End-to-end Botnet Monitoring… Botconf 2019

Kevin O’Reilly & Keith Jarvis

Counter Threat Unit (CTU)
Research Team

Secureworks®
Cybersecurity Technologies. Services. Solutions.

…With Automated Config Extraction and Emulated Network Participation

Secureworks® Cybersecurity Technologies. Services. Solutions.

Agenda

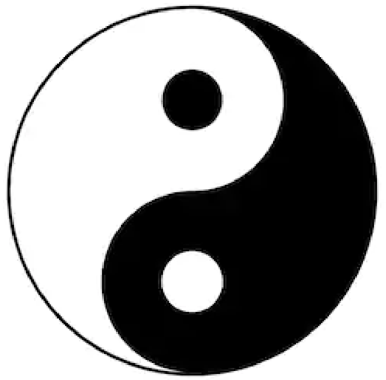
What will we be discussing today?

- **Two distinct angles:**
  - **Sandbox**
  - **Emulator**

More than the sum of their parts?

Emulator – Sandbox Synergy

- Emulator inputs:
  - C2 domains/IP addresses
  - RSA public keys
  - Version numbers
  - …
- Sandbox inputs:
  - Fresh samples!



Bots in the Sandbox

Secureworks®
Cybersecurity Technologies. Services. Solutions.

Sandbox

Essential Capabilities

- Automated Unpacking
- Configuration Decoding/Parsing

CAPE Sandbox

“**C**onfig **A**nd **P**ayload **E**xtraction”

- Open Source Project - began 2015
- Derived from spender-sandbox
  - Itself derived from Cuckoo Sandbox (v1.3) in 2014
- Overlap with Cuckoo today minimal
- https://github.com/kevoreilly/CAPE
- Community version: https://capesandbox.com

Parallels With Manual Approach

- Dynamic analysis
  - Victim machine/malware lab
  - API monitor
  - Debugger
  - Dumper
  - Import reconstructor

- Static analysis
  - Disassembler for unpacked payload
  - YARA for detection
  - Decoder or parser for configuration

The Sharpest Tool in the Sandbox?

The Debugger

- Powerful tool allowing instruction-level control
- Processor (hardware) breakpoints
    - 4 breakpoints on read/write/execute
    - Single-step mode
        - Instruction traces
- Trigger Actions:
    - Manipulate register/flag values
    - Dump payload or configuration
    - Set/clear further breakpoints
- Set initial breakpoints via Yara signatures or API hooks

Debugger-in-a-DLL

- In-process debugger
  - Within monitor DLL

- Processor (hardware) breakpoints
  - Debug registers Dr0 – Dr7
  - 4 breakpoints (per-thread)
  - EXCEPTION_SINGLE_STEP

- RtlDispatchException hook

- SetThreadContext API
  - NtGet/SetContextThread hook protection

Debugger demo

QakBot Instruction Trace & Anti-VM Bypass

Family Packages

- A whole package devoted to one malware family
    - Handle specific behaviours

- Have to have seen family before

- Future versions of family may break package

Behavioural Packages

- Capture payloads for a given behaviour:
  - Injection into other processes
  - Extraction of code
  - Decompression of code

- Not dependent on signatures or having seen malware before

- Captured payloads can then be detected by Yara signatures

- If the extracted/injected payloads contain configuration data:
  - Yara signature triggers configuration parser

‘Compression’ Package

- Simplest package

- Captures PE payload decompressed by RtlDecompressBuffer function:

```
HOOKDEF(NTSTATUS, WINAPI, RtlDecompressBuffer, CompressionFormat, UncompressedBuffer, UncompressedBufferSize, CompressedBuffer,
CompressedBufferSize, FinalUncompressedSize) {
    NTSTATUS ret = Old_RtlDecompressBuffer(CompressionFormat, UncompressedBuffer, UncompressedBufferSize,
                     CompressedBuffer, CompressedBufferSize, FinalUncompressedSize);

    if (NT_SUCCESS(ret)) {
        DoOutputDebugString("RtlDecompressBuffer hook: scanning region 0x%x size 0x%x for PE image(s).\n", UncompressedBuffer,
           *FinalUncompressedSize);
           DumpPEsInRange(UncompressedBuffer, *FinalUncompressedSize);
    }

    return ret;
}
```

‘Injection’ Package

- Captures payloads injected into other processes

- Uses API hooks
  - Track newly created processes and threads
  - Injected directly
    - WriteProcessMemory, NtWriteVirtualMemory, etc
  - Process hollowing
    - NtMapViewOfSection
  - Transacted hollowing

