Breaking the bank(er)

Automated configuration data extraction for banking malware

James Wyke

Senior Threat Researcher

SOPHOS

SOPHOS

Agenda

Agenda

- Introduction
- What can we find out about banking malware?
- How do we get this information?
- Our solution
- Output and results
- Pitfalls
- Summing up

Introduction

Introduction

- High profile takedowns
  - Operation Tovar
  - Shylock
- Replacement families emerge
  - Dridex
  - Dyreza
  - Vawtrak
  - KINS
  - …

What to do?

- Arrest People ☺

- Find out as much as we can about the malware
  - Track campaigns
  - Detect active compromises
  - Prevent future compromises

What can we find out about banking malware?

Extractable Data

- Command and control addresses – all of them
- Cryptographic keys
- Campaign IDs
- Botnet names
- Build versions
- DGA seed values
- C2 server issued commands
- Downloaded configuration files
    - Web injects
    - Redirects
    - Modules + more

Uses of data

- Indicators of compromise
  - Host based – file names, registry key names etc.
  - Network based – addresses, URL patterns etc.
- Reveal targets
  - Targeted URLS
  - Applications
  - Industry sectors
  - Countries
- Reveal actions
  - Commands executed
  - Data stolen

SOPHOS

How do we get this data?

Extracting data - manual

- RE
- Debug, disassemble, dump, decode. Repeat
- Slow the first time, still slow every time after that
- Requires analyst time. Every time

Extracting data - automation

- Scalable
- Repeatable
- Human input only required initially

Automation – how?

- Crawler
  - Needs feeding
  - Only getting downloaded files

- Embed in Sandbox system
  - Piggy back on existing infrastructure
  - Constant stream of samples
  - Convenient per-sample results
  - Well positioned for downstream systems

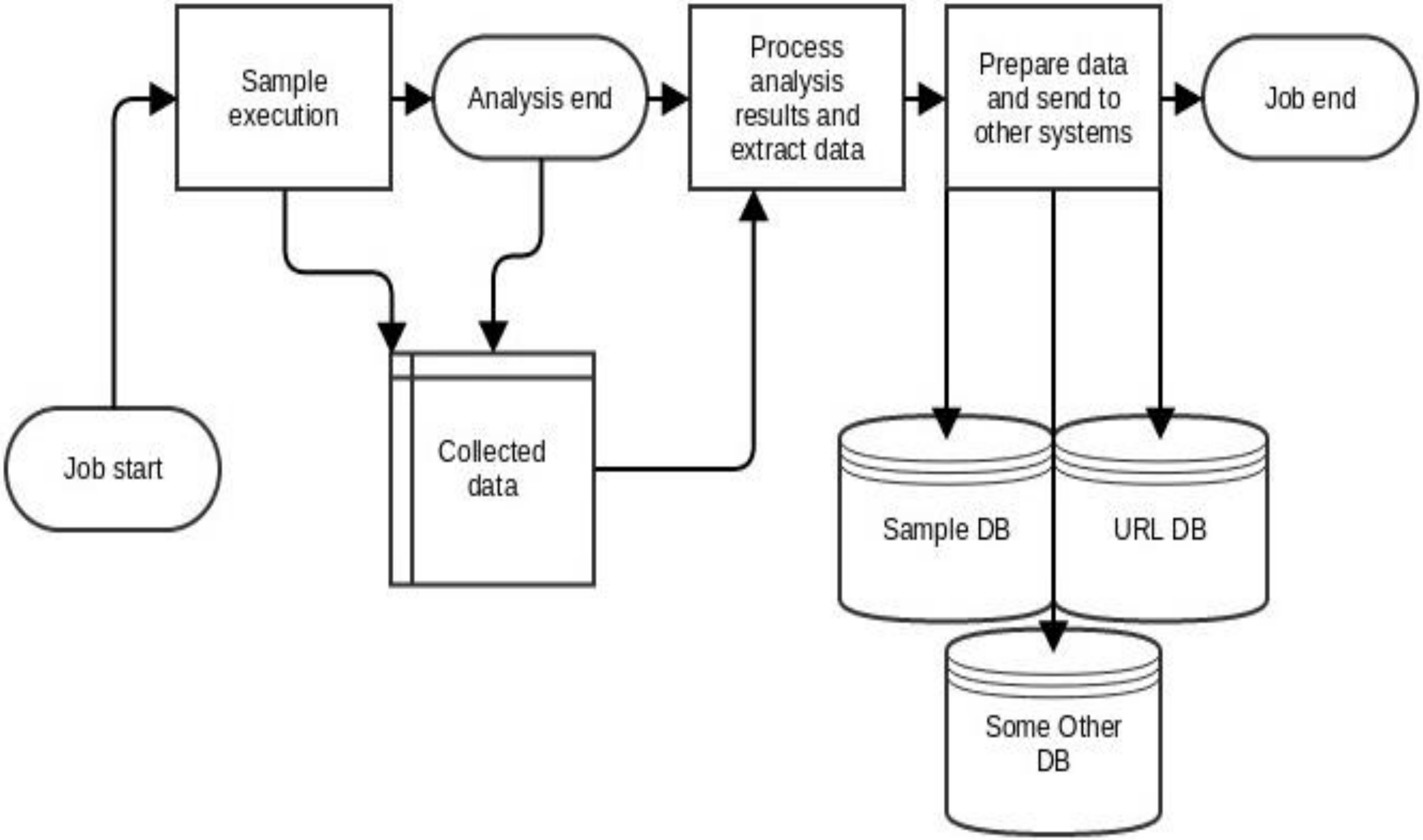
Our solution

Architecture

- Based on Cuckoo

- Additions include:
  - Enhanced data capture
  - Custom processing modules
  - Custom reporting modules

