Threat Hunting

HW06

Name: 劉定睿 ID: M11409313

Date: 2025-10-19

目錄

There's different kind of Domain name generation method use by DGA tools. In the real world scenario. Attacker can change their domain name if it was record by the IoC. So attacker can evasion the defense. According to the above, we can generating the "mutation" domain name to hunting the threat.

Generation

Zodiac-based DGA Generation

```
Zodiac-based DGA Generation
Explanation: This DGA generates domains based on zodiac signs and random
strings.
Usefulness: This type of DGA can be useful for malware authors to create a
unique set of domains for command and control (C&C) communication or data
exfiltration.
Strengths: The use of zodiac signs and random strings makes the generated
domains unpredictable and difficult to block.
Weaknesses: The zodiac sign pattern may be detectable, and the random string
generation algorithm may be reverse-engineered.
Deception: This DGA can be deciphered by analyzing the pattern of generated
domains and identifying the relationship between the zodiac signs and dates.
Random seed: $-i]b"}B,R*KG!?,
www.2de154bbd.net
www.b66c09d9.biz
www.18d851v.com
www.b109635x8.net
www.f4w8t15.biz
www.f644d25ac8.org
www.04996c0d9b.biz
www.935052844.net
www.8049b2gd.org
www.f6c122b.org

Regex for Zodiac Sign Generated DGAs: [r|w|e|o|b|c|d|i|g|n|f|t|z]+|[9|4|0|8|2|3|5|6|1]+|[^mxav.]{0,2}{1,3}

Yara rule:
 rule dga_domain_detection {
    meta:
        description = "Detects DGA-generated domain names"
    strings:
        $dga_regex = /[r|w|e|o|b|c|d|i|g|n|f|t|z]+|[9|4|0|8|2|3|5|6|1]+|[^mxav.]{0,2}{1,3}/ nocase
    condition:
        $dga_regex
}
```

Time-based DGA Generation

```
Time-based DGA Generation
Explanation: This DGA generates domains based on the current time.
Usefulness: This type of DGA can be used by malware authors to create a set of
constantly changing domains for C&C communication or data exfiltration.
Strengths: The generated domains change frequently, making it difficult to block
them all.
Weaknesses: The time-based pattern may be detectable, and the generated domains
may be predictable if the algorithm is known.
Deception: This DGA can be deciphered by analyzing the pattern of generated
domains and identifying the relationship between the domains and the current
time.
www.2024031819.info
www.2024031819.org
www.2024031.org
www.2024031819.info
www.2024031.net
www.2024031819.biz
www.2024031.biz
www.2024031819.biz
www.202403181.com
www.20240318.biz

Regex for Time-based Generated DGAs: [r|w|o|b|i|g|n|f|z]+|[9|4|0|8|2|3|1]+|[^mect.]{0,2}{1,3}

Yara rule:
 rule dga_domain_detection {
    meta:
        description = "Detects DGA-generated domain names"
    strings:
        $dga_regex = /[r|w|o|b|i|g|n|f|z]+|[9|4|0|8|2|3|1]+|[^mect.]{0,2}{1,3}/ nocase
    condition:
        $dga_regex
}


Note: Can use  https://riskmitigation.ch/yara-scan/ or MalwareBazaar https://bazaar.abuse.ch/ but will need API Key
```