- Dumps prior to execution
  - NtResumeProcess/NtResumeThread

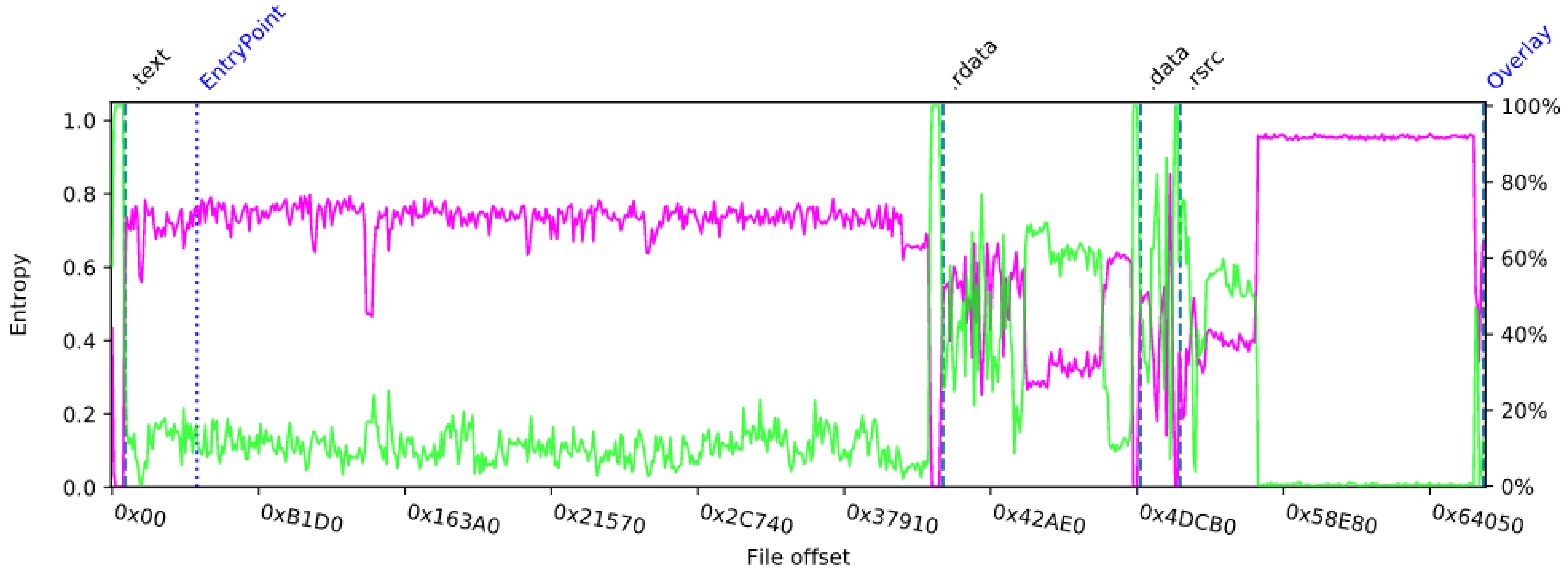
‘Extraction’ Package

- Captures payloads ‘extracted’ inside processes

- Tracks executable memory regions
  - Newly allocated
  - New executable permissions

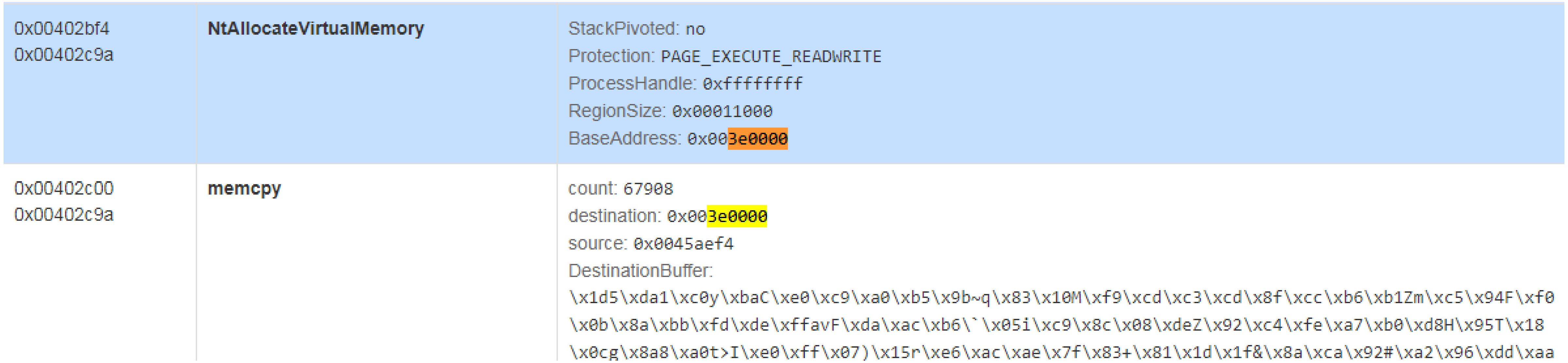
- Uses debugger breakpoints
  - Write breakpoints
    - Capture payloads just after they have been written
  - Execution breakpoints
    - Capture payloads before they are executed

Emotet



.text
EntryPoint
.rdata
.data
.rsrc
Overlay
Entropy
1.0
0.8
0.6
0.4
0.2
0.0
100%
80%
60%
40%
20%
0%
0x00
0xB1D0
0x163A0
0x21570
0x2C740
0x37910
0x42AE0
0x4DCB0
0x58E80
0x64050
File offset

Emotet Automated Unpacking



0x00402bf4 0x00402c9a	**NtAllocateVirtualMemory**	StackPivoted: no Protection: PAGE_EXECUTE_READWRITE ProcessHandle: 0xffffffff RegionSize: 0x00011000 BaseAddress: 0x003e0000
0x00402c00 0x00402c9a	**memcpy**	count: 67908 destination: 0x003e0000 source: 0x0045aef4 DestinationBuffer: \x1d5\xda1\xc0y\xbaC\xe0\xc9\xa0\xb5\x9b~q\x83\x10M\xf9\xcd\xc3\xcd\x8f\xcc\xb6\xb1Zm\xc5\x94F\xf0 \x0b\x8a\xbb\xfd\xde\xffavF\xda\xac\xb6\`\x05i\xc9\x8c\x08\xdeZ\x92\xc4\xfe\xa7\xb0\xd8H\x95T\x18 \x0cg\x8a8\xa0t>I\xe0\xff\x07)\x15r\xe6\xac\xae\x7f\x83+\x81\x1d\x1f&\x8a\xca\x92#\xa2\x96\xdd\xaa