Architecture - overview



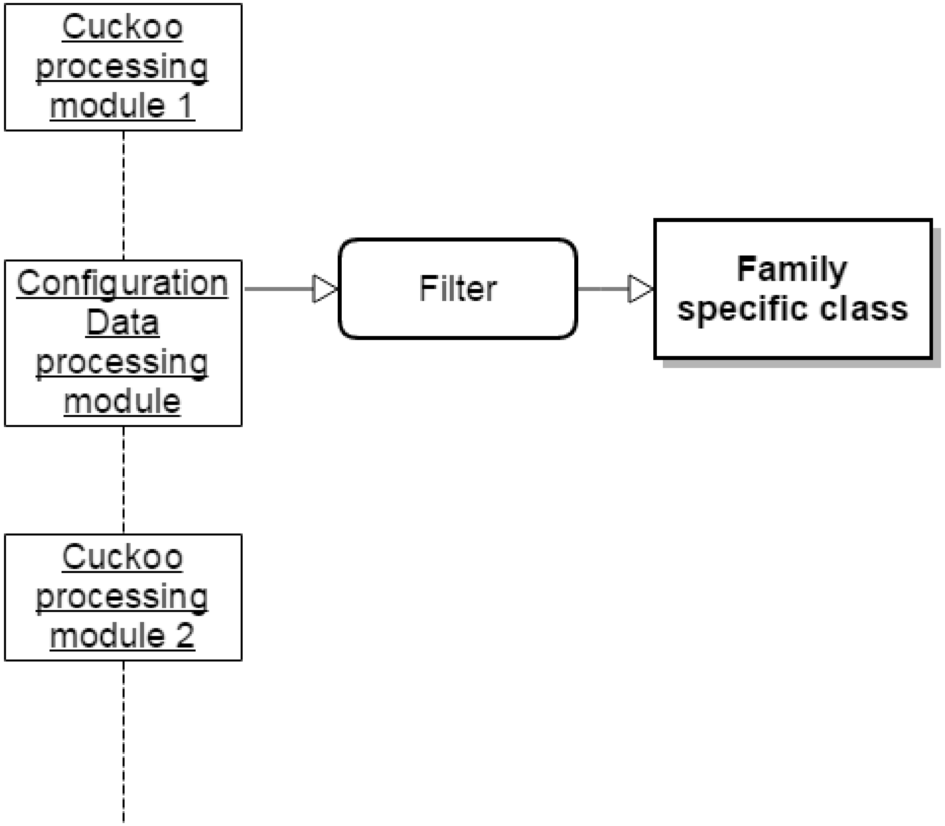
Key stages - capture

- Ensure raw material is available for later stages
- Memory
  - Full memory dumps
  - In-line memory dumps – ExitProcess, NtFreeVirtualMemory
- Network
  - Pcap
  - SSL/TLS MitM
- Disk
  - Files and registry
- Execution conditions
  - Analysis packages

Key stages - processing

- *ConfigurationData* processing module
- Filter incoming data – identify family
- Instantiate appropriate class for malware family

Key stages - processing


Cuckoo processing module 1
Configuration Data processing module
Cuckoo processing module 2
Filter
Family specific class

