td></tr>
            <tr>
                <td colspan="2" class="form-header">Us
Privilege Level</td>
            </tr>
            <tr><td></td></tr>
            <tr>
                <td class="label" style="text-align:
left;">Application ID</td>
                <td style="text-align: left;">A1B1</td>
            </tr>
            <tr>
                <td class="label" style="text-align: left;">U
ID</td>
```

Search: Application ID

Result 9 | Intruder attack 1

```
Payload:    6bc24fc1bb650b25b4114e93a98f1eba
Status:     200
Length:     51297
Timer:      14
```

```
            <tr><td></td></tr>
            <tr>
                <td class="label" style="text-align:
left;">Application ID</td>
                <td style="text-align: left;">A1B2</td>
            </tr>
            <tr>
                <td class="label" style="text-align: left;">User
ID</td>
                <td style="text-align: left;">!00 ( Hint: 0X21
0X30 0X30 )</td>
            </tr>
            <tr>
                <td class="label" style="text-align:
left;">Group ID</td>
```

Search: User ID — 2 matches

```
51297    A1B2    600 ( Hint: 0X36 0
```

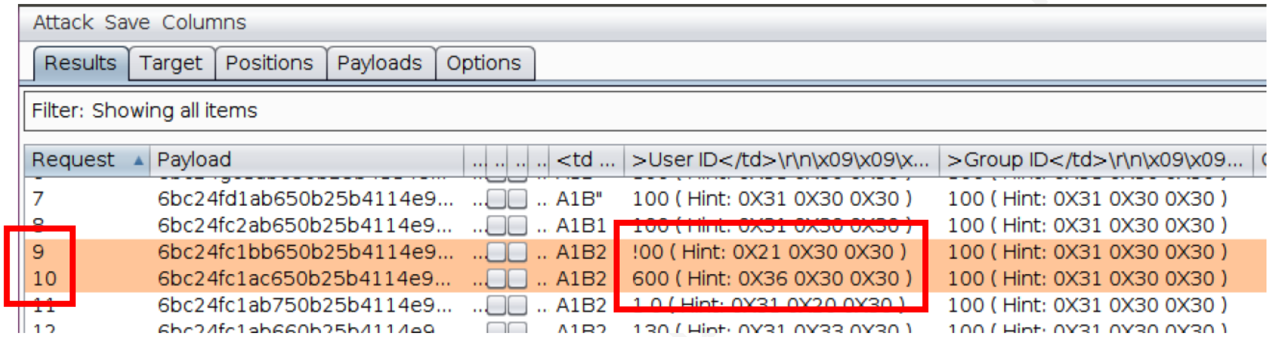
Figure 18: Comparison of the effect of manipulating character 8 and 9 in the IV

Opening each response for manual comparison is tedious. The User ID and Group ID fields can be added to Grep - Extract for easy comparison. The output shows IV-characters 9 and 10 (IV byte 5) map to the first digit in the User ID (***Figure 19***). Similarly, IV-characters 15 and 16 (IV byte 8) map to the first digit in the Group ID (***Figure 20***). Recall these are the two digits need to be changed to "0" to win the game (***Figure 10***). ***Table 4*** updates how each byte of the IV

[19] Highlight rows 8 and 9 in the Intruder Attack window (***Figure 17***), right-click and select "Send to Comparer (Responses)".

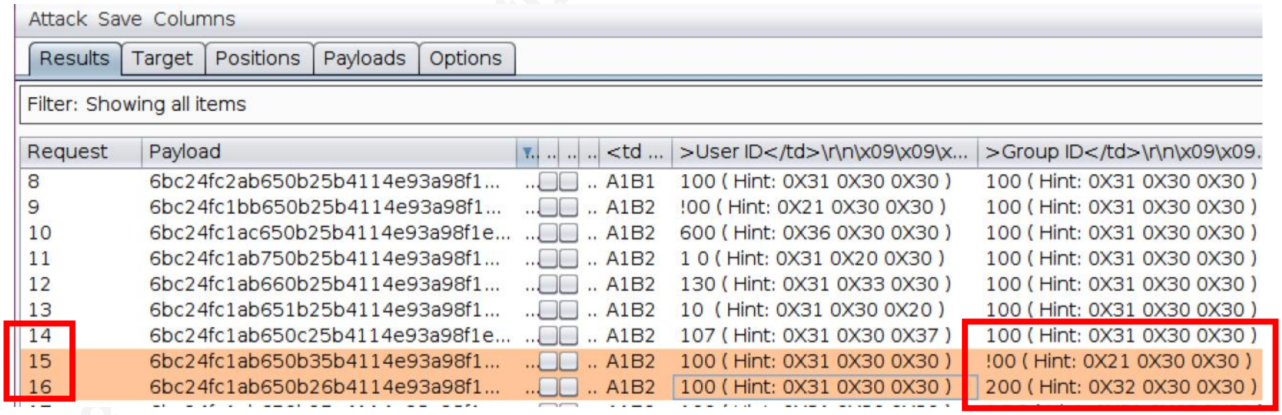
[20] Double-click on row 8 and 9 respectively in the Intruder Attack window (***Figure 17***), click the Response tab, search for "Application ID" (***Figure 18***)

maps to the digits of the Application ID, User ID, and Group ID. The targeted IV bytes 5 and 8 are highlighted.



Request	Payload	..	..	..	<td ...	>User ID</td>\r\n\x09\x09\x...	>Group ID</td>\r\n\x09\x09...