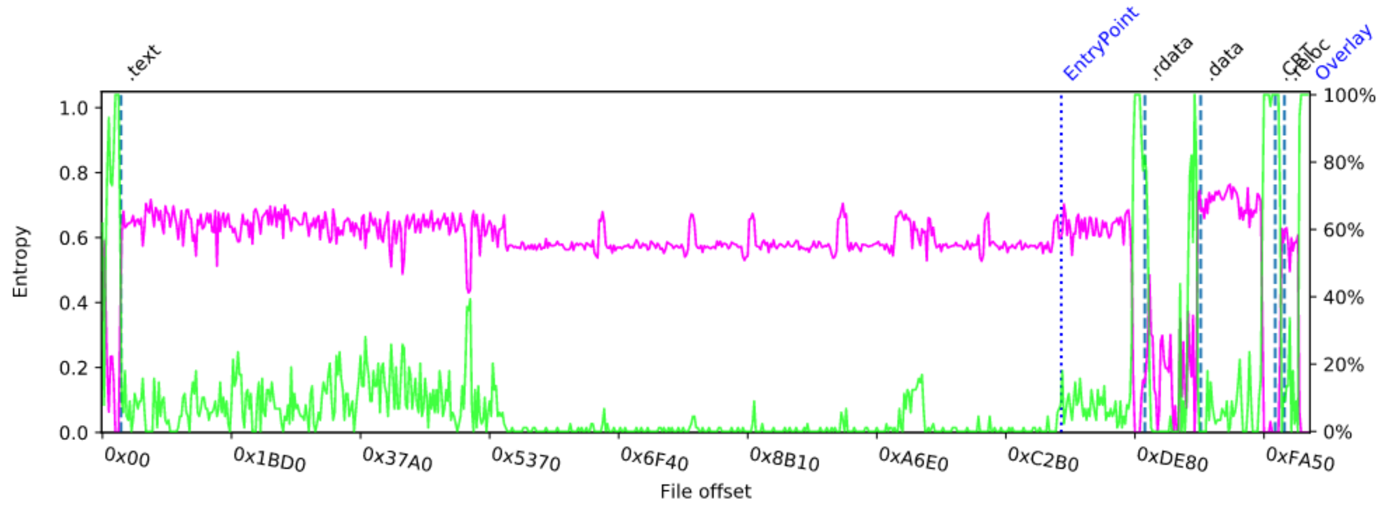
```
AllocationHandler: Adding allocation to tracked region list: 0x003E0000, size: 0x11000.
ActivateBreakpoints: TrackedRegion->AllocationBase: 0x003E0000, TrackedRegion->RegionSize: 0x11000, thread 2664
SetDebugRegister: Setting breakpoint 0 hThread=0xdc, Size=0x2, Address=0x003E0000 and Type=0x1.
SetThreadBreakpoint: Set bp 0 thread id 2664 type 1 at address 0x003E0000, size 2 with Callback 0x74af7510.
ActivateBreakpoints: Set write breakpoint on empty protect address: 0x003E0000
```

Emotet Automated Unpacking

```
CAPEExceptionFilter: breakpoint hit by instruction at 0x75FE9B60 (thread 2664)
BaseAddressWriteCallback: Breakpoint 0 at Address 0x003E0000.
ContextSetDebugRegister: Setting breakpoint 2 within Context, Size=0x0, Address=0x003E0000 and Type=0x0.
BaseAddressWriteCallback: byte written to 0x3e0000: 0x1d.
BaseAddressWriteCallback: Exec bp set on tracked region protect address.

CAPEExceptionFilter: breakpoint hit by instruction at 0x003E0000 (thread 2664)
ShellcodeExecCallback: Breakpoint 2 at Address 0x003E0000 - about to scan region for a PE image (base 0x003E0000, size 0x11000).
ScanForDisguisedPE: PE image located at: 0x3e053f
DumpImageInCurrentProcess: Attempting to dump 'raw' PE image.
DumpPE: PE file in memory dumped successfully
```

Type	Emotet Payload: 32-bit executable
Size	66560 bytes
Virtual Address	0x003E0000
Process	g8569c.exe
PID	2400