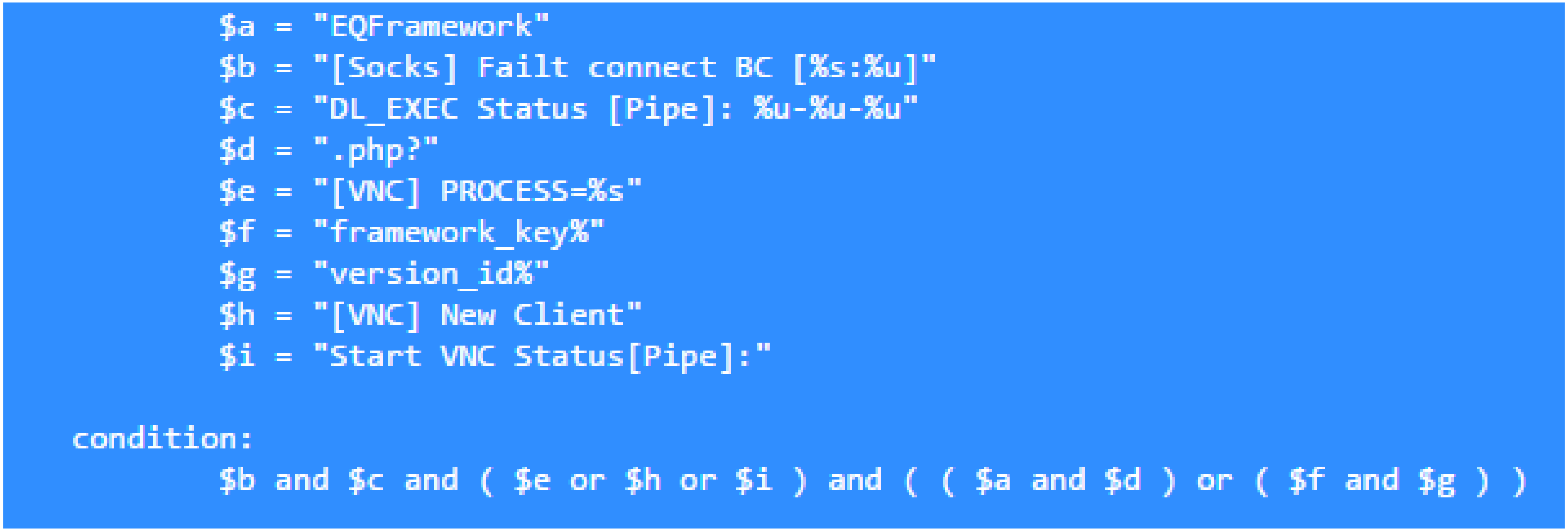
Key stages - processing

- For each family only implement the family specific Python class
- Find data structures from memory, extract, decode
- Decode files, registry items, network traffic
- Store in results dictionary, pass to *Reporting* modules

Processing - example

- Vawtrak aka NeverQuest

- Filter stage – Yara sig against memory

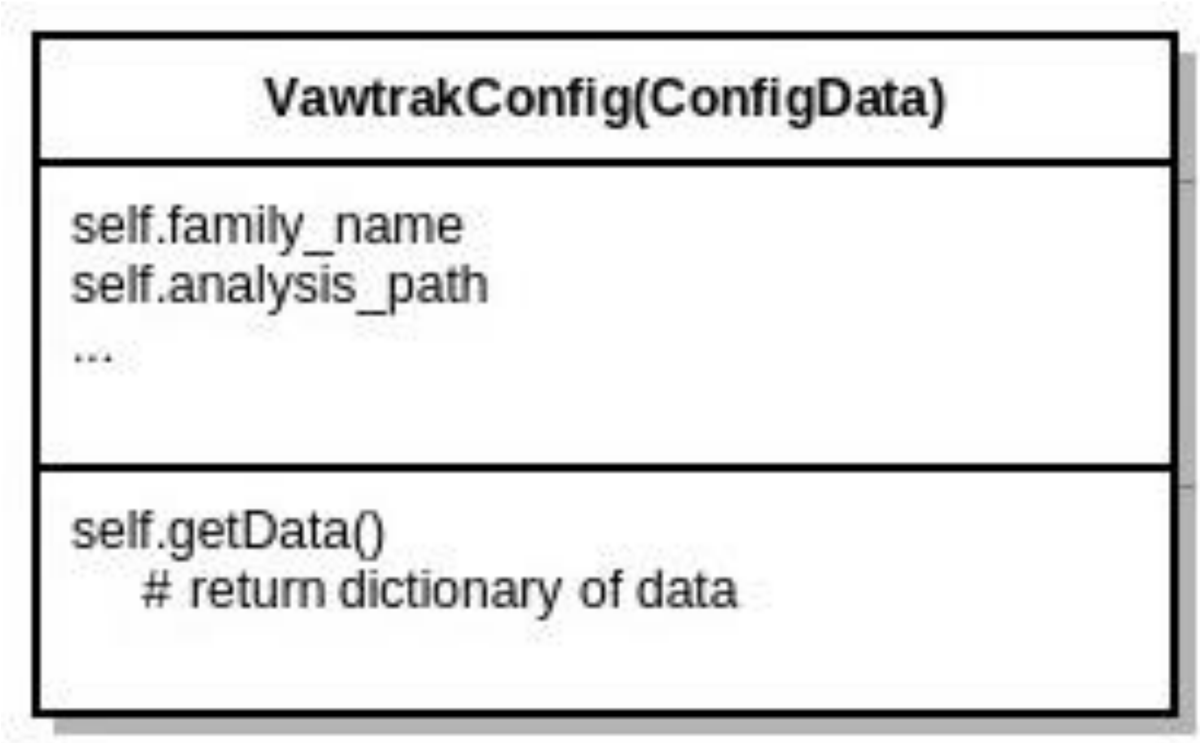


```
        $a = "EQFramework"
        $b = "[Socks] Failt connect BC [%s:%u]"
        $c = "DL_EXEC Status [Pipe]: %u-%u-%u"
        $d = ".php?"
        $e = "[VNC] PROCESS=%s"
        $f = "framework_key%"
        $g = "version_id%"
        $h = "[VNC] New Client"
        $i = "Start VNC Status[Pipe]:"

condition:
        $b and $c and ( $e or $h or $i ) and ( ( $a and $d ) or ( $f and $g ) )
```

