SecureWorks®

The Information Security Experts

Inside the Storm: Protocols and Encryption of the Storm Botnet

Joe Stewart, GCIH
Director of Malware Research, SecureWorks

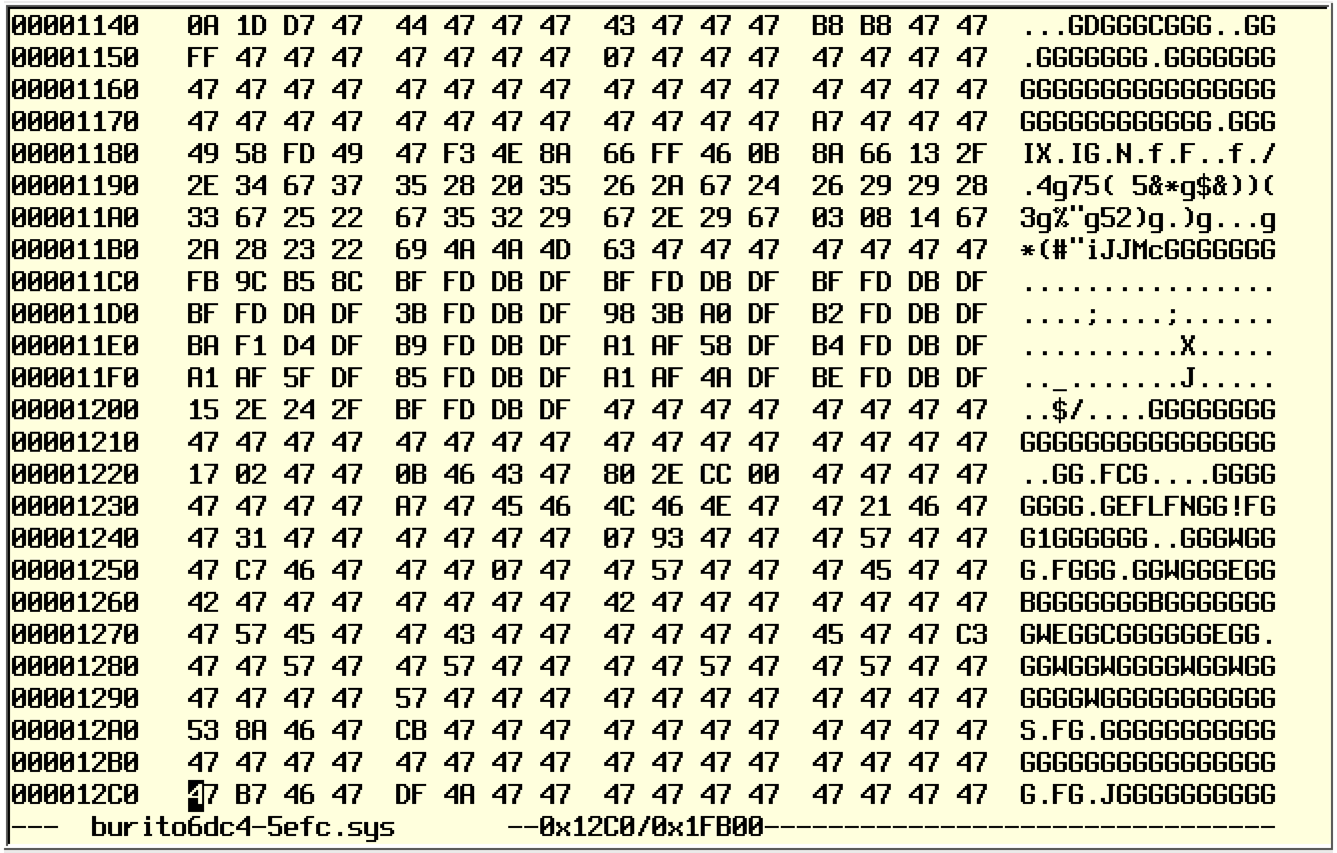
To be covered in this talk:

- Quick-and-dirty unpacking of Storm
- Structure of the Storm botnet
- Introduction to the Overnet protocol
- Storm's use of Overnet
- Encryption algorithms within Storm
- Storm node activation
- Storm node communication
- Notes on enumerating the Storm botnet

Unpacking Storm (the easy way)

- Why do we need to unpack at all?
  - Didn't you write Truman for behavioral analysis of stuff like this?
- Storm installation procedure
  - Dropper EXE installs rootkit
  - Rootkit injects userland bot code on load
- Userland code is simply XORed inside kernel driver binary
- Extraction:
  - Run in sandnet, grab dropped .sys file
  - Extract, XOR and run (no anti-VM/debugging checks!)