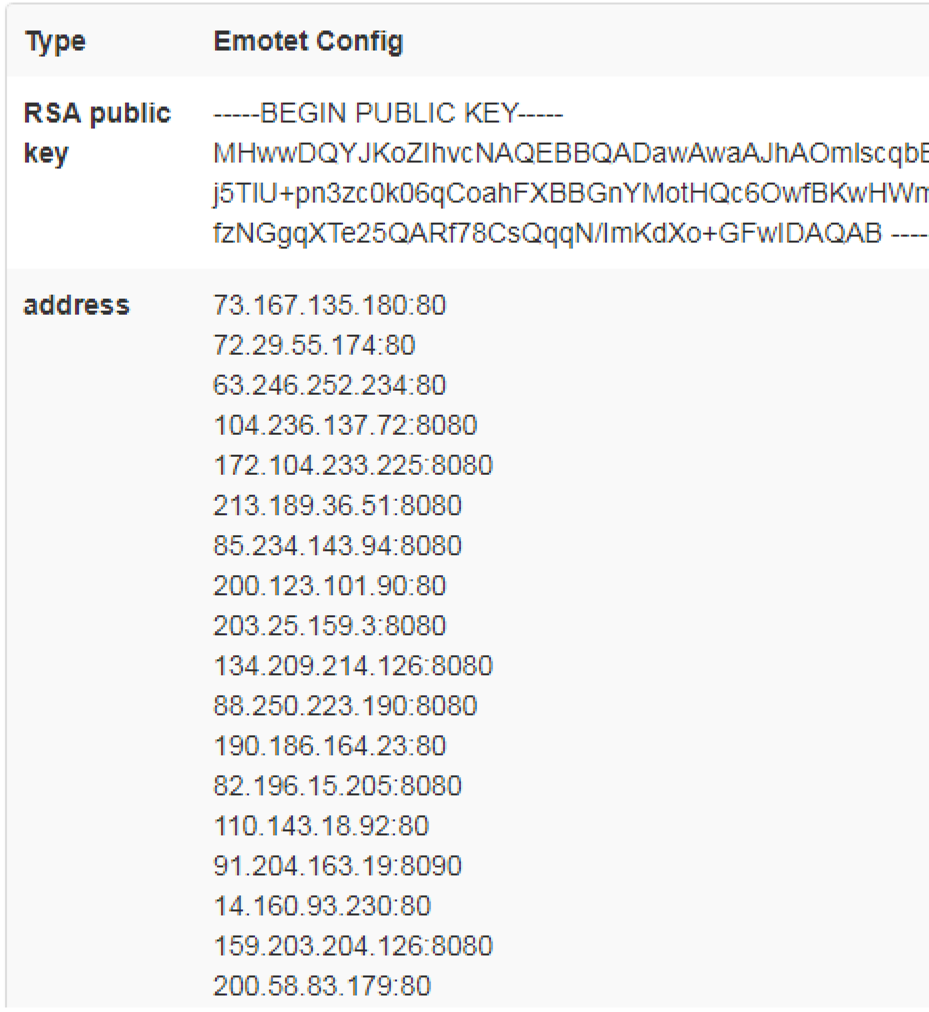


Secureworks

Emotet Config Extraction

```
rule Emotet
{
    meta:
        cape_type = "Emotet Payload"
    strings:
        $snippet = {33 C0 21 05 ?? ?? ?? ?? A3 ?? ?? ?? ?? 39 05 ?? ?? ?? ?? 74 18 40 A3
    condition:
        uint16(0) == 0x5A4D and ($snippet)
}

refc2list = yara_scan(filebuf, '$snippet')
c2list_va_offset = int(refc2list['$snippet'])
c2_list_va = struct.unpack('i', filebuf[c2list_va_offset+15:c2list_va_offset+19])[0]
c2_list_rva = c2_list_va - image_base
c2_list_offset = pe.get_offset_from_rva(c2_list_rva)
while 1:
    ip = struct.unpack('<I', filebuf[c2_list_offset:c2_list_offset+4])[0]
    c2_address = socket.inet_ntoa(struct.pack('!L', ip))
    port = str(struct.unpack('H', filebuf[c2_list_offset+4:c2_list_offset+6])[0])
    self.reporter.add_metadata('address', c2_address+':' + port)
    c2_list_offset += 8
```



Type	Emotet Config
RSA public key	-----BEGIN PUBLIC KEY----- MHwwDQYJKoZIhvcNAQEBBQADawAwaAJhAOmlscqb j5TIU+pn3zc0k06qCoahFXBBGnYMotHQc6OwfBKwHW fzNGgqXTe25QARf78CsQqqN/ImKdXo+GFwIDAQAB ----
address	73.167.135.180:80 72.29.55.174:80 63.246.252.234:80 104.236.137.72:8080 172.104.233.225:8080 213.189.36.51:8080 85.234.143.94:8080 200.123.101.90:80 203.25.159.3:8080 134.209.214.126:8080 88.250.223.190:8080 190.186.164.23:80 82.196.15.205:8080 110.143.18.92:80 91.204.163.19:8090 14.160.93.230:80 159.203.204.126:8080 200.58.83.179:80

Secureworks®

QakBot

- Extraction package extracts:
  - Main executable payload
  - DLL embedded in resources

```
rule QakBot
{
    meta:
        cape_type = "QakBot Payload"
    strings:
        $anti_vm = {8D 4D FC 51 E8 ?? ?? ?? ?? 83 C4 04 E8 ?? ?? ?? ?? 85 C0 7E 07 C7 45 F8 00 00 00 00 33 D2 74 02 EB FA 8B 45
F8 EB 08 33 C0 74 02 EB FA 33 C0 8B E5 5D C3}
        $decrypt_config = {8B 45 08 8B 88 24 04 00 00 51 8B 55 10 83 EA 14 52 8B 45 0C 83 C0 14 50 6A 14 8B 4D 0C 51 E8 6C 08 00
00}
    condition:
        uint16(0) == 0x5A4D and any of them
}
```