Vawtrak processing

- *VawtrakConfig()*



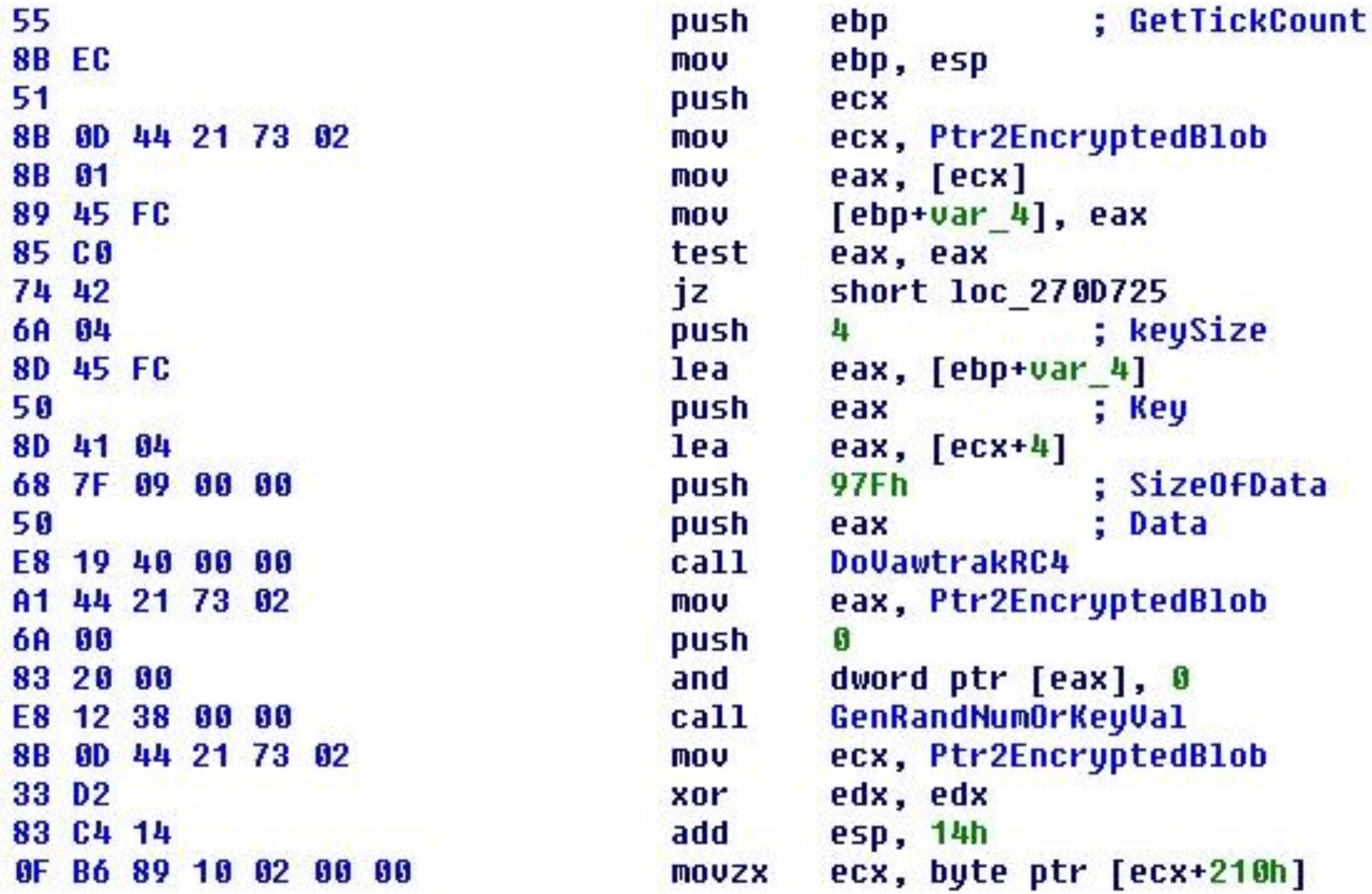
VawtrakConfig(ConfigData)

self.family_name
self.analysis_path
...

self.getData()
    # return dictionary of data

Vawtrak processing

- *get_c2_addresses()*



```
55                      push    ebp             ; GetTickCount
8B EC                   mov     ebp, esp
51                      push    ecx
8B 0D 44 21 73 02       mov     ecx, Ptr2EncryptedBlob
8B 01                   mov     eax, [ecx]
89 45 FC                mov     [ebp+var_4], eax
85 C0                   test    eax, eax
74 42                   jz      short loc_270D725
6A 04                   push    4               ; keySize
8D 45 FC                lea     eax, [ebp+var_4]
50                      push    eax             ; Key
8D 41 04                lea     eax, [ecx+4]
68 7F 09 00 00          push    97Fh            ; SizeOfData
50                      push    eax             ; Data
E8 19 40 00 00          call    DoVawtrakRC4
A1 44 21 73 02          mov     eax, Ptr2EncryptedBlob
6A 00                   push    0
83 20 00                and     dword ptr [eax], 0
E8 12 38 00 00          call    GenRandNumOrKeyVal
8B 0D 44 21 73 02       mov     ecx, Ptr2EncryptedBlob
33 D2                   xor     edx, edx
83 C4 14                add     esp, 14h
0F B6 89 10 02 00 00    movzx   ecx, byte ptr [ecx+210h]
```