7	6bc24fd1ab650b25b4114e9...	..	□	□ ..	A1B"	100 (Hint: 0X31 0X30 0X30)	100 (Hint: 0X31 0X30 0X30)
8	6bc24fc2ab650b25b4114e9...	..	□	□ ..	A1B1	100 (Hint: 0X31 0X30 0X30)	100 (Hint: 0X31 0X30 0X30)
9	6bc24fc1bb650b25b4114e9...	..	□	□ ..	A1B2	!00 (Hint: 0X21 0X30 0X30)	100 (Hint: 0X31 0X30 0X30)
10	6bc24fc1ac650b25b4114e9...	..	□	□ ..	A1B2	600 (Hint: 0X36 0X30 0X30)	100 (Hint: 0X31 0X30 0X30)
11	6bc24fc1ab750b25b4114e9...	..	□	□ ..	A1B2	1 0 (Hint: 0X31 0X20 0X30)	100 (Hint: 0X31 0X30 0X30)
12	6bc24fc1ab660b25b4114e9		□	□	A1B2	130 (Hint: 0X31 0X33 0X30)	100 (Hint: 0X31 0X30 0X30)

Figure 19: IV-characters 9 and 10 influence the first digit of the User ID



Request	Payload	..	..	..	<td ...	>User ID</td>\r\n\x09\x09\x...	>Group ID</td>\r\n\x09\x09.
8	6bc24fc2ab650b25b4114e93a98f1...	..	□	□ ..	A1B1	100 (Hint: 0X31 0X30 0X30)	100 (Hint: 0X31 0X30 0X30)
9	6bc24fc1bb650b25b4114e93a98f1...	..	□	□ ..	A1B2	!00 (Hint: 0X21 0X30 0X30)	100 (Hint: 0X31 0X30 0X30)
10	6bc24fc1ac650b25b4114e93a98f1e...	..	□	□ ..	A1B2	600 (Hint: 0X36 0X30 0X30)	100 (Hint: 0X31 0X30 0X30)
11	6bc24fc1ab750b25b4114e93a98f1...	..	□	□ ..	A1B2	1 0 (Hint: 0X31 0X20 0X30)	100 (Hint: 0X31 0X30 0X30)
12	6bc24fc1ab660b25b4114e93a98f1...	..	□	□ ..	A1B2	130 (Hint: 0X31 0X33 0X30)	100 (Hint: 0X31 0X30 0X30)
13	6bc24fc1ab651b25b4114e93a98f1...	..	□	□ ..	A1B2	10 (Hint: 0X31 0X30 0X20)	100 (Hint: 0X31 0X30 0X30)
14	6bc24fc1ab650c25b4114e93a98f1e...	..	□	□ ..	A1B2	107 (Hint: 0X31 0X30 0X37)	100 (Hint: 0X31 0X30 0X30)
15	6bc24fc1ab650b35b4114e93a98f1...	..	□	□ ..	A1B2	100 (Hint: 0X31 0X30 0X30)	!00 (Hint: 0X21 0X30 0X30)
16	6bc24fc1ab650b26b4114e93a98f1...	..	□	□ ..	A1B2	100 (Hint: 0X31 0X30 0X30)	200 (Hint: 0X32 0X30 0X30)

Figure 20: IV-characters 15 and 16 influence the first digit of the Group ID

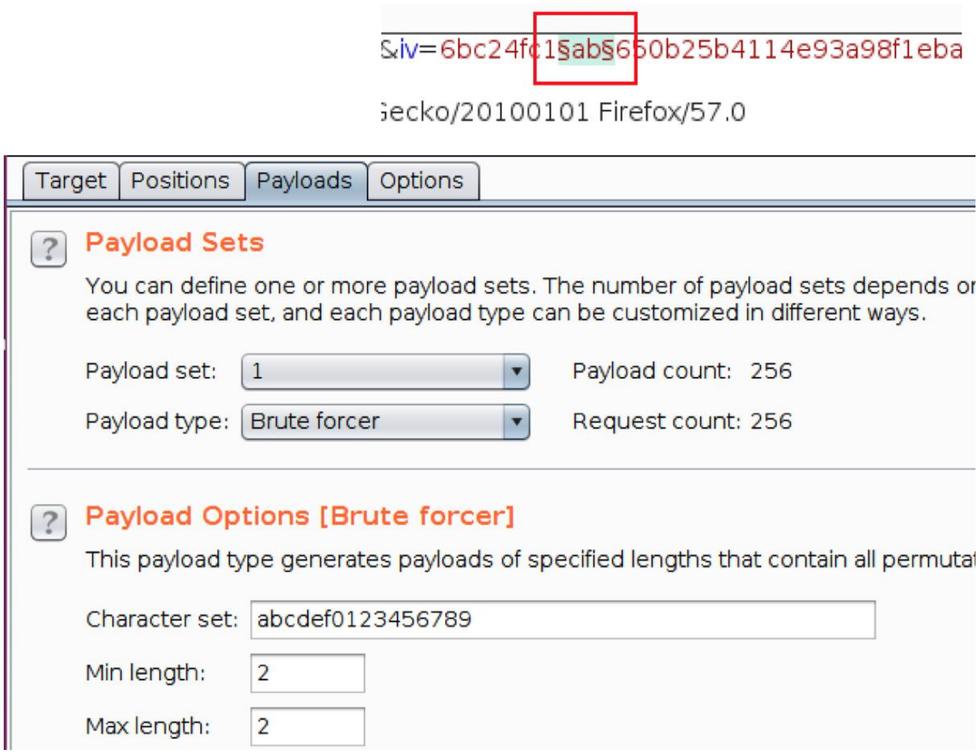
Table 4: Bytes of Initialization Vector by Position and Effect

Position	0	1	2	3	**4**	5	6	**7**	8	9	10	11	12	13	14	15
Value	6b	c2	4f	c1	**ab**	65	0b	**25**	b4	11	4e	93	a9	8f	1e	ba
Effect	A	1	B	2	**1**	0	0	**1**	0	0	?	?	?	?	?	?