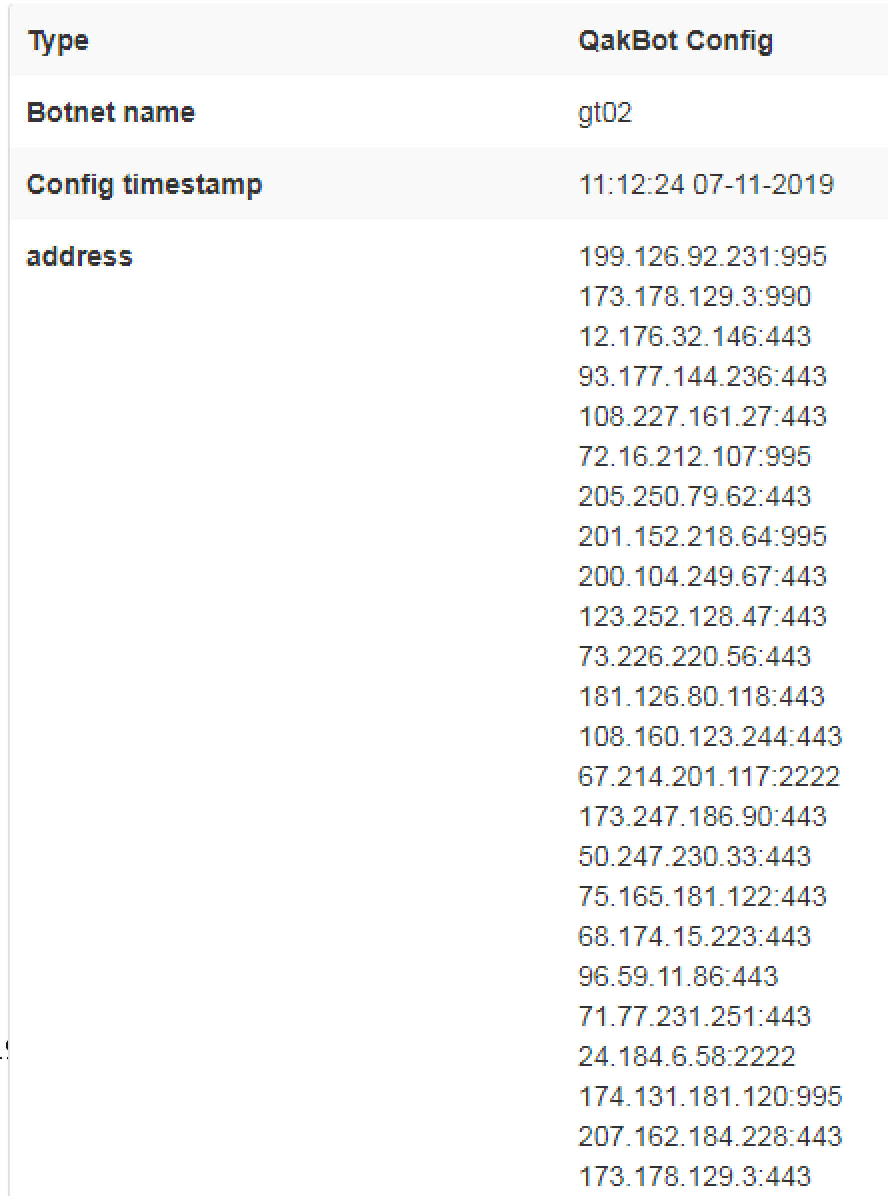
QakBot Config Extraction

- Dedicated family package triggered
  - Breakpoints:
    - Anti-VM bypass
    - Dump 2 x config region

```
DumpSize = (SIZE_T)*(DWORD*)((BYTE*)ExceptionInfo->ContextRecord->Esp+4*3);
DumpAddress = (PVOID)*(DWORD*)((BYTE*)ExceptionInfo->ContextRecord->Esp+4*4);

CAPEExceptionFilter: breakpoint hit by instruction at 0x004054AF
0x4054af (05) e86c080000              CALL 0x871
Trace: CALL detected, grabbing size 0x2b and buffer 0x3bcff10 from stack.

CAPEExceptionFilter: breakpoint hit by instruction at 0x004054B4
0x4054b4 (03) 83c414                  ADD ESP, 0x14
DumpMemory: CAPE output file successfully created: C:\iaDUZcSim\CAPE\2580_70260233749341
Added new CAPE file to list with path: C:\iaDUZcSim\CAPE\2580_70260233749341954122019
Trace: dumped QakBot config from 0x3bcff10.
```



Type	QakBot Config
Botnet name	gt02
Config timestamp	11:12:24 07-11-2019
address	199.126.92.231:995 173.178.129.3:990 12.176.32.146:443 93.177.144.236:443 108.227.161.27:443 72.16.212.107:995 205.250.79.62:443 201.152.218.64:995 200.104.249.67:443 123.252.128.47:443 73.226.220.56:443 181.126.80.118:443 108.160.123.244:443 67.214.201.117:2222 173.247.186.90:443 50.247.230.33:443 75.165.181.122:443 68.174.15.223:443 96.59.11.86:443 71.77.231.251:443 24.184.6.58:2222 174.131.181.120:995 207.162.184.228:443 173.178.129.3:443

The Future…

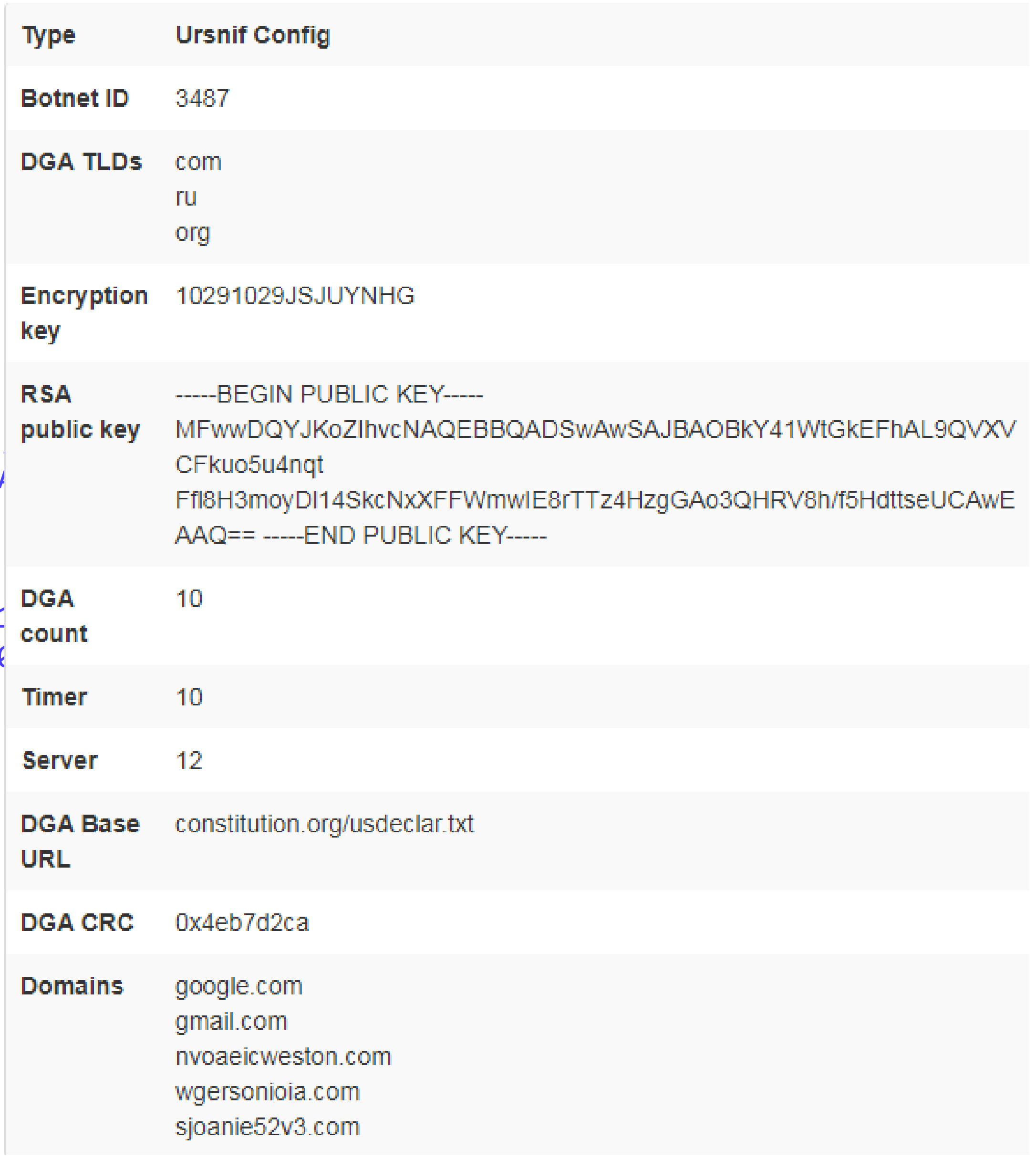
- CAPE v2 just released
  - Python 3
  - A huge thank you to Andriy Brukhovetskyy (@d00m3dr4v3n) – FireEye
  - https://github.com/kevoreilly/CAPEv2
    - (KVM hardended anti-anti-vm: https://github.com/doomedraven/Tools/blob/master/Virtualization/kvm-qemu.sh)