Vawtrak processing

- *get_downloaded_config()*
- *get_received_commands()*
- *get_decoded_modules()*
- Convert pcap to har file for easier processing
- All gathered data returned as a Python dictionary

SOPHOS

Output and results

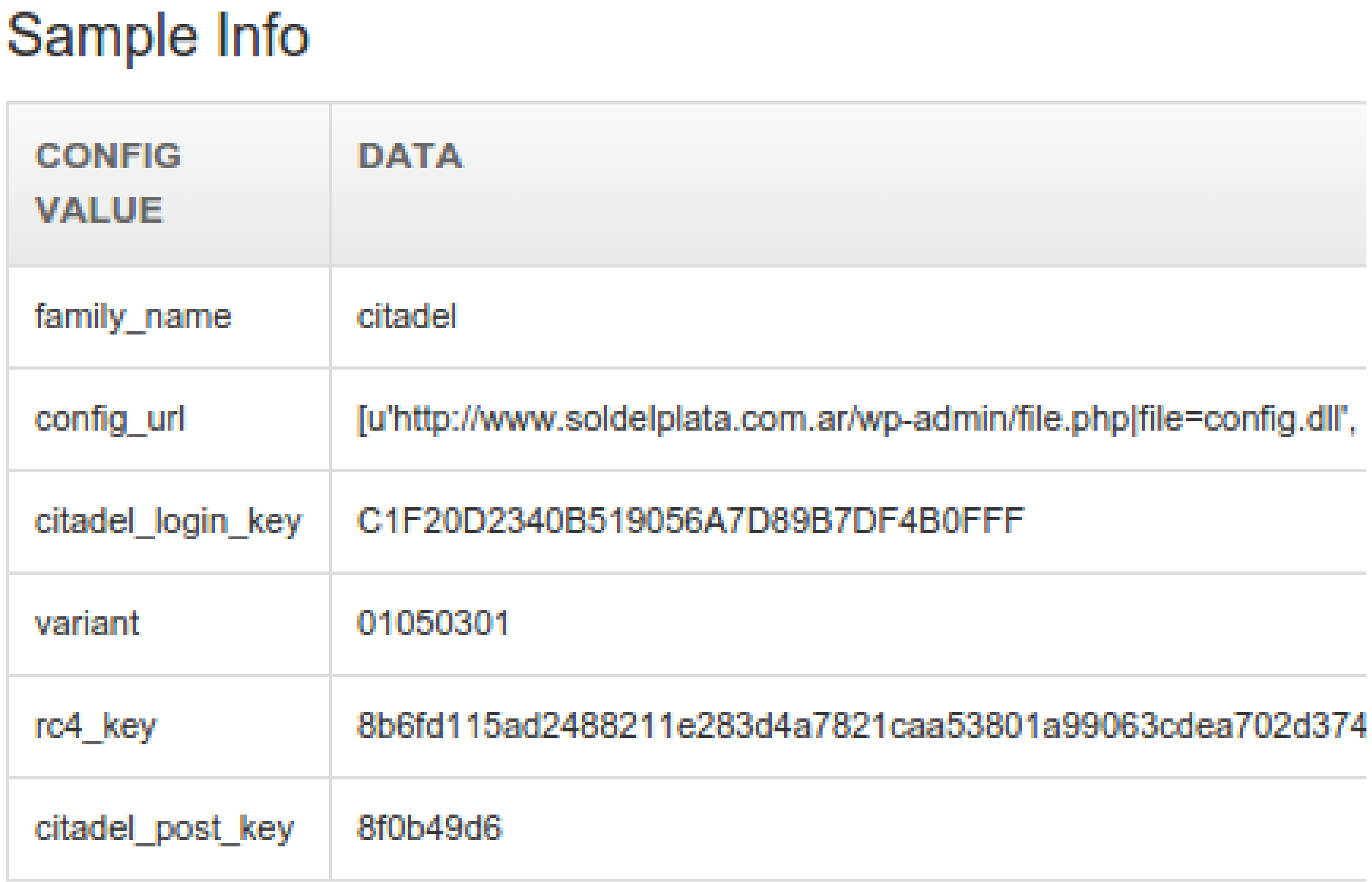
Reporting modules

- Take collected data, present it to other systems

- Web based report for humans

- Other modules take sections of the data and pass them on
  - URL’s/domains/IP’s to network reputation system
  - Decoded PE files to sample processing
  - Web injects/downloaded configs to dynamic config system