Seed-based DGA Generation

```
Seed-based DGA Generation
Explanation: This DGA generates domains based on a seed value and a hash
function.
Usefulness: This type of DGA can be useful for malware authors to create a
unique set of domains for C&C communication or data exfiltration, based on a
shared seed value.
Strengths: The generated domains are unpredictable without knowledge of the seed
value and the hash function used.
Weaknesses: If the seed value or the hash function is compromised, the generated
domains can be predicted.
Deception: This DGA can be deciphered by analyzing the pattern of generated
domains and attempting to reverse-engineer the seed value and the hash function.
Random seed: G+/Mn@IrfnvN%(C)
www.ffa1a30f.net
www.dd9a96b.info
www.8fdee18b65.biz
www.96d6e8b2c.biz
www.1ca4b60.net
www.4d5f90ef2.biz
www.3774bea66.org
www.6e5a4085.net
www.fb6e92d.biz
www.596200c.org

Regex for Seed-based Generated DGAs: [r|w|e|o|a|b|c|d|i|g|n|f|t|z]+|[9|4|0|8|2|3|5|7|6|1]+|[^.]{0,2}{1,3}

Yara rule:
 rule dga_domain_detection {
    meta:
        description = "Detects DGA-generated domain names"
    strings:
        $dga_regex = /[r|w|e|o|a|b|c|d|i|g|n|f|t|z]+|[9|4|0|8|2|3|5|7|6|1]+|[^.]{0,2}{1,3}/ nocase
    condition:
        $dga_regex
}


Note: Can use  https://riskmitigation.ch/yara-scan/ or MalwareBazaar https://bazaar.abuse.ch/ but will need API Key
```

Dictionary-based DGA Generation

```
Dictionary-based DGA Generation
Explanation: This DGA generates domains by combining random words from a
predefined dictionary.
Usefulness: This type of DGA can be useful for malware authors to create domains
that appear more human-readable and less suspicious.
Strengths: The generated domains are more likely to bypass filters and appear
legitimate.
Weaknesses: The dictionary used may be known or detectable, and the pattern of
combining words may be recognizable.
Deception: This DGA can be deciphered by analyzing the pattern of generated
domains and identifying the dictionary used and the word combination algorithm.
www.birdfish.com
www.blue.org
www.birdred.org
www.reddog.biz
www.bird.info
www.fishfish.net
www.bluered.net
www.fishred.org
www.redfish.org
www.bluedog.org

Regex for Dictionary-based Generated DGAs: [r|w|o|u|e|b|d|i|l|g|n|f|h|t|s]+|[^c.zm]{0,2}{1,3}

Yara rule:
 rule dga_domain_detection {
    meta:
        description = "Detects DGA-generated domain names"
    strings:
        $dga_regex = /[r|w|o|u|e|b|d|i|l|g|n|f|h|t|s]+|[^c.zm]{0,2}{1,3}/ nocase
    condition:
        $dga_regex
}


Note: Can use  https://riskmitigation.ch/yara-scan/ or MalwareBazaar https://bazaar.abuse.ch/ but will need API Key
```

Pseudorandom Number Generator (PRNG) based DGA Generation

```
Pseudorandom Number Generator (PRNG) based DGA Generation
Explanation: This DGA generates domains using a pseudorandom number generator
(PRNG) seeded with a specific value.
Usefulness: This type of DGA can be useful for malware authors to create a
unique set of domains for C&C communication or data exfiltration, based on a
shared seed value.
Strengths: The generated domains are unpredictable without knowledge of the seed
value and the PRNG algorithm used.
Weaknesses: If the seed value or the PRNG algorithm is compromised, the
generated domains can be predicted.
Deception: This DGA can be deciphered by analyzing the pattern of generated
domains and attempting to reverse-engineer the seed value and the PRNG
algorithm.
www.atxmr1kxh1.org
www.9lwf706k.net
www.emvub3vb.biz
www.aaklsg0glv.net
www.coh5n9lg1.biz
www.cktajow8.org
www.vuadd2fy.net
www.xn6h4da.info
www.09mxh9jf5.net
www.onkksi8f7.com

Regex for Pseudorandom Number Generator (PRNG) based Generated DGAs: [v|d|y|l|w|o|x|a|e|i|g|n|m|k|b|c|j|f|t|s|r|u|h|z]+|[9|0|8|5|7|6|1]+|[^423q.]{0,2}{1,3}

Yara rule:
 rule dga_domain_detection {
    meta:
        description = "Detects DGA-generated domain names"
    strings:
        $dga_regex = /[v|d|y|l|w|o|x|a|e|i|g|n|m|k|b|c|j|f|t|s|r|u|h|z]+|[9|0|8|5|7|6|1]+|[^423q.]{0,2}{1,3}/ nocase
    condition:
        $dga_regex
}


Note: Can use  https://riskmitigation.ch/yara-scan/ or MalwareBazaar https://bazaar.abuse.ch/ but will need API Key
```