- Behavioural packages combined
  - Enabled by default
  - Reduce executions to maximum of 2

- Expanded Debugger options
  - Family-specific “package” all within YARA signature

Ursnif/ISFB

```
rule Ursnif3
{
    meta:
          cape_type = "Ursnif Payload"
          cape_options = "dll=Debugger.dll,step-out=$crypto32, dumpsize=eax,
dumptype0=0x24, base-on-api=RtlAddVectoredExceptionHandler, dump-on-api=RtlA
dump-on-api-type=0x25"
    strings:
        $crypto32 = {8B C3 83 EB 01 85 C0 75 0D 0F B6 16 83 C6 01 89 74 24 1
E0 01 03 D2 85 C0 0F 84 AB 01 00 00 8B C3 83 EB 01 85 C0 89 5C 24 20 75 13 0
00}
        $golden_ratio = {8B 70 EC 33 70 F8 33 70 08 33 30 83 C0 04 33 F1 81
08 41 81 F9 84 00 00 00}
          condition:
        uint16(0) == 0x5A4D and (all of them)
}
```



Type	Ursnif Config
Botnet ID	3487
DGA TLDs	com ru org
Encryption key	10291029JSJUYNHG
RSA public key	-----BEGIN PUBLIC KEY----- MFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBAOBkY41WtGkEFhAL9QVXV CFkuo5u4nqt Ffl8H3moyDI14SkcNxXFFWmwIE8rTTz4HzgGAo3QHRV8h/f5HdttseUCAwE AAQ== -----END PUBLIC KEY-----
DGA count	10
Timer	10
Server	12
DGA Base URL	constitution.org/usdeclar.txt
DGA CRC	0x4eb7d2ca
Domains	google.com gmail.com nvoaeicweston.com wgersonioia.com sjoanie52v3.com

Secureworks

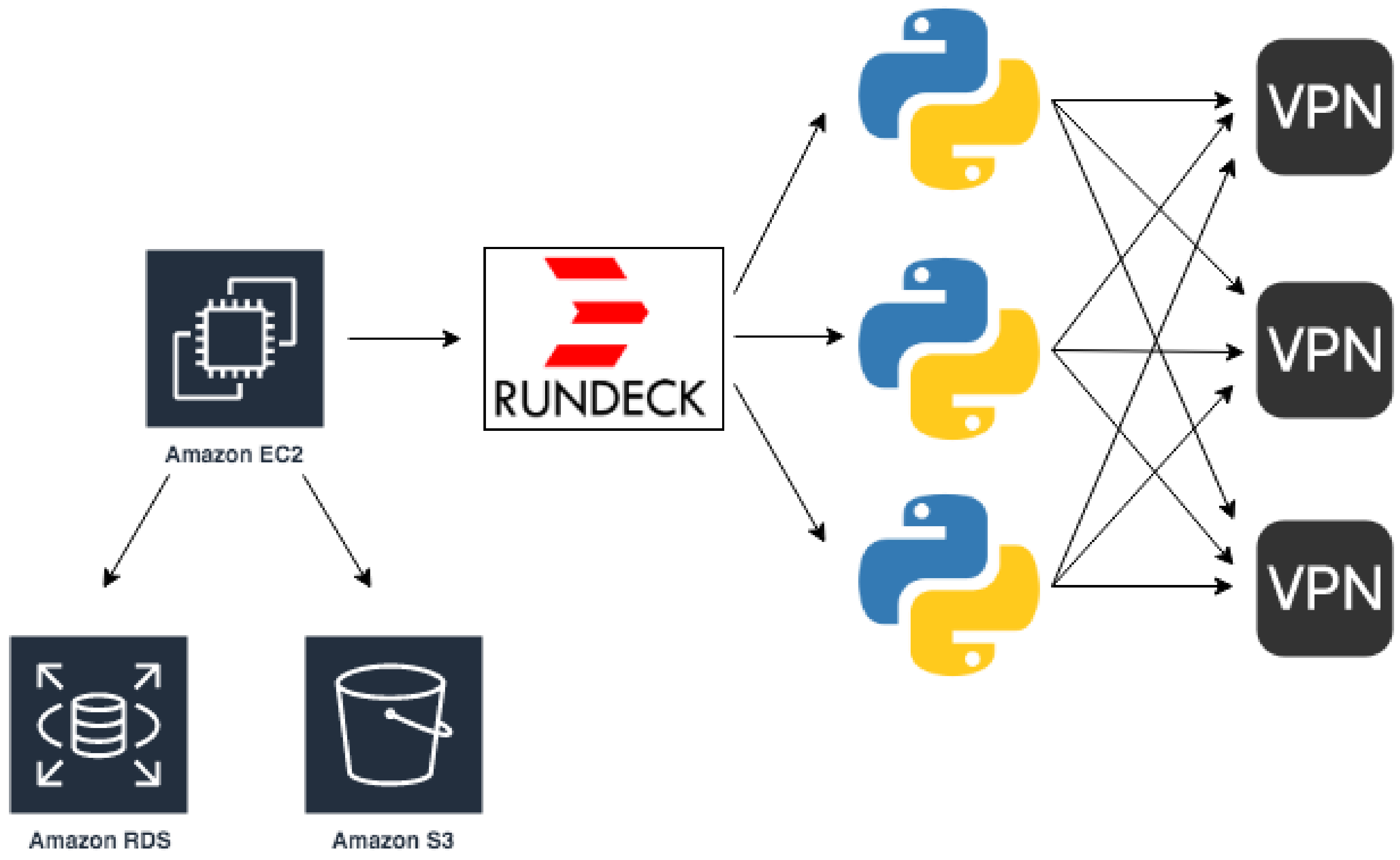
Secureworks®
Cybersecurity Technologies. Services. Solutions.