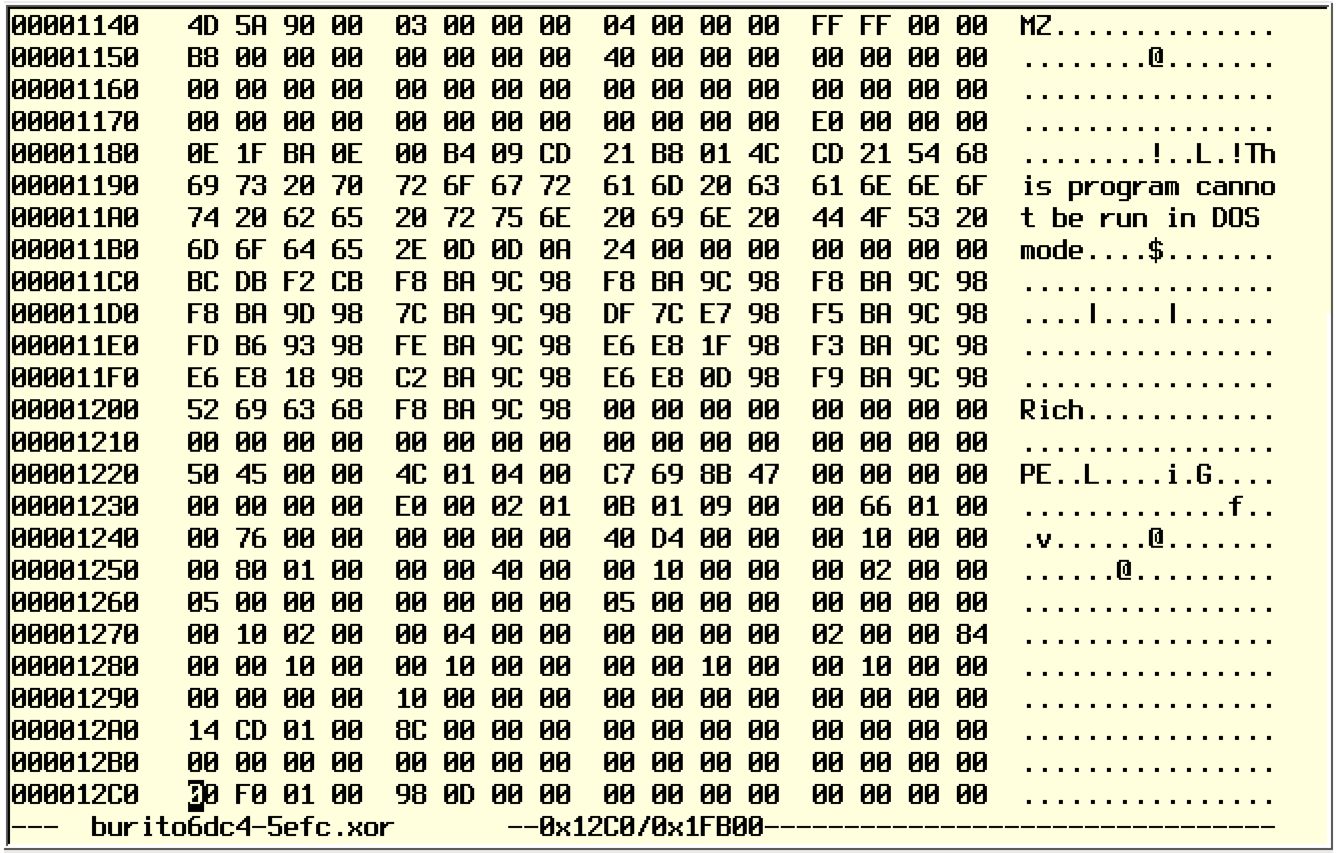
Locating the XORed EXE



```
00001140   0A 1D D7 47  44 47 47 47  43 47 47 47  B8 B8 47 47  ...GDGGGCGGG..GG
00001150   FF 47 47 47  47 47 47 47  07 47 47 47  47 47 47 47  .GGGGGGG.GGGGGGG
00001160   47 47 47 47  47 47 47 47  47 47 47 47  47 47 47 47  GGGGGGGGGGGGGGGG
00001170   47 47 47 47  47 47 47 47  47 47 47 47  A7 47 47 47  GGGGGGGGGGGG.GGG
00001180   49 58 FD 49  47 F3 4E 8A  66 FF 46 0B  8A 66 13 2F  IX.IG.N.f.F..f./
00001190   2E 34 67 37  35 28 20 35  26 2A 67 24  26 29 29 28  .4g75( 5&*g$&))(
000011A0   33 67 25 22  67 35 32 29  67 2E 29 67  03 08 14 67  3g%"g52)g.)g...g
000011B0   2A 28 23 22  69 4A 4A 4D  63 47 47 47  47 47 47 47  *(#"iJJMcGGGGGGG
000011C0   FB 9C B5 8C  BF FD DB DF  BF FD DB DF  BF FD DB DF  ................
000011D0   BF FD DA DF  3B FD DB DF  98 3B A0 DF  B2 FD DB DF  ....;....;......
000011E0   BA F1 D4 DF  B9 FD DB DF  A1 AF 58 DF  B4 FD DB DF  ..........X.....
000011F0   A1 AF 5F DF  85 FD DB DF  A1 AF 4A DF  BE FD DB DF  .._.......J.....
00001200   15 2E 24 2F  BF FD DB DF  47 47 47 47  47 47 47 47  ..$/....GGGGGGGG