Affected Field Name	**Application ID**				**User ID**			**Group ID**			?	?	?	?	?	?

Jeremy Druin, jdruin@gmail.com

2.3.3. How to modify the vulnerable bytes to win

The two IV bytes identified (***Table 4***) can be altered manually until the User ID and Group ID read "000". The bytes are encoded as ASCII hex so there are 256 values for each byte ranging from 00 to FF. However, switching the value and resubmitting the URL up to 512 times could be tiresome. Burp-Suite can try all possible values of each byte (***Figure 21***); however, the Community Edition throttles the Intruder (PortSwigger, 2016). Without the "Pro" license, the Intruder operates too slowly when fuzzing with more than about 25 values.



&iv=6bc24fc1§ab§650b25b4114e93a98f1eba

Gecko/20100101 Firefox/57.0

Payload Sets

You can define one or more payload sets. The number of payload sets depends o
each payload set, and each payload type can be customized in different ways.

Payload set: 1 Payload count: 256

Payload type: Brute forcer Request count: 256

Payload Options [Brute forcer]

This payload type generates payloads of specified lengths that contain all permuta

Character set: abcdef0123456789

Min length: 2

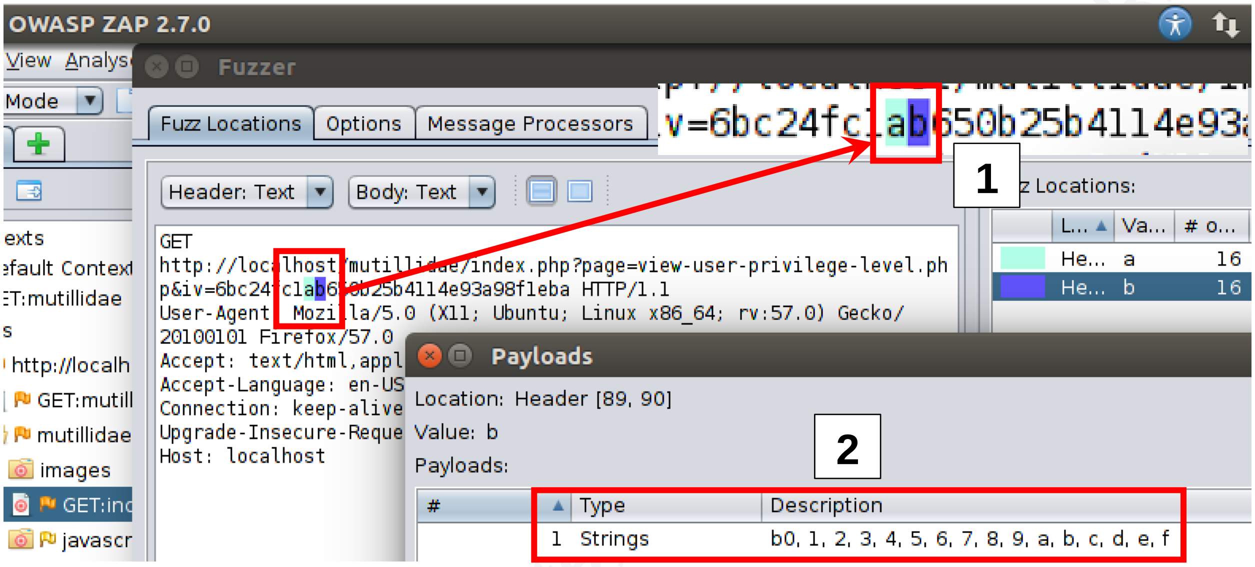
Max length: 2

Figure 21: Burp-Suite Intruder Brute-Force payload configured to try ASCII hex values 00 - FF

The OWASP Zed Attack Proxy (ZAP) [21] is a free interception proxy that contains a versatile fuzzing tool (McRee, 2011) [22]. A new ZAP Fuzzer is opened. IV byte 5 which has a default value of "ab" for is prepared for fuzzing. To send all possible values, the digits "a" and "b" are selected independently and configured so that values 0-9 and a-f will be injected (***Figure 22 - 1***). Sixteen values are inserted into each digit for a total of 256 combinations (***Figure 22 - 2***).

[21] OWASP ZAP Download: https://github.com/zaproxy/zaproxy/wiki/Downloads

[22] Please refer to ***Appendix B*** for video tutorials on notable OWASP ZAP features



```
GET
http://localhost/mutillidae/index.php?page=view-user-privilege-level.ph
p&iv=6bc24fc1ab650b25b4114e93a98f1eba HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:57.0) Gecko/
20100101 Firefox/57.0
Accept: text/html,appl
Accept-Language: en-US
Connection: keep-alive
Upgrade-Insecure-Reque
Host: localhost
```

v=6bc24fc1ab650b25b4114e93a

z Locations:

	L... ▲	Va...	# o...