Botnet Emulator Framework

Why emulation?

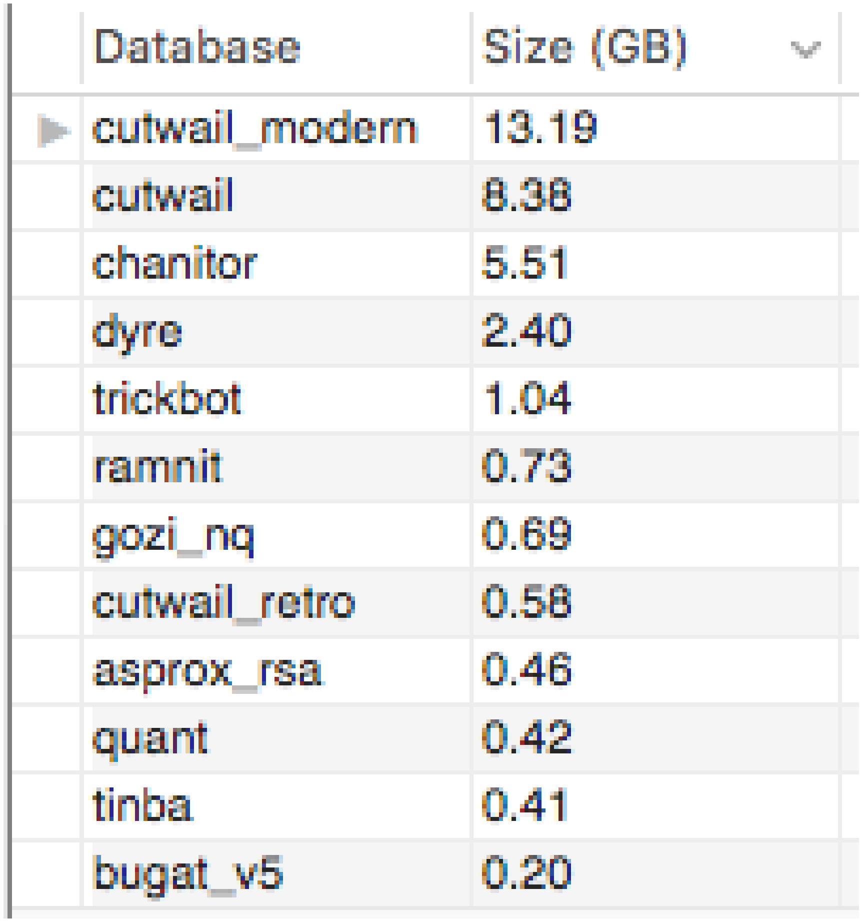
- Botnet participation allows retrieval of artefacts and context unavailable from samples
- Long-term tracking of targeting and operational details
- Vetting indicators with high-fidelity
- Synergy with config extraction for class of malware with features unavailable in ordinary sandbox detonation

Architecture



Data Retention

- Less data than you think to store years of botnet interactions
- Save everything! verbose logging, HTTP sessions, etc.
- Database is under 100 GB
- Local EBS volume is 100 GB
- S3 storage is 230 GB



Database	Size (GB)
cutwail_modern	13.19
cutwail	8.38
chanitor	5.51
dyre	2.40
trickbot	1.04
ramnit	0.73
gozi_nq	0.69
cutwail_retro	0.58
asprox_rsa	0.46
quant	0.42
tinba	0.41
bugat_v5	0.20

Cost

- EC2: $30/mo
- RDS: $60/mo
- S3: $5/mo
- EBS: $10/mo

- VPS: $30/mo
- VPN: $20/mo

$255

per month

OPSEC

- Make traffic and behaviour near replica of actual bot
- Generate plausible but contrived metadata (e.g., computer and domain names)
- Withstand competent non-state investigation
- Non-attributable infrastructure unless subject to subpoena
- Log timestamps and egress IP addresses for future correlation