00001210   47 47 47 47  47 47 47 47  47 47 47 47  47 47 47 47  GGGGGGGGGGGGGGGG
00001220   17 02 47 47  0B 46 43 47  80 2E CC 00  47 47 47 47  ..GG.FCG....GGGG
00001230   47 47 47 47  A7 47 45 46  4C 46 4E 47  47 21 46 47  GGGG.GEFLFNGG!FG
00001240   47 31 47 47  47 47 47 47  07 93 47 47  47 57 47 47  G1GGGGGG..GGGWGG
00001250   47 C7 46 47  47 47 07 47  47 57 47 47  47 45 47 47  G.FGGG.GGWGGGEGG
00001260   42 47 47 47  47 47 47 47  42 47 47 47  47 47 47 47  BGGGGGGGBGGGGGGG
00001270   47 57 45 47  47 43 47 47  47 47 47 47  45 47 47 C3  GWEGGCGGGGGGEGG.
00001280   47 47 57 47  47 57 47 47  47 47 57 47  47 57 47 47  GGWGGWGGGGWGGWGG
00001290   47 47 47 47  57 47 47 47  47 47 47 47  47 47 47 47  GGGGWGGGGGGGGGGG
000012A0   53 8A 46 47  CB 47 47 47  47 47 47 47  47 47 47 47  S.FG.GGGGGGGGGGG
000012B0   47 47 47 47  47 47 47 47  47 47 47 47  47 47 47 47  GGGGGGGGGGGGGGGG
000012C0   47 B7 46 47  DF 4A 47 47  47 47 47 47  47 47 47 47  G.FG.JGGGGGGGGGG
---  burito6dc4-5efc.sys       --0x12C0/0x1FB00--------------------------------
```