🟩	He...	a	16
🟦	He...	b	16

Payloads

Location: Header [89, 90]

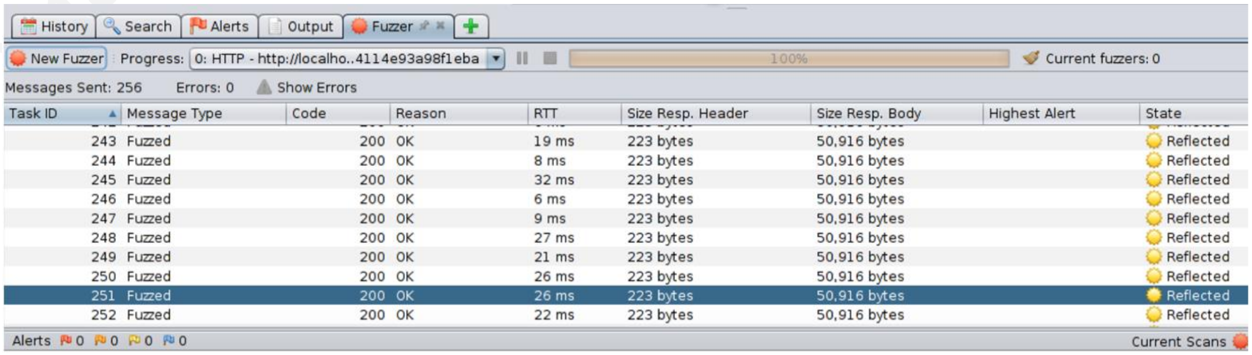
Value: b

Payloads:

#	Type	Description
1	Strings	b0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f

Figure 22: Using OWASP ZAP Fuzzer to inject all hex digits

The fuzzing results appear in the *ZAP Fuzzer* output window (***Figure 23***). Somewhere within is an HTTP response with a value of "000" for the User ID. The *ZAP Search* feature can locate strings inside the fuzz results. Inspection of the User ID output[23] shows the value is within an HTML table data element *<td style="text-align: left;">**100** (Hint: 0X31 0X30 0X30)</td>*. A search pattern such as "<td style="text-align: left;">**000**" or just ">**000**" is enough to locate the corresponding HTTP response (***Figure 24***).



Progress: 0: HTTP - http://localho..4114e93a98f1eba

Current fuzzers: 0

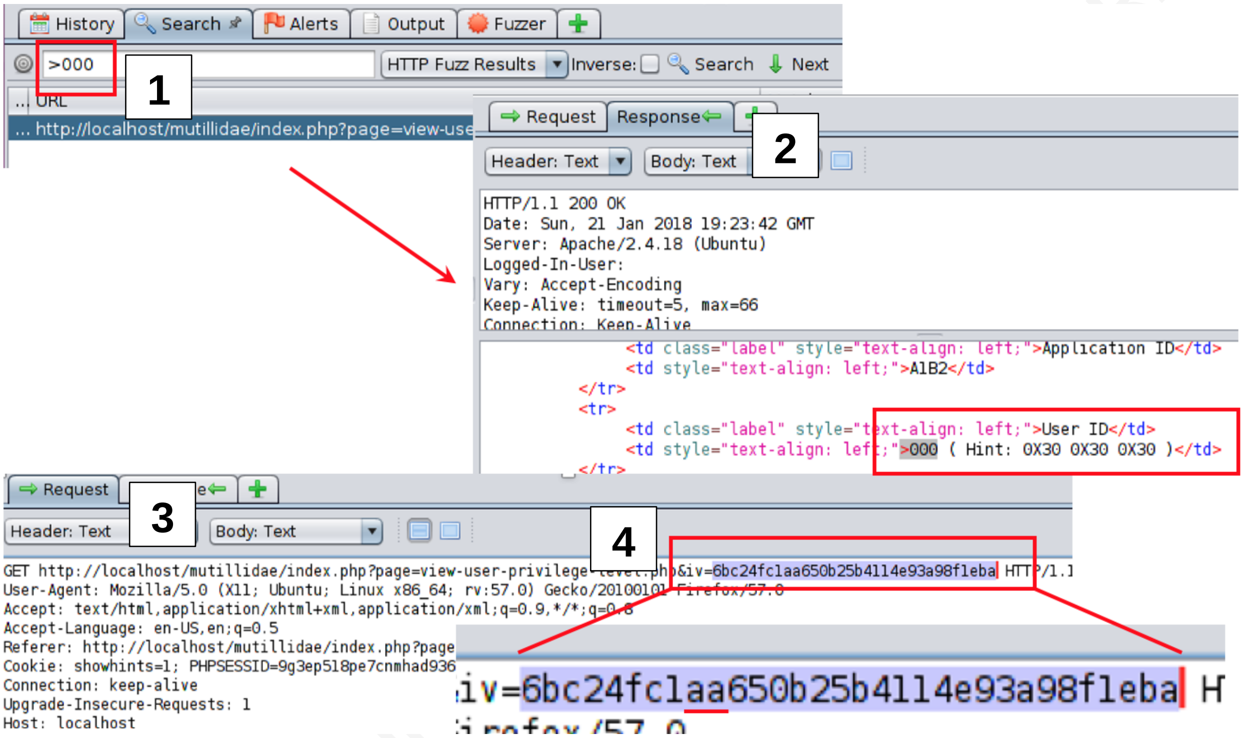
Messages Sent: 256 Errors: 0

Task ID ▲	Message Type	Code	Reason	RTT	Size Resp. Header	Size Resp. Body	Highest Alert	State
243	Fuzzed	200	OK	19 ms	223 bytes	50,916 bytes		Reflected
244	Fuzzed	200	OK	8 ms	223 bytes	50,916 bytes		Reflected
245	Fuzzed	200	OK	32 ms	223 bytes	50,916 bytes		Reflected
246	Fuzzed	200	OK	6 ms	223 bytes	50,916 bytes		Reflected
247	Fuzzed	200	OK	9 ms	223 bytes	50,916 bytes		Reflected
248	Fuzzed	200	OK	27 ms	223 bytes	50,916 bytes		Reflected
249	Fuzzed	200	OK	21 ms	223 bytes	50,916 bytes		Reflected
250	Fuzzed	200	OK	26 ms	223 bytes	50,916 bytes		Reflected
251	Fuzzed	200	OK	26 ms	223 bytes	50,916 bytes		Reflected
252	Fuzzed	200	OK	22 ms	223 bytes	50,916 bytes		Reflected

Alerts 0 0 0 0

Figure 23: Results from OWASP ZAP Fuzzer injecting all combinations of two hex digits into IV byte 5

[23] The View Source feature of the browser can show the HTML or the HTTP response can be viewed in Burp-Suite or OWASP ZAP



```
... http://localhost/mutillidae/index.php?page=view-use
```

```
HTTP/1.1 200 OK
Date: Sun, 21 Jan 2018 19:23:42 GMT
Server: Apache/2.4.18 (Ubuntu)
Logged-In-User:
Vary: Accept-Encoding
Keep-Alive: timeout=5, max=66
Connection: Keep-Alive
```

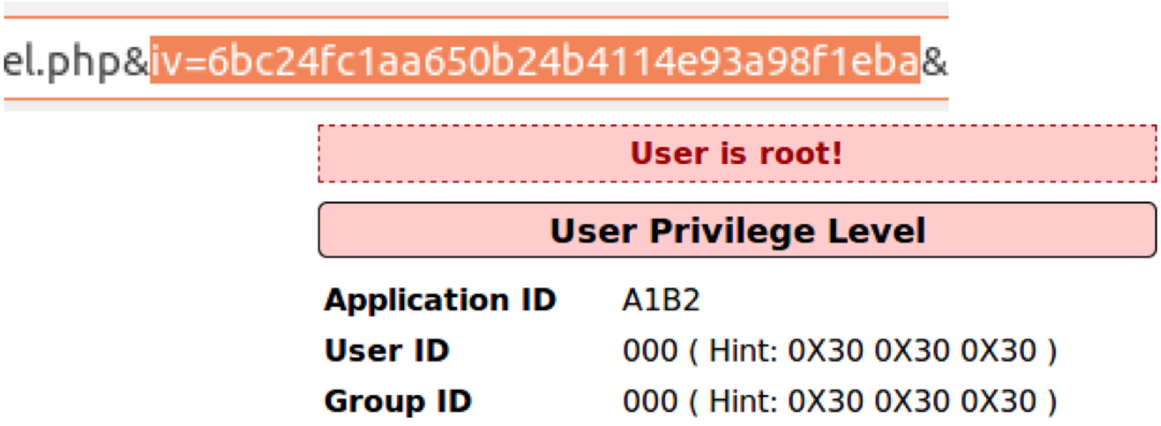
```
                <td class="label" style="text-align: left;">Application ID</td>
                <td style="text-align: left;">A1B2</td>
            </tr>
            <tr>