Network Egress

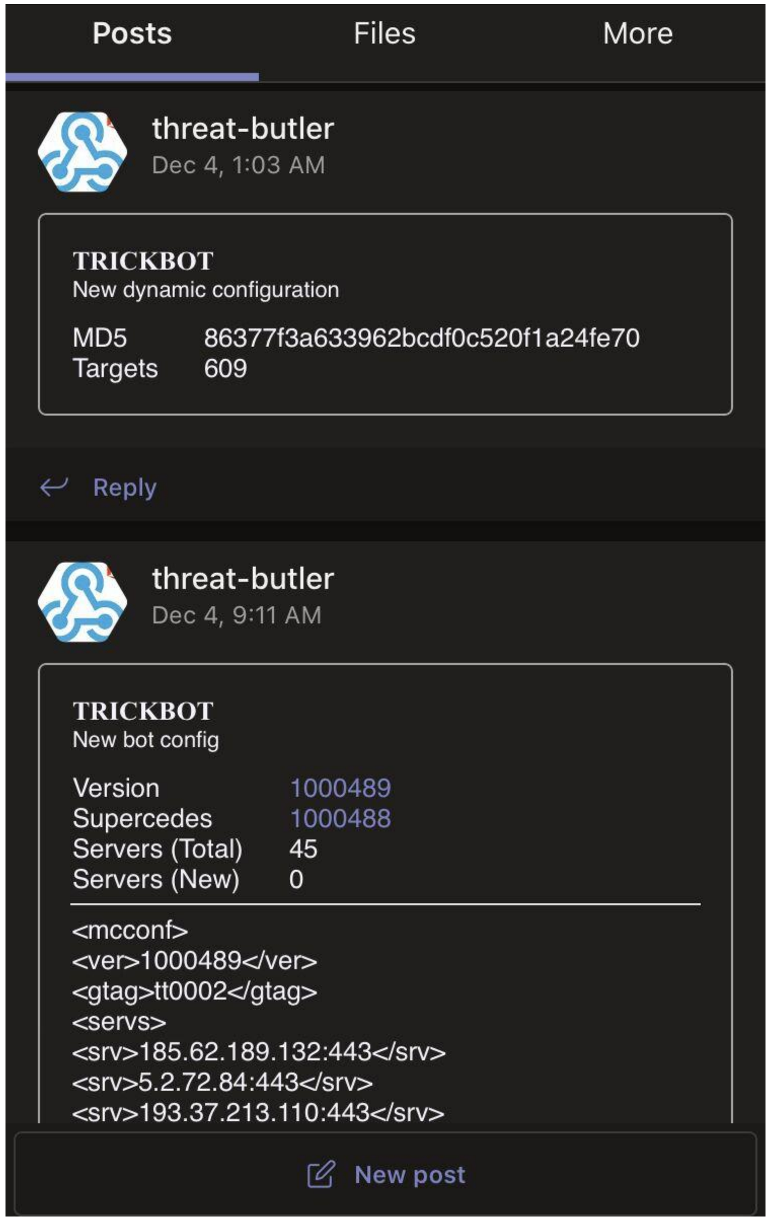
- Use a combination of Tor, VPS, and commercial VPNs
- Geographic diversity is great for empirically finding geofencing
- Simple round robin is largely suitable
- VPN microservice allows an emulator to request specific regions/countries and capabilities

Working with Commercial Providers

- Don’t (intentionally) trash someone else’s IP space
- VPNs WILL ban you (no refund)
- Beware the Internet do-gooder
- Few if any allow outbound TCP 25 (necessary for Cutwail)
- By product is good intel source of VPN exit nodes

Action on Data

- “ChatOps” notifications to analysts
- Email distribution lists (third-parties, LE, working groups)
- Intelligence platforms
- High fidelity blacklists



threat-butler
Dec 4, 1:03 AM

TRICKBOT
New dynamic configuration

MD5	86377f3a633962bcdf0c520f1a24fe70
Targets	609

threat-butler
Dec 4, 9:11 AM

TRICKBOT
New bot config

Version	1000489
Supercedes	1000488
Servers (Total)	45
Servers (New)	0

```
<mcconf>
<ver>1000489</ver>
<gtag>tt0002</gtag>
<servs>
<srv>185.62.189.132:443</srv>
<srv>5.2.72.84:443</srv>
<srv>193.37.213.110:443</srv>
```

Pitfalls: SOCKS Proxy

- OpenSSH SOCKS5 server is unstable during high throughput
- Use Dante, open source SOCKS5 server
- Protect with iptables and configuration-level filters and authentication

Pitfalls: Custom HTTP Library

- Reasons to not use Requests
  - Protocol violations ⇉ Uncommon in malware
  - Absent SOCKS support ⇉ Added in 2016
  - Custom header ordering ⇉ Use OrderedDict()
  - Provide IP address for request ⇉ Patch Requests
  - Access to endpoint SSL certificate ⇉ Patch Requests

Pitfalls: Neglect DNS Data

- Nature of system allows tracking of DNS resolutions over time
- Augment this data with passive DNS (Farsight)
- Useful for C2 with flaky DNS but stable hosting
- Need to be careful with C2 living in fast-flux systems

Pitfalls: Neglect SSL Certificates

- Save unique certificates as they appear and associate with IP
- Flowsynth to generate PCAP from X.509, check coverage
- Augment other data sets like Censys, Shodan, or SONAR

Pitfalls: Dealing with Sinkholes

- Regularly retrieve sinkhole list from abuse.ch SinkDB
- Detect common sinkhole-related HTTP features
- Avoiding sinkholes saves execution times and prevents poisoning other researchers' data
- BitSight won't track us down and try to sell us a report about our emulator being infected with dangerous malware

Pitfalls: Data Pruning

- Always prune your list of C2 servers
- General formula: has the C2 accepted connections in two weeks? Given a valid response in past month?

PITFALLS: “SKIN DEEP” EMULATION

- Register bot, interrogate C2, dispose of bot
- Instead, store registered bots and periodically reconnect them to create pool of long-lived “infections”
- Good way to get additional payloads to “mature” bots