The Information Security Experts

Locating the XORed EXE

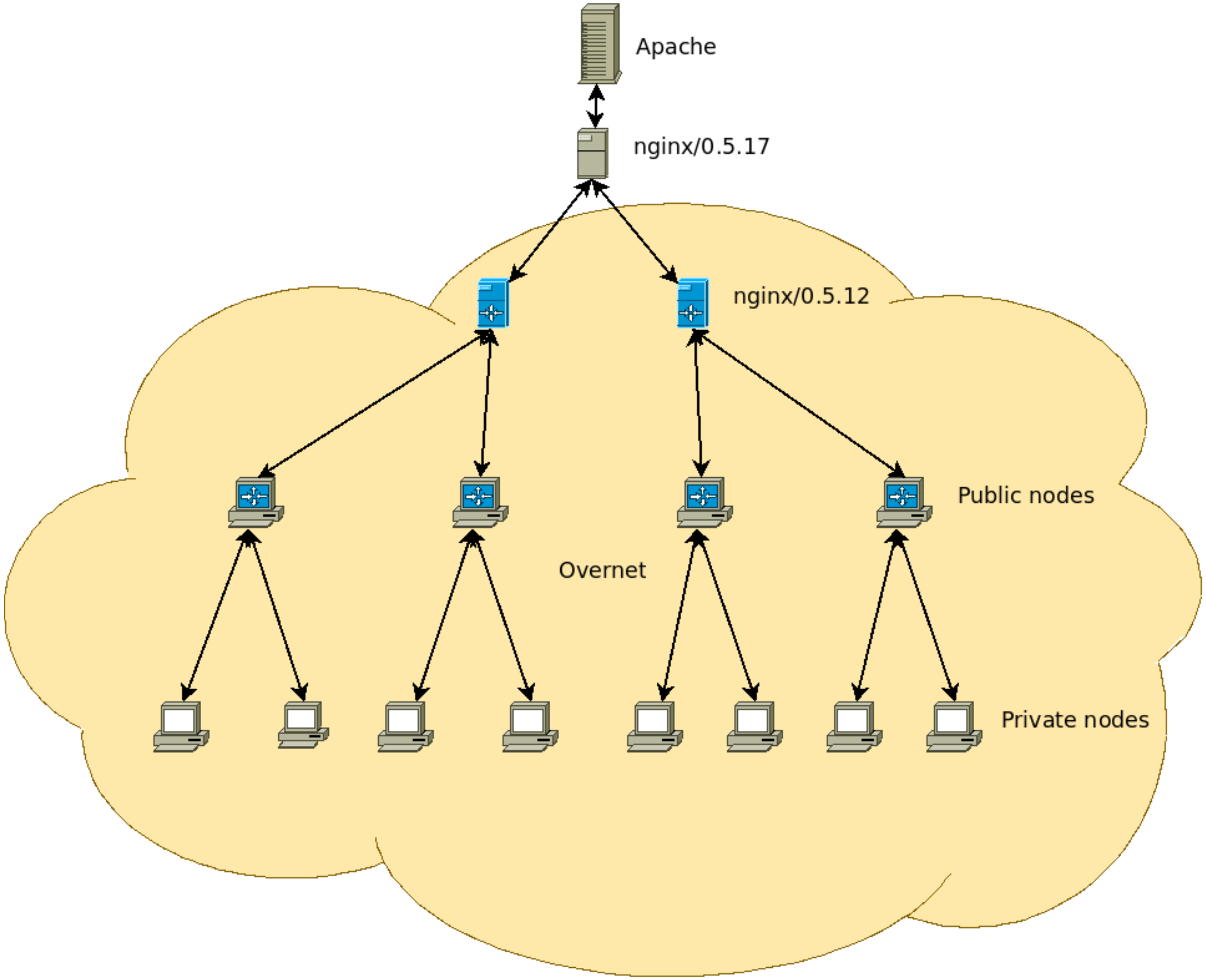


```
00001140   4D 5A 90 00  03 00 00 00  04 00 00 00  FF FF 00 00  MZ..............
00001150   B8 00 00 00  00 00 00 00  40 00 00 00  00 00 00 00  ........@.......
00001160   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
00001170   00 00 00 00  00 00 00 00  00 00 00 00  E0 00 00 00  ................
00001180   0E 1F BA 0E  00 B4 09 CD  21 B8 01 4C  CD 21 54 68  ........!..L.!Th
00001190   69 73 20 70  72 6F 67 72  61 6D 20 63  61 6E 6E 6F  is program canno
000011A0   74 20 62 65  20 72 75 6E  20 69 6E 20  44 4F 53 20  t be run in DOS
000011B0   6D 6F 64 65  2E 0D 0D 0A  24 00 00 00  00 00 00 00  mode....$.......
000011C0   BC DB F2 CB  F8 BA 9C 98  F8 BA 9C 98  F8 BA 9C 98  ................
000011D0   F8 BA 9D 98  7C BA 9C 98  DF 7C E7 98  F5 BA 9C 98  ....|....|......
000011E0   FD B6 93 98  FE BA 9C 98  E6 E8 1F 98  F3 BA 9C 98  ................
000011F0   E6 E8 18 98  C2 BA 9C 98  E6 E8 0D 98  F9 BA 9C 98  ................
00001200   52 69 63 68  F8 BA 9C 98  00 00 00 00  00 00 00 00  Rich............
00001210   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
00001220   50 45 00 00  4C 01 04 00  C7 69 8B 47  00 00 00 00  PE..L....i.G....
00001230   00 00 00 00  E0 00 02 01  0B 01 09 00  00 66 01 00  .............f..
00001240   00 76 00 00  00 00 00 00  40 D4 00 00  00 10 00 00  .v......@.......
00001250   00 80 01 00  00 00 40 00  00 10 00 00  00 02 00 00  ......@.........
00001260   05 00 00 00  00 00 00 00  05 00 00 00  00 00 00 00  ................
00001270   00 10 02 00  00 04 00 00  00 00 00 00  02 00 00 84  ................
00001280   00 00 10 00  00 10 00 00  00 00 10 00  00 10 00 00  ................
00001290   00 00 00 00  10 00 00 00  00 00 00 00  00 00 00 00  ................
000012A0   14 CD 01 00  8C 00 00 00  00 00 00 00  00 00 00 00  ................
000012B0   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
000012C0   00 F0 01 00  98 0D 00 00  00 00 00 00  00 00 00 00  ................
---  burito6dc4-5efc.xor      --0x12C0/0x1FB00--------------------------------
```