Arithmetic-based DGA Generation

```
Arithmetic-based DGA Generation
Explanation: This DGA generates domains by performing arithmetic operations on a
base value and a random number.
Usefulness: This type of DGA can be useful for malware authors to create a
unique set of domains for C&C communication or data exfiltration, based on a
shared base value.
Strengths: The generated domains are unpredictable without knowledge of the base
value and the arithmetic operation used.
Weaknesses: If the base value or the arithmetic operation is compromised, the
generated domains can be predicted.
Deception: This DGA can be deciphered by analyzing the pattern of generated
domains and attempting to reverse-engineer the base value and the arithmetic
operation.
www.0012595.info
www.0012634.org
www.0000012962.com
www.0000013254.info
www.00012631.net
www.000012472.org
www.0000012625.net
www.00012627.info
www.0000012970.org
www.0012568.info

Regex for Arithmetic-based Generated DGAs: [r|w|o|e|i|g|n|f|t]+|[9|4|0|2|3|5|7|6|1]+|[^8c.m]{0,2}{1,3}

Yara rule:
 rule dga_domain_detection {
    meta:
        description = "Detects DGA-generated domain names"
    strings:
        $dga_regex = /[r|w|o|e|i|g|n|f|t]+|[9|4|0|2|3|5|7|6|1]+|[^8c.m]{0,2}{1,3}/ nocase
    condition:
        $dga_regex
}


Note: Can use  https://riskmitigation.ch/yara-scan/ or MalwareBazaar https://bazaar.abuse.ch/ but will need API Key
```

Permutation-based DGA Generation

```
Permutation-based DGA Generation
Explanation: This DGA generates domains by permuting the characters of a base
domain.
Usefulness: This type of DGA can be useful for malware authors to create a
unique set of domains for C&C communication or data exfiltration, based on a
shared base domain.
Strengths: The generated domains are unpredictable without knowledge of the base
domain and the permutation algorithm used.
Weaknesses: If the base domain or the permutation algorithm is compromised, the
generated domains can be predicted.
Deception: This DGA can be deciphered by analyzing the pattern of generated
domains and attempting to reverse-engineer the base domain and the permutation
algorithm.
www.elxempa.com
www.eelxampmm.org
www.eaxmelpm.net
www.explaemee.biz
www.epaelmxee.org
www.ampelexaaa.info
www.lmaxepe.com
www.leeamxplll.com
www.plamxee.biz
www.mpaxelem.org

Regex for Permutation-based Generated DGAs: [r|w|m|e|x|a|o|b|c|p|i|l|g|n|z]+|[^t.f]{0,2}{1,3}

Yara rule:
 rule dga_domain_detection {
    meta:
        description = "Detects DGA-generated domain names"
    strings:
        $dga_regex = /[r|w|m|e|x|a|o|b|c|p|i|l|g|n|z]+|[^t.f]{0,2}{1,3}/ nocase
    condition:
        $dga_regex
}


Note: Can use  https://riskmitigation.ch/yara-scan/ or MalwareBazaar https://bazaar.abuse.ch/ but will need API Key
```