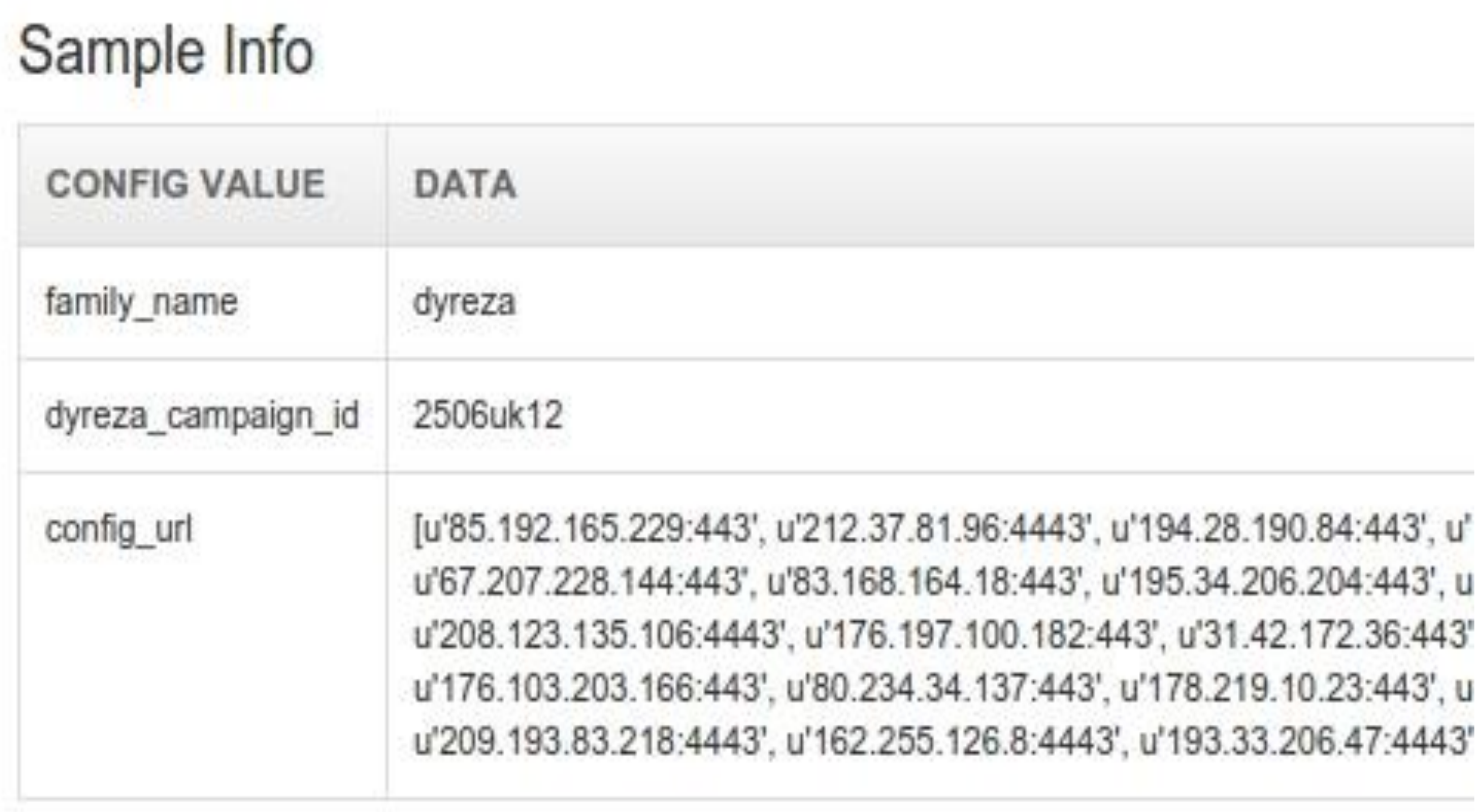
Output examples - Citadel



Sample Info

CONFIG VALUE	DATA
family_name	citadel
config_url	[u'http://www.soldelplata.com.ar/wp-admin/file.php|file=config.dll',
citadel_login_key	C1F20D2340B519056A7D89B7DF4B0FFF
variant	01050301
rc4_key	8b6fd115ad2488211e283d4a7821caa53801a99063cdea702d374
citadel_post_key	8f0b49d6