About Storm's “polymorphism”

- AV companies struggle to keep up with Storm variants
- The packing code changes every 10 minutes (controlled server-side)
- However, the last-stage payload (XORed code) changes much less frequently (on the order of months these days)

Storm Architecture

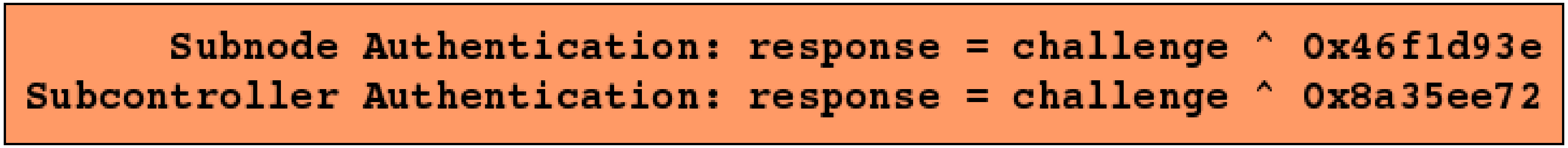


Storm Architecture

- Level 1: Apache C&C
- Level 2: Nginx 0.5.17 proxy
  - Hides top-level Apache server somewhere in the world
- Level 3: Nginx 0.5.12 proxies (subcontrollers)
  - Hide master Nginx host, listen in on Overnet traffic
- Level 4: Public nodes (supernodes)
  - Act as reverse HTTP proxies to controller, fast-flux nameservers
- Level 5: Private nodes (subnodes)
  - Send spam or DDoS packets