                <td class="label" style="text-align: left;">User ID</td>
                <td style="text-align: left;">000 ( Hint: 0X30 0X30 0X30 )</td>
            </tr>
```

```
GET http://localhost/mutillidae/index.php?page=view-user-privilege-level.php&iv=6bc24fc1aa650b25b4114e93a98f1eba HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:57.0) Gecko/20100101 Firefox/57.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Referer: http://localhost/mutillidae/index.php?page
Cookie: showhints=1; PHPSESSID=9g3ep518pe7cnmhad936
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Host: localhost
```

```
iv=6bc24fc1aa650b25b4114e93a98f1eba H
```

Figure 24: (1)(2) Using the "Search" feature of OWASP ZAP to locate fuzzing result with a User ID of "000". (3) The corresponding HTTP request shows the IV that produced the User ID. (4) The injected byte was "aa" (0XAA).

The digits needed in IV byte 8 to cause the Group ID to equal "000" is determined using the same method. Fuzzing will show the default value of 0x25 must be changed to 0x24. The "iv" parameter in the URL is updated with 6bc24fc1**aa**650b**24**b4114e93a98f1eba, then the page is submitted to the server. The User ID and Group ID fields match "000", so the game is won (***Figure** 25*).



el.php&iv=6bc24fc1aa650b24b4114e93a98f1eba&

User is root!

User Privilege Level

Application ID	A1B2
User ID	000 (Hint: 0X30 0X30 0X30)
Group ID	000 (Hint: 0X30 0X30 0X30)

Figure 25: Bytes 5 and 8 are updated to cause the User ID and Group ID to equal "000". The web page outputs "User is Root!" indicating the game is won.

Jeremy Druin, jdruin@gmail.com

3. Conclusion

Training environments like Mutillidae II and Damn Vulnerable Web Application (DVWA)[24] provide safe environments to practice complex penetration testing tasks. Particularly abstract concepts like CBC bit-flipping can be easier to learn by experience. The game provided allows the user to work through two bit-flipping challenges: one for novices and another requiring intermediate skill. Also, the game in Mutillidae lets the user practice remediating the vulnerability. Mutillidae also shows the user how to fix the issue. In "Security Level 5", Mutillidae does not accept an initialization vector from the user. The IV is stored server-side and any input from the user is ignored.