Base32/Base64 DGA Generation

```
Fibonacci-based Generator DGA Generation


    Base32/Base64 DGA Generation
    Explanation: This DGA generates domains by encoding a seed value using Base32 or
    Base64 encoding.
    Usefulness: This type of DGA can be useful for malware authors to create a
    unique set of domains for C&C communication or data exfiltration, based on a
    shared seed value.
    Strengths: The generated domains are unpredictable without knowledge of the seed
    value and the encoding scheme used.
    Weaknesses: If the seed value or the encoding scheme is compromised, the
    generated domains can be predicted.
    Deception: This DGA can be deciphered by analyzing the pattern of generated
    domains and attempting to reverse-engineer the seed value and the encoding
    scheme.
    www.bXlzZWVk56.info
    www.ZGVlc3l.org
    www.bXlzZWVk.gov
    www.ZGVlc3lt.com
    www.bXlzZWVk.info
    www.ZGVlc3lt9.org
    www.bXlzZWVk.edu
    www.ZGVlc3lt.org
    www.bXlzZWVk17.info
    www.ZGVlc3lt2.edu

    Regex for Base32/Base64 Generated DGAs: [X|d|l|w|o|G|e|W|Z|i|g|n|k|b|c|f|V|t|r|u|z]+|[3]+|[^9mv257.61]{0,2}{1,3}

    Yara rule:
     rule dga_domain_detection {
        meta:
            description = "Detects DGA-generated domain names"
        strings:
            $dga_regex = /[X|d|l|w|o|G|e|W|Z|i|g|n|k|b|c|f|V|t|r|u|z]+|[3]+|[^9mv257.61]{0,2}{1,3}/ nocase
        condition:
            $dga_regex
    }


    Note: Can use  https://riskmitigation.ch/yara-scan/ or MalwareBazaar https://bazaar.abuse.ch/ but will need API Key
```

Wordlist-based DGA Generation

```
Wordlist-based DGA Generation
Explanation: This DGA generates domains by combining random words from a
predefined wordlist.
Usefulness: This type of DGA can be useful for malware authors to create domains
that appear more human-readable and less suspicious.
Strengths: The generated domains are more likely to bypass filters and appear
legitimate.
Weaknesses: The wordlist used may be known or detectable, and the pattern of
combining words may be recognizable.
Deception: This DGA can be deciphered by analyzing the pattern of generated
domains and identifying the wordlist used and the word combination algorithm.
www.figdate.net
www.cherrykiwi.org
www.grapemango.org
www.appleapple.gov
www.grapeapple.net
www.cherryfig.org
www.mangolemon.com
www.figkiwi.gov
www.figmango.net
www.elderberry.info

Regex for Wordlist-based Generated DGAs: [v|d|y|l|w|e|o|a|i|g|n|m|k|c|p|f|t|r|h]+|[^b.]{0,2}{1,3}

Yara rule:
 rule dga_domain_detection {
    meta:
        description = "Detects DGA-generated domain names"
    strings:
        $dga_regex = /[v|d|y|l|w|e|o|a|i|g|n|m|k|c|p|f|t|r|h]+|[^b.]{0,2}{1,3}/ nocase
    condition:
        $dga_regex
}


Note: Can use  https://riskmitigation.ch/yara-scan/ or MalwareBazaar https://bazaar.abuse.ch/ but will need API Key
```

Vowel-Consonant DGA Generation

```
Vowel-Consonant DGA Generation
Explanation: This DGA generates domains by alternating between vowels and
consonants.
Usefulness: This type of DGA can be useful for malware authors to create domains
that appear more human-readable and less suspicious.
Strengths: The generated domains are more likely to bypass filters and appear
legitimate.
Weaknesses: The alternating vowel-consonant pattern may be detectable, and the
character sets used may be known.
Deception: This DGA can be deciphered by analyzing the pattern of generated
domains and identifying the vowel-consonant alternation algorithm and the
character sets used.
www.igiforeb.net
www.oqudugata.net
www.ekehavale.info
www.owazosu.gov
www.ejayutahas.info
www.agozacom.org
www.oguwoqay.com
www.umulizew.info
www.iteseko.org
www.ubexotel.org

Regex for Vowel-Consonant Generated DGAs: [v|y|l|w|o|e|a|i|g|n|q|m|k|b|c|f|t|s|r|u|h|z]+|[^j.xd]{0,2}{1,3}

Yara rule:
 rule dga_domain_detection {
    meta:
        description = "Detects DGA-generated domain names"
    strings:
        $dga_regex = /[v|y|l|w|o|e|a|i|g|n|q|m|k|b|c|f|t|s|r|u|h|z]+|[^j.xd]{0,2}{1,3}/ nocase
    condition:
        $dga_regex
}


Note: Can use  https://riskmitigation.ch/yara-scan/ or MalwareBazaar https://bazaar.abuse.ch/ but will need API Key
```