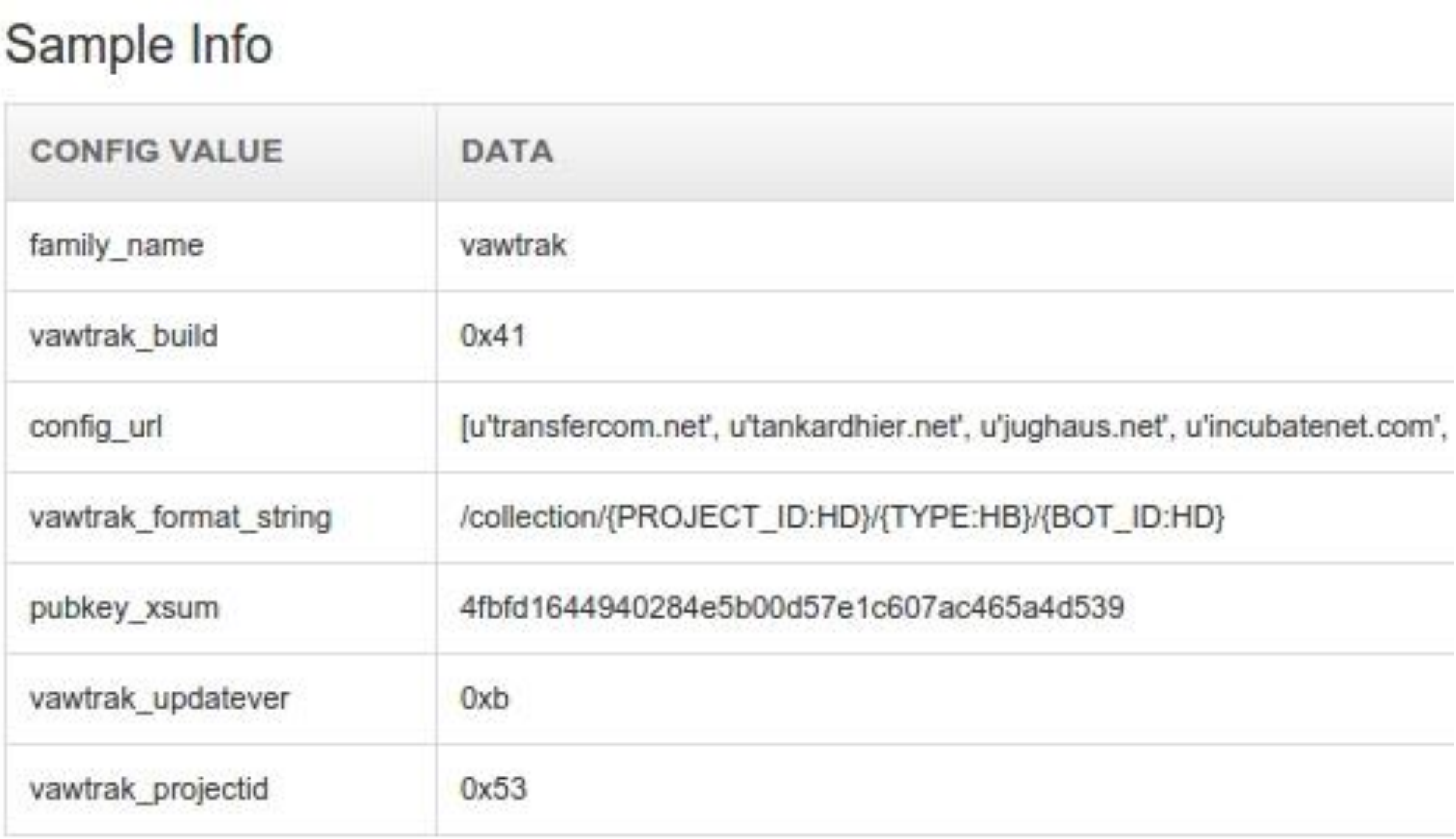
Output examples - Dyreza



Sample Info

CONFIG VALUE	DATA
family_name	dyreza
dyreza_campaign_id	2506uk12
config_url	[u'85.192.165.229:443', u'212.37.81.96:4443', u'194.28.190.84:443', u' u'67.207.228.144:443', u'83.168.164.18:443', u'195.34.206.204:443', u u'208.123.135.106:4443', u'176.197.100.182:443', u'31.42.172.36:443' u'176.103.203.166:443', u'80.234.34.137:443', u'178.219.10.23:443', u u'209.193.83.218:4443', u'162.255.126.8:4443', u'193.33.206.47:4443'