Mutillidae chooses to remediate the bit-flipping vulnerability by preventing the user from changing the initialization vector. Because the IV can be private to the server in this use-case, keeping the IV from the user works well to fix the issue. In many cases, the IV must be changed frequently and/or passed between systems. In systems where the IV is only required to be unique and unpredictable, the integrity must be protected (Dworkin, 2001). The sender can provide a Message Authentication Code (MAC) for the encrypted message (Encrypt-then-MAC) or use encryption that includes authentication such as AES-GCM (Ducklin, 2013).

Other games are available to help learn other security penetration testing and secure application development concepts. Mutillidae and DVWA have many other web application security challenges. VulnHub provides vulnerable virtual machines with "boot2root" games in which a participant uses operating system and network service vulnerabilities to practice penetration testing (g0tmi1k, 2018). Several sites list practice systems and sites are available[25] [26] [27] giving students of information security many scenarios and contexts to hone skills (Vonnegut, 2015) (Edwards, 2017) (Shukla, 2014). These systems help learn concepts that can be challenging to learn through academic study alone. They also provide "hands-on" exercises in a safe but realistic environment giving students an invaluable resource that might otherwise not be offered.

[24] http://www.dvwa.co.uk

Appendix A: Video Tutorials for Notable Burp-Suite Features

Title	URL
How to Install Burp Suite on Linux	https://www.youtube.com/watch?v=OsSPwe-DUOU
Introduction to using the Burp-Suite Targets Tab	https://www.youtube.com/watch?v=q3iG7YMjcmE
Introduction to Web Request and Response Interception with Burp-Suite	https://www.youtube.com/watch?v=qsE04AhlJrc
Introduction to Burp-Suite's Repeater Tool	https://www.youtube.com/watch?v=IstpobN5azo
Introduction to Burp-Suite Intruder's Character Frobber Payload	https://www.youtube.com/watch?v=7vWTEbOfa-8
Introduction to Burp-Suite Intruder's "Grep Extract" Feature	https://www.youtube.com/watch?v=t0uMReqs8Ng
Introduction to Burp-Suite Comparer Tool	https://www.youtube.com/watch?v=KxqY_bp13gc

Appendix B: Video Tutorials for Notable OWASP ZAP Features

Title	URL
How to Install OWASP Zap on Linux	https://www.youtube.com/watch?v=MpuFW_mkJ3M
How to Proxy Web Traffic through OWASP ZAP	https://www.youtube.com/watch?v=ICPqz1Al9fk
How to Spider a Web Site with OWASP ZAP	https://www.youtube.com/watch?v=pGCBivHNRn8
How to Intercept HTTP Requests with OWASP ZAP	https://www.youtube.com/watch?v=fa5LAfXmwoo
How to Fuzz Web Applications with OWASP ZAP (Part 1)	https://www.youtube.com/watch?v=uSfGeyJKIVA
How to Fuzz Web Applications with OWASP ZAP (Part 2)	https://www.youtube.com/watch?v=tBXX_GAK7BU

[25] https://www.checkmarx.com/2015/04/16/15-vulnerable-sites-to-legally-practice-your-hacking-skills/

[26] https://www.bonkersabouttech.com/security/40-plus-list-of-intentionally-vulnerable-websites-to-practice-your-hacking-skills/392

[27] https://wheresmykeyboard.com/2016/07/hacking-sites-ctfs-wargames-practice-hacking-skills/

References

abpolym. (2015, 3 25). *tinyCTF 2014: ECB, it's easy as 123*. Retrieved from GitHub: https://github.com/ctfs/write-ups-2014/tree/master/tinyctf-2014/ecb-its-easy-as-123

Barker, E., & Barker, W. (2016, 8). *Guideline for Using Cryptographic Standards in the Federal Government.* Retrieved from http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-175A.pdf

BEHRENS, M. (2014, 11 20). *Understanding the 3 Main Types of Encryption.* Retrieved from Atomic Object: https://spin.atomicobject.com/2014/11/20/encryption-symmetric-asymmetric-hashing/

Bhoge, J. P., & Chatur, D. P. (2014). *Avalanche Effect of AES Algorithm.* Retrieved from International Journal of Computer Science and Information Technologies, Vol. 5: http://ijcsit.com/docs/Volume%205/vol5issue03/ijcsit2014050394.pdf

Bunzel, A. F. (2018). *Modes of Operation*. Retrieved from Cryptography Academy: https://cryptographyacademy.com/modes-of-operation/

Chaudhary, S. (2014, 3 16). *Speeding Up WEP Hacking : ARP request replay attack*. Retrieved 4 1, 2017, from Kali Tutorials: http://www.kalitutorials.net/2014/03/speeding-up-wep-hacking-in-kali.html

Crowley, D. (2013, 1 17). *Defeating AES without a PhD*. Retrieved from SpiderLabs Blog: https://www.trustwave.com/Resources/SpiderLabs-Blog/Defeating-AES-without-a-PhD/

D.I. Management Services Pty Ltd. (2001). *Block Cipher Modes and Initialization Vectors*. Retrieved from CryptoSys API Library Manual: https://www.cryptosys.net/manapi/api_blockciphermodes.html

Diana-Lynn Contesti, D. A. (2007). *Official (ISC)2 Guide to the SSCP CBK.* Boca Raton, Fl, USA: Auerbach Publications.