Morse Code DGA Generation

```
Morse Code DGA Generation
Explanation: This DGA generates domains using Morse code representation of
characters.
Usefulness: This type of DGA can be useful for malware authors to create domains
that appear obfuscated and less suspicious.
Strengths: The generated domains are less likely to be recognized as DGA domains
and may bypass filters.
Weaknesses: The Morse code pattern may be detectable, and the character mapping
may be known or reverse-engineered.
Deception: This DGA can be deciphered by analyzing the pattern of generated
domains and identifying the Morse code character mapping.
www.-.-.-.----.--..----.---..-.--.-..org
www.-....-...----.-..---.......---.--..net
www...-.--..-.--...-----....-....-....-..org
www.----..--.--.--.-..--.-...--------.--.info
www..-...--..---.--.-..---..........org
www..----....---...-.-------..------...-..info
www.-....---....-.--.......----..----..info
www.--.---..-..--.-.-.-.-..-...-...-..net
www.--.--...----...-.-.-..---........info
www.-.---...--..-.-...--..---........--.--.edu

Regex for Morse Code Generated DGAs: [r|w|o|e|i|g|n|f|t]+|[^-.ud]{0,2}{1,3}

Yara rule:
 rule dga_domain_detection {
    meta:
        description = "Detects DGA-generated domain names"
    strings:
        $dga_regex = /[r|w|o|e|i|g|n|f|t]+|[^-.ud]{0,2}{1,3}/ nocase
    condition:
        $dga_regex
}


Note: Can use  https://riskmitigation.ch/yara-scan/ or MalwareBazaar https://bazaar.abuse.ch/ but will need API Key
```

Emoji DGA Generation

```
Emoji DGA Generation
Explanation: This DGA generates domains using emojis.
Usefulness: This type of DGA can be useful for malware authors to create domains
that appear obfuscated and less suspicious.
Strengths: The generated domains are less likely to be recognized as DGA domains
and may bypass filters.
Weaknesses: The emoji pattern may be detectable, and the emoji set used may be
known or reverse-engineered.
Deception: This DGA can be deciphered by analyzing the pattern of generated
domains and identifying the emoji set used.
www.💡💡🙌😀😀🚀🤗😀🙌.net
www.🚀🌎🤗😂😍😍🤗.gov
www.😍🎉😍🎉🎉🙌👍🙌.net
www.👍😀🙌👍🙌😍🚀😀.info
www.😂🚀🎉🤗😍🙌🤗👍😀🤗.gov
www.🎉💡😂🌎🚀👍💡😀.org
www.😍😍💡🙌🤗😀🚀🤗.com
www.😀😂🤗🚀😍😂😍🎉🤗🌎.com
www.🚀🙌🙌🚀🌎😂🌎🌎.info
www.😂😀🤗🎉🤗🌎😂🙌.org

Regex for Emoji Generated DGAs: [r|w|m|e|o|c|v|i|g|n|f|t]+|[^🙌🎉😀😂💡😍🤗🚀👍.🌎]{0,2}{1,3}

Yara rule:
 rule dga_domain_detection {
    meta:
        description = "Detects DGA-generated domain names"
    strings:
        $dga_regex = /[r|w|m|e|o|c|v|i|g|n|f|t]+|[^🙌🎉😀😂💡😍🤗🚀👍.🌎]{0,2}{1,3}/ nocase
    condition:
        $dga_regex
}


Note: Can use  https://riskmitigation.ch/yara-scan/ or MalwareBazaar https://bazaar.abuse.ch/ but will need API Key
```