Storm encryption overview

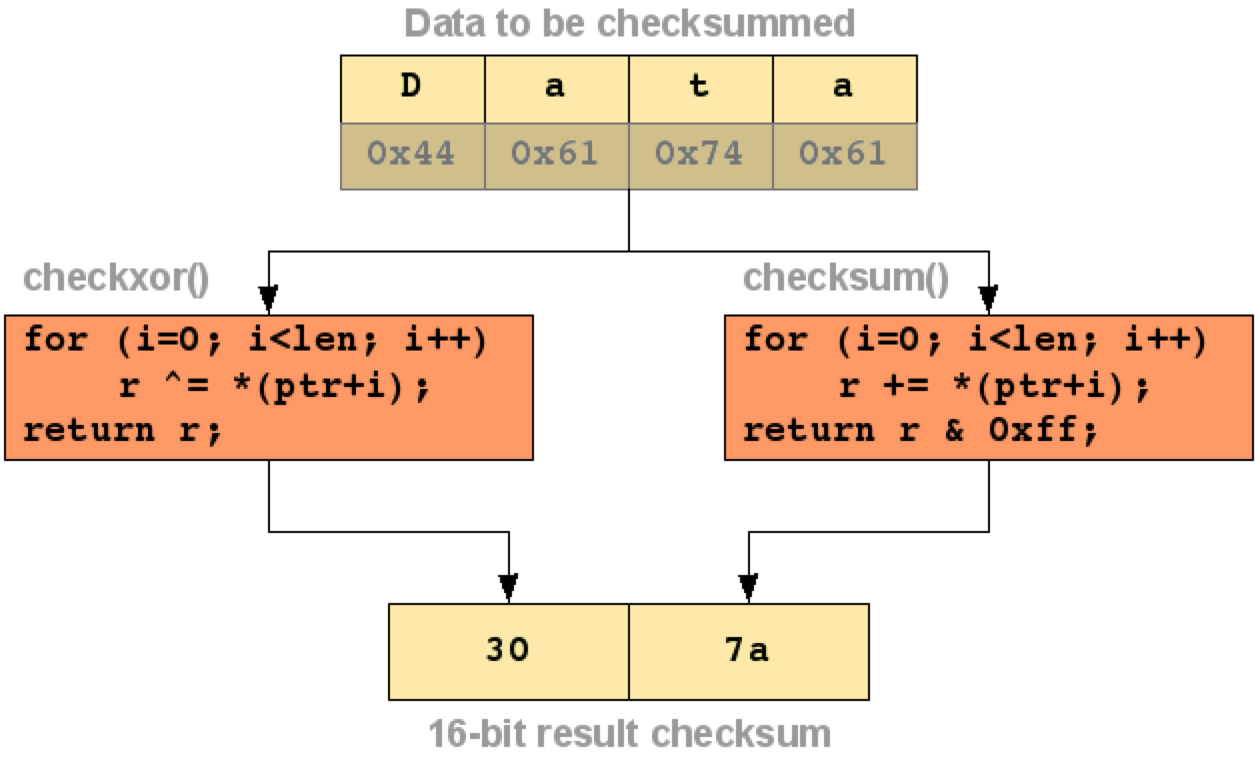
- RSA is used for encoding controller list packet
- Other TCP communication: base64 and zlib compression
- For Overnet, Storm uses two different custom hash permutation algorithms
- Authentication: 4-byte challenge/response



```
      Subnode Authentication: response = challenge ^ 0x46f1d93e
Subcontroller Authentication: response = challenge ^ 0x8a35ee72
```

Storm encryption overview 2

- Simple checksum algorithm used throughout Storm code:



Data to be checksummed

D	a	t	a
0x44	0x61	0x74	0x61

checkxor()

```
for (i=0; i<len; i++)
    r ^= *(ptr+i);
return r;
```

checksum()

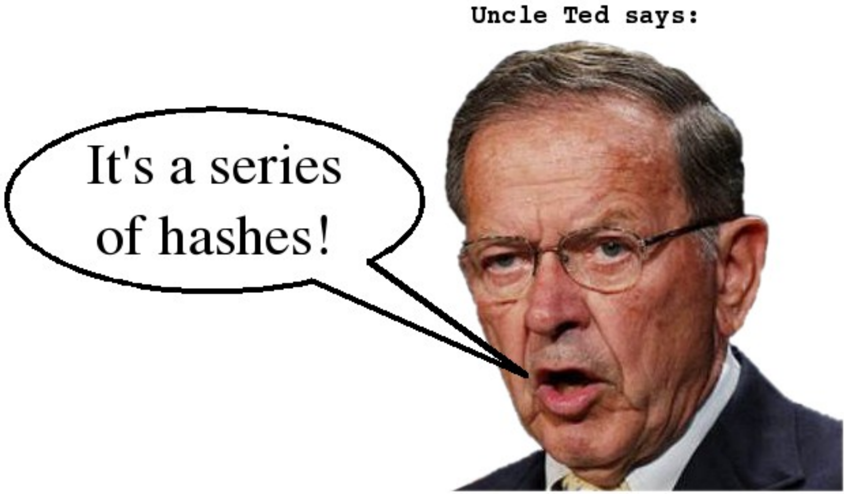
```
for (i=0; i<len; i++)
    r += *(ptr+i);
return r & 0xff;
```