Druin, J. (2012, 8 7). *Introduction to CBC Bit-Flipping Attack*, 1. (J. Druin, Editor, & J. Druin, Producer) Retrieved 4 1, 2017, from YouTube - webpwnized Channel: https://www.youtube.com/watch?v=TNt2rJcxdyg

Druin, J. (2017, 1 7). *OWASP Mutillidae II Web Pen-Test Practice Application*. (J. Druin, Producer) Retrieved 4 1, 2017, from SourceForge: https://sourceforge.net/projects/mutillidae/

Druin, J. (2017, 1 7). *view-user-privilege-level.php.* Retrieved from OWASP Mutillidae II Web Pen-Test Practice Application: https://sourceforge.net/p/mutillidae/git/ci/master/tree/view-user-privilege-level.php

Druin, J. (2018). *Listing of Vulnerabilities*. Retrieved from SourceForge: https://sourceforge.net/projects/mutillidae/files/documentation/listing-of-vulnerabilities-in-mutillidae.txt/download

Druin, J. (2018, 1 14). *OWASP Mutillidae II*. Retrieved from SourceForge: https://sourceforge.net/projects/mutillidae/

Druin, J. (2018). *OWASP Mutillidae II*. Retrieved from SourceForge: https://sourceforge.net/projects/mutillidae/

Ducklin, P. (2013, 2 7). *Boffins 'crack' HTTPS encryption in Lucky Thirteen attack.* Retrieved from Naked Security by Sophos: https://nakedsecurity.sophos.com/2013/02/07/boffins-crack-https-encryptionin-lucky-thirteen-attack/

Dworkin, M. (2001, 12). *Recommendation for Block Cipher Modes of Operation.* Retrieved from NIST Special Publication 800-38A 2001 Edition: http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf

Edwards, B. (2017, 1 2). *40+ INTENTIONALLY VULNERABLE WEBSITES TO (LEGALLY) PRACTICE YOUR HACKING SKILLS*. Retrieved from Bonkers About Tech: https://www.bonkersabouttech.com/security/40-plus-list-of-intentionally-vulnerable-websites-to-practice-your-hacking-skills/392

Electronics Tutorials. (2018). *Exclusive-OR Gate Tutorial*. Retrieved from Electronics Tutorials: https://www.electronics-tutorials.ws/logic/logic_7.html

Feltner, S. (2016, 12 28). *Single-factor Authentication (SFA) vs. Multi-factor Authentication (MFA)*. Retrieved 4 1, 2017, from CENTRIFY PERSPECTIVE: http://blog.centrify.com/sfa-mfa-difference/

Fielding, R., & Reschke, J. (2014, 6 1). *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*. Retrieved 4 1, 2017, from PROPOSED STANDARD: https://tools.ietf.org/html/rfc7230

Focardi, R. (2014). *Block cipher modes*. Retrieved from SecGroup Unive: https://secgroup.dais.unive.it/teaching/cryptography/block-cipher-modes/

g0tmi1k. (2018). *About Vulnhub*. Retrieved from Vulnhub: https://www.vulnhub.com/about/

General. (2017, 4 1). *Block cipher mode of operation*. Retrieved 4 1, 2017, from Wikipedia, the free encyclopedia: https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation#Electronic_Codebook_.28ECB.29

Gerard, N. (2018, 1 31). *Cryptography - Nonce (Number Only used once)*. Retrieved from Gerardnico: https://gerardnico.com/wiki/security/nonce

Germundsson, R. a. (2002). *XOR*. Retrieved from MathWorld - A Wolfram Web Resource: http://mathworld.wolfram.com/XOR.html

Gjertsen, E. G., Gjære, E. A., Bartnes, M., & Flores, W. R. (2017). *Gamification of Information Security Awareness and Training.* Retrieved from brage.bibsys.no: https://brage.bibsys.no/xmlui/bitstream/handle/11250/2462736/ICISSP_2017-gamification.pdf?sequence=2

GOODIN, D. (2016, 10 23). *Using Rowhammer bitflips to root Android phones is now a thing*. Retrieved from Ars Technica: https://arstechnica.com/information-technology/2016/10/using-rowhammer-bitflips-to-root-android-phones-is-now-a-thing/

Gordon, A. (2015). *Official (ISC)2 Guide to the CISSP CBK, Fourth Edition.* Boca Raton, FL: CRC Press.

Griffiths, T., & Guile, D. (2004). *Learning through work experience for the knowledge economy.* Luxembourg: Office for Official Publications of the European Communities.