Coordinate-based DGA

```
Coordinate-based DGA
Explanation: This DGA generates domains using coordinates.
Usefulness: This type of DGA can be useful for malware authors to create domains
that appear obfuscated and less suspicious.
Strengths: The generated domains are less likely to be recognized as DGA domains
and may bypass filters.
Weaknesses: The coordinate pattern may be detectable, and the coordinate range
and format may be known or reverse-engineered.
Deception: This DGA can be deciphered by analyzing the pattern of generated
domains and identifying the coordinate range and format used.
www.3430861388.info
www.3798493944.gov
www.6843572356.edu
www.7555993606.edu
www.9831958260.net
www.3361956922.info
www.7156758525.com
www.0826384271.com
www.7031541547.org
www.5425648304.gov

Regex for Coordinate-based Generated DGAs: [w|m|o|e|u|c|v|d|i|g|n|f]+|[9|4|0|8|2|3|5|7|6|1]+|[^rt.]{0,2}{1,3}

Yara rule:
 rule dga_domain_detection {
    meta:
        description = "Detects DGA-generated domain names"
    strings:
        $dga_regex = /[w|m|o|e|u|c|v|d|i|g|n|f]+|[9|4|0|8|2|3|5|7|6|1]+|[^rt.]{0,2}{1,3}/ nocase
    condition:
        $dga_regex
}


Note: Can use  https://riskmitigation.ch/yara-scan/ or MalwareBazaar https://bazaar.abuse.ch/ but will need API Key
```

Musical Notes DGA

```
Musical Notes DGA
Explanation: This DGA generates domains using musical notes and octaves.
Usefulness: This type of DGA can be useful for malware authors to create domains
that appear obfuscated and less suspicious.
Strengths: The generated domains are less likely to be recognized as DGA domains
and may bypass filters.
Weaknesses: The musical note pattern may be detectable, and the note and octave
sets used may be known or reverse-engineered.
Deception: This DGA can be deciphered by analyzing the pattern of generated
domains and identifying the note and octave sets used.
www.B3G3C4F2G5E3C1G2C5E1.net
www.C4G3G2F1G1F5A3A3.info
www.E5D5D2F4E1E4A3B5C5F2.info
www.C4E2B4D3E2A1B3E1.com
www.F1A5G1F5E2B1E3D3C3.com
www.F3A3A1C4F2B2D4.org
www.A2C5D5G5B5C5G1.net
www.F3B2D3C1B4G5E2.com
www.A1D3A1B3D3E4C4.net
www.A3F1F1E5D4D2E5.gov

Regex for Musical Notes Generated DGAs: [w|m|e|G|o|D|c|A|C|i|E|g|B|n|f|t|F]+|[4|2|3|5|1]+|[^r.v]{0,2}{1,3}

Yara rule:
 rule dga_domain_detection {
    meta:
        description = "Detects DGA-generated domain names"
    strings:
        $dga_regex = /[w|m|e|G|o|D|c|A|C|i|E|g|B|n|f|t|F]+|[4|2|3|5|1]+|[^r.v]{0,2}{1,3}/ nocase
    condition:
        $dga_regex
}


Note: Can use  https://riskmitigation.ch/yara-scan/ or MalwareBazaar https://bazaar.abuse.ch/ but will need API Key
```