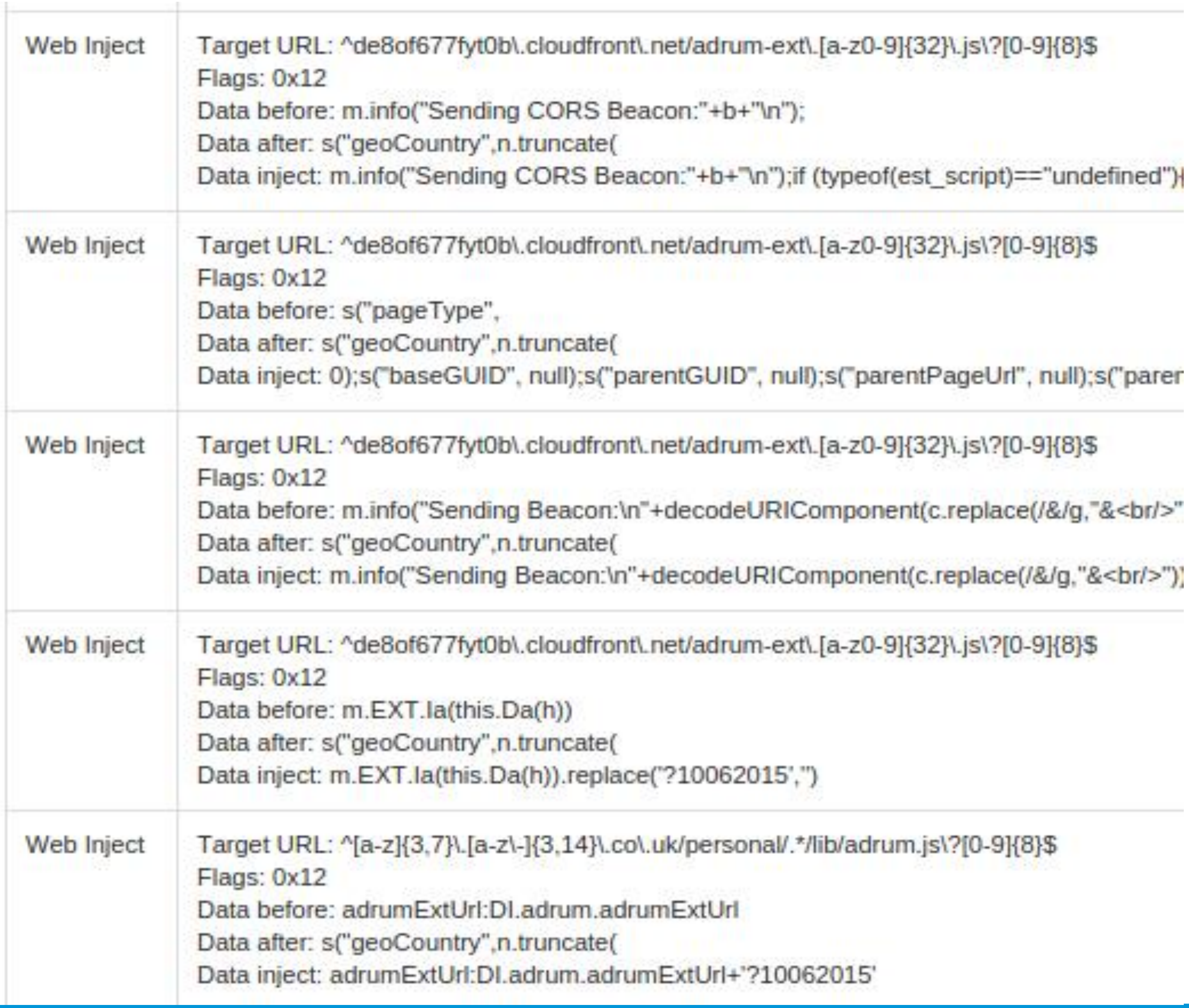
Output examples - Vawtrak



Sample Info

CONFIG VALUE	DATA
family_name	vawtrak
vawtrak_build	0x41
config_url	[u'transfercom.net', u'tankardhier.net', u'jughaus.net', u'incubatenet.com',
vawtrak_format_string	/collection/{PROJECT_ID:HD}/{TYPE:HB}/{BOT_ID:HD}
pubkey_xsum	4fbfd1644940284e5b00d57e1c607ac465a4d539
vawtrak_updatever	0xb
vawtrak_projectid	0x53

Output examples – web injects



Web Inject	Target URL: ^de8of677fyt0b\.cloudfront\.net/adrum-ext\.[a-z0-9]{32}\.js\?[0-9]{8}$ Flags: 0x12 Data before: m.info("Sending CORS Beacon:"+b+"\n"); Data after: s("geoCountry",n.truncate(Data inject: m.info("Sending CORS Beacon:"+b+"\n");if (typeof(est_script)=="undefined")
Web Inject	Target URL: ^de8of677fyt0b\.cloudfront\.net/adrum-ext\.[a-z0-9]{32}\.js\?[0-9]{8}$ Flags: 0x12 Data before: s("pageType", Data after: s("geoCountry",n.truncate(Data inject: 0);s("baseGUID", null);s("parentGUID", null);s("parentPageUrl", null);s("parer
Web Inject	Target URL: ^de8of677fyt0b\.cloudfront\.net/adrum-ext\.[a-z0-9]{32}\.js\?[0-9]{8}$ Flags: 0x12 Data before: m.info("Sending Beacon:\n"+decodeURIComponent(c.replace(/&/g,"&<br/>" Data after: s("geoCountry",n.truncate(Data inject: m.info("Sending Beacon:\n"+decodeURIComponent(c.replace(/&/g,"&<br/>"))
Web Inject	Target URL: ^de8of677fyt0b\.cloudfront\.net/adrum-ext\.[a-z0-9]{32}\.js\?[0-9]{8}$ Flags: 0x12 Data before: m.EXT.la(this.Da(h)) Data after: s("geoCountry",n.truncate(Data inject: m.EXT.la(this.Da(h)).replace('?10062015','')
Web Inject	Target URL: ^[a-z]{3,7}\.[a-z\-]{3,14}\.co\.uk/personal/.*/lib/adrum.js\?[0-9]{8}$ Flags: 0x12 Data before: adrumExtUrl:DI.adrum.adrumExtUrl Data after: s("geoCountry",n.truncate(Data inject: adrumExtUrl:DI.adrum.adrumExtUrl+'?10062015'

Extending

- New filter – Yara signature

- New class - *SomeOtherFamilyConfig()*

- Plenty of time to focus on the reverse engineering part

Pitfalls

Pitfalls

- Execution conditions
  - Browser requests
  - Analysis length
  - Cuckoo analysis packages
- Geo-targeting
  - Different files delivered based on country of victim
  - Multiple exit points
- Platform coverage
  - x64 modules delivered only under x64

SOPHOS

Summing up

Summing up

- Automated config data extraction
- Human readable and machine readable
- Easily extendable
- Scales as well as the underlying Sandbox system does
- Actively being used to protect against and track current banking malware botnets

SOPHOS