Heaton, R. (2013, 7 29). *The Padding Oracle Attack - why crypto is terrifying*. Retrieved 4 1, 2017, from Rob Heaton: http://robertheaton.com/2013/07/29/padding-oracle-attack/

Hudde, H. C. (2009, 2 18). *Building Stream Ciphers from Block Ciphers and their Security.* Retrieved from www.emsec.rub.de: https://www.emsec.rub.de/media/crypto/attachments/files/2011/03/hudde.pdf

IBM. (2018). *IBM Knowledge Center*. Retrieved from Electronic Code Book (ECB) Mode: https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.3.0/com.ibm.zos.v2r3.csfb400/csfb4429.htm

Investopedia. (2018). *Pareto Principle*. Retrieved from Investopedia: https://www.investopedia.com/terms/p/paretoprinciple.asp

Kowalczyk, C. (2017). *Block Ciphers Modes of Operation*. Retrieved from Crypto-IT: http://www.crypto-it.net/eng/theory/modes-of-block-ciphers.html

Laboratories, R. (2017, 1 1). *2.1.4.3 WHAT IS CIPHER BLOCK CHAINING MODE?* (D. EMC, Producer, & Dell EMC) Retrieved 4 1, 2017, from RSA Laboratories: http://www.rsacertificate.com/emc-plus/rsa-labs/standards-initiatives/what-is-cipher-block-chaining-mode.htm

Marclass. (2015, 6 25). *Blog*. Retrieved from MultiBit: https://multibit.org/blog/2015/07/25/bit-flipping-attack.html

McRee, R. (2011, 11). *OWASP ZAP – Zed Attack Proxy*. Retrieved from Toolsmith: https://holisticinfosec.org/toolsmith/pdf/november2011.pdf

Mehmood, A. (2017, 10 27). *Differences between Hash functions, Symmetric & Asymmetric Algorithms.* Retrieved from Cryptomathic: https://www.cryptomathic.com/news-events/blog/differences-between-hash-functions-symmetric-asymmetric-algorithms

Microsoft. (2017, 1 7). *Description of Symmetric and Asymmetric Encryption.* Retrieved from Microsoft: https://support.microsoft.com/en-us/help/246071/description-of-symmetric-and-asymmetric-encryption

Miles, E., & Viola, E. (2015, 8 20). *Substitution-permutation networks, pseudorandom functions, and natural proofs*. Retrieved 4 2, 2017, from http://www.ccs.neu.edu: http://www.ccs.neu.edu/home/viola/papers/spn.pdf

Morris, R., & Thompson, K. (1978, 4 3). *Password Security: A Case History*. Retrieved from Internet Archive: https://web.archive.org/web/20130821093338/http://cm.bell-labs.com/cm/cs/who/dmr/passwd.ps

Perrin, C. (2010, 2 1). *The use and misuse of the XOR stream cipher*. Retrieved from TechRepublic: https://www.techrepublic.com/blog/it-security/the-use-and-misuse-of-the-xor-stream-cipher/

Plummer, D. C. (1982, 11 1). *An Ethernet Address Resolution Protocol*. Retrieved 4 1, 2017, from INTERNET STANDARD IETF: https://tools.ietf.org/html/rfc826

PortSwigger. (2016, 2 8). *Burp Intruder Bruteforcing too slowly*. Retrieved from PortSwigger: https://support.portswigger.net/customer/portal/questions/16147440-burp-intruder-bruteforcing-too-slowly

Portswigger. (2018). *Payload Types*. Retrieved from Portswigger Web Security: https://portswigger.net/burp/help/intruder_payloads_types#charfrobber

PortSwigger. (2018). *Payload Types - Character Frobber*. Retrieved from PortSwigger: https://portswigger.net/burp/help/intruder_payloads_types#charfrobber

Regalado, D. (2013, 8 22). *CBC Byte Flipping Attack—101 Approach*. Retrieved 4 1, 2017, from InfoSec Institute: http://resources.infosecinstitute.com/cbc-byte-flipping-attack-101-approach/#gref

Regalado, D. (2013, 8 22). *CBC Byte Flipping Attack—101 Approach*. Retrieved from InfoSec Institute: http://resources.infosecinstitute.com/cbc-byte-flipping-attack-101-approach/#gref

Rogaway, P. (2011, 2 10). *Evaluation of Some Blockcipher Modes of Operation.* Retrieved from University of California, Davis Dept. of Computer Science: http://web.cs.ucdavis.edu/~rogaway/papers/modes.pdf

Sharma, V. (2007, 11 27). *Block Ciphers: Simple attack on ECB mode.* Retrieved from Varun Sharma's security blog: https://blogs.msdn.microsoft.com/varun_sharma/2007/11/27/block-ciphers-simple-attack-on-ecb-mode/

Shukla, T. (2014). *22 Hacking Sites, CTFs and Wargames To Practice Your Hacking Skills*. Retrieved from WheresMyKeyboard?: https://wheresmykeyboard.com/2016/07/hacking-sites-ctfs-wargames-practice-hacking-skills/

Stallings, W. (2017, 1 1). *Cryptography and Network Security Chapter 3*. Retrieved 4 1, 2017, from www.cs.man.ac.uk: http://www.cs.man.ac.uk/~banach/COMP61411.Info/CourseSlides/Wk2.1.DES.pdf

The Linux Information Project. (2005, 7 22). *User ID Definition*. (T. L. Project, Producer) Retrieved 4 1, 2017, from LINFO: http://www.linfo.org/uid.html

Thijssen, J. (2010, 12 8). *Encryption operating modes: ECB vs CBC*. Retrieved from A Day in the Life of: https://adayinthelifeof.nl/2010/12/08/encryption-operating-modes-ecb-vs-cbc/

Titcomb, J. (2016, 3 22). *Do you have one of the most common passwords? They're ridiculously easy to guess*. Retrieved 4 1, 2017, from The Telegraph: http://www.telegraph.co.uk/technology/2016/01/26/most-common-passwords-revealed---and-theyre-ridiculously-easy-to/

Tutorials Point. (2018). *Block Cipher Modes of Operation*. Retrieved from Tutorials Point: https://www.tutorialspoint.com/cryptography/block_cipher_modes_of_operation.htm

TYSON, J. (2018). *How Encryption Works*. Retrieved from How Stuff Works : https://computer.howstuffworks.com/encryption2.htm

Vonnegut, S. (2015, 4 16). *15 Vulnerable Sites To (Legally) Practice Your Hacking Skills*. Retrieved from CheckMarx: https://www.checkmarx.com/2015/04/16/15-vulnerable-sites-to-legally-practice-your-hacking-skills/

Wikipedia. (2017, 3 29). *Hexadecimal*. Retrieved 4 1, 2017, from Wikipedia, the free encyclopedia: https://en.wikipedia.org/wiki/Hexadecimal

Young, D. B. (2018). *Stream and Block Encryption*. Retrieved from Foundations of Computer Security: https://www.cs.utexas.edu/~byoung/cs361/lecture45.pdf