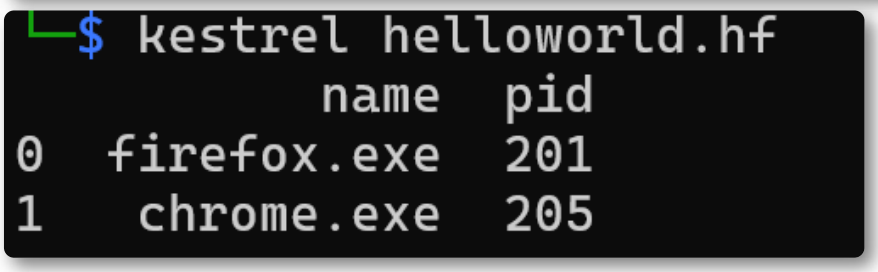
Kestrel threat hunting

Installation process

```
┌──$ python3 -m venv huntingspace

┌──(jonafk555㉿jonafk)-[~]
└─$ . huntingspace/bin/activate

┌──(huntingspace)(jonafk555㉿jonafk)-[~]
└─$ pip install --upgrade pip setuptools wheel
Requirement already satisfied: pip in ./huntingspace/lib/python3.13/site-packages (25.2)
Collecting pip
  Using cached pip-25.3-py3-none-any.whl.metadata (4.7 kB)
Collecting setuptools
  Using cached setuptools-80.9.0-py3-none-any.whl.metadata (6.6 kB)
Collecting wheel
  Downloading wheel-0.45.1-py3-none-any.whl.metadata (2.3 kB)
Using cached pip-25.3-py3-none-any.whl (1.8 MB)
Using cached setuptools-80.9.0-py3-none-any.whl (1.2 MB)
Downloading wheel-0.45.1-py3-none-any.whl (72 kB)
Installing collected packages: wheel, setuptools, pip
  Attempting uninstall: pip
    Found existing installation: pip 25.2
    Uninstalling pip-25.2:
      Successfully uninstalled pip-25.2
Successfully installed pip-25.3 setuptools-80.9.0 wheel-0.45.1
```

run the Hello World hunt flow.

```
# 建立測試資料
proclist = NEW process [
  {"name": "cmd.exe", "pid": "123"},
  {"name": "explorer.exe", "pid": "99"},
  {"name": "firefox.exe", "pid": "201"},
  {"name": "chrome.exe",  "pid": "205"}
]

# 在變數上做過濾（ASSIGN 形式）
browsers = proclist WHERE name IN ('firefox.exe', 'chrome.exe')

# 檢視
DISP browsers ATTR name, pid
~
~
~
~
~
~
~
~
~
~
~
~
"helloworld.hf" 13L, 354B
```



The logic of the hunt

I. Hunt Intent (Overall Objective)

The purpose of this script is to simulate the process of “filtering a list of system processes to extract browser processes.”
In other words, it is a simplified process-filtering hunt flow whose logic corresponds to real-world activity:

Locate processes related to specific applications (e.g., `chrome.exe`, `firefox.exe`) within event data and inspect their attributes.

II. Logical Step Breakdown

Step 1 — Create test data

```
proclist = NEW process [
  {"name": "cmd.exe", "pid": "123"},
  {"name": "explorer.exe", "pid": "99"},
  {"name": "firefox.exe", "pid": "201"},
```

```
  {"name": "chrome.exe",  "pid": "205"}
]
```

Logical function:

- `NEW process [...]` creates a simulated entity set of processes inside Kestrel.
- In a real scenario, this step is equivalent to reading `process`-type events from an EDR / SIEM, for example:

  ```
  GET process FROM stixshifter://ELASTIC WHERE [process:name = 'chrome.exe']
  ```

- The four JSON objects above represent four system processes, each containing two attributes:
  - `name` (executable name)
  - `pid` (process ID)

Significance:
This step builds the hunt's observation domain — the set from which you will search for suspicious or target processes.

Step 2 — Filter matching entities

```
browsers = proclist WHERE name IN ('firefox.exe', 'chrome.exe')
```

Logical function:

- Execute a filter on the existing variable `proclist`.
- Condition syntax: `WHERE name IN ('firefox.exe', 'chrome.exe')`
  - i.e., keep only entities whose `name` attribute matches one of those two values.
- The result is stored in a new variable `browsers`.

Real threat-hunting correspondence:

- If the data comes from a real SIEM, this step is a filter such as:

  ```
  [process:name IN ('firefox.exe', 'chrome.exe')]
  ```

- Purpose: focus on a class of potentially relevant application behavior, for example:
    - adversaries disguising activity as browser processes (browser-based persistence)
    - investigating suspicious child processes or anomalous behavior spawned by browsers (browser injection / proxy abuse)

Step 3 — Display results

```
DISP browsers ATTR name, pid
```

Logical function:

- `DISP` (Display) prints specified attributes to the command line.
- It shows the `name` and `pid` for each entity in the `browsers` set.

Real threat-hunting correspondence:

- Practically, this step typically involves:
    - sending hunt results to a visualization (tables/graphs)
    - exporting results for further analysis in Python/pandas
    - or feeding results into `APPLY` analytics modules (e.g., `top_count`, `cluster`, `timeline`)