30	7a

16-bit result checksum

Overnet/Edonkey protocol overview

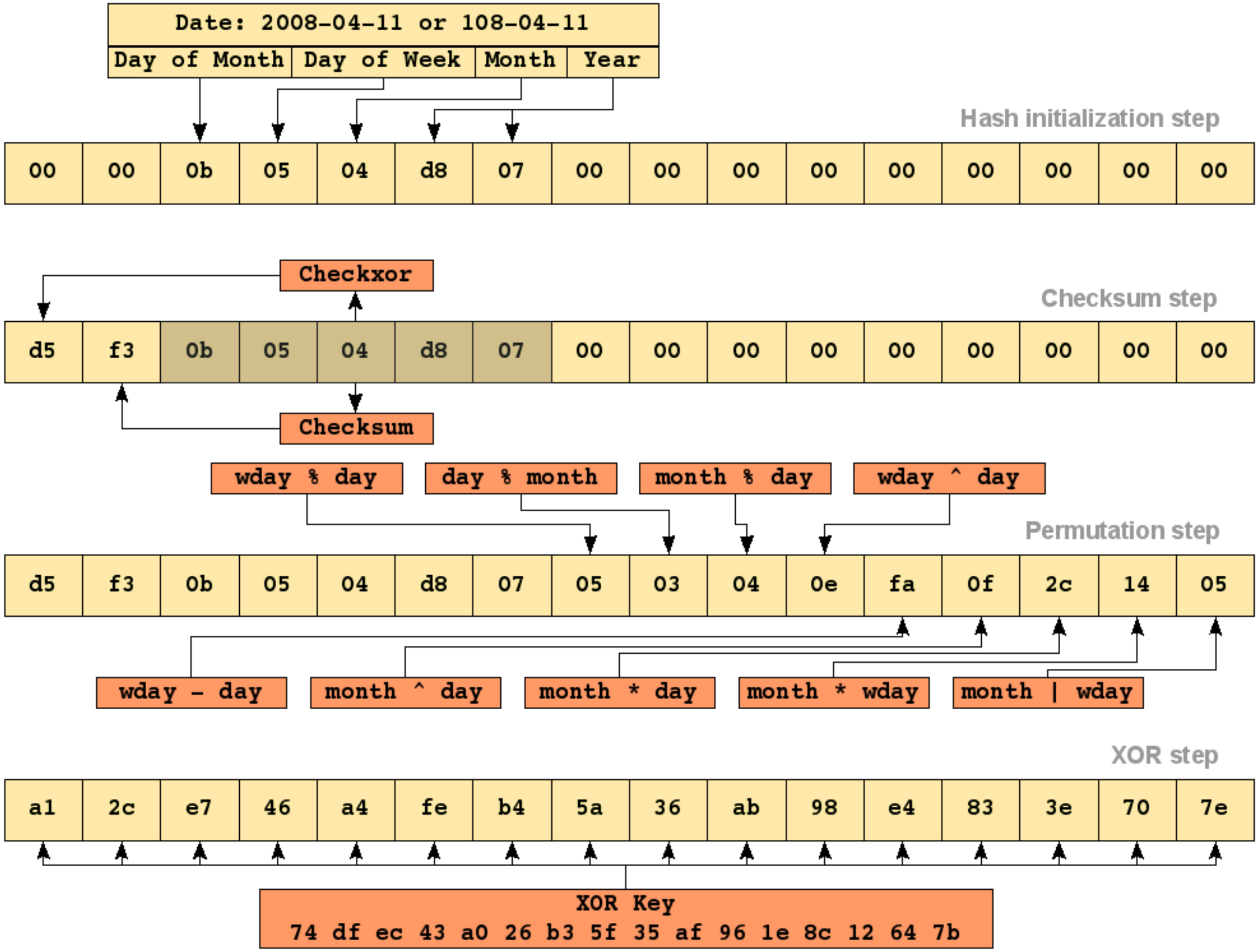
- Every file on Overnet is searched by its MD4 hash
- Basically it's a distributed hash table
  - Storm uses Kademlia's DHT implementation
  - Peers maintain a list of other peer hashes
  - If a peer doesn't know the location of what you're searching for, it will give you a list of peers closest to your searched file's hash



Storm P2P search

- Uses partial Overnet protocol
  - Not for communication, but location information
- Instead of sharing files, Storm encodes information into the hash itself – the current date and a checksum value
- 32 possible hashes are generated for each day of the year (peers search + or – 1 day also)
- Subcontrollers search for current date hashes
- Subnodes search for current date -1900 years

Generating a search hash set (part 1)



Date: 2008-04-11 or 108-04-11

Day of Month	Day of Week	Month	Year

Hash initialization step

00	00	0b	05	04	d8	07	00	00	00	00	00	00	00	00	00

Checkxor

Checksum step

d5	f3	0b	05	04	d8	07	00	00	00	00	00	00	00	00	00

Checksum

wday % day

day % month

month % day

wday ^ day

Permutation step

d5	f3	0b	05	04	d8	07	05	03	04	0e	fa	0f	2c	14	05

wday - day

month ^ day

month * day

month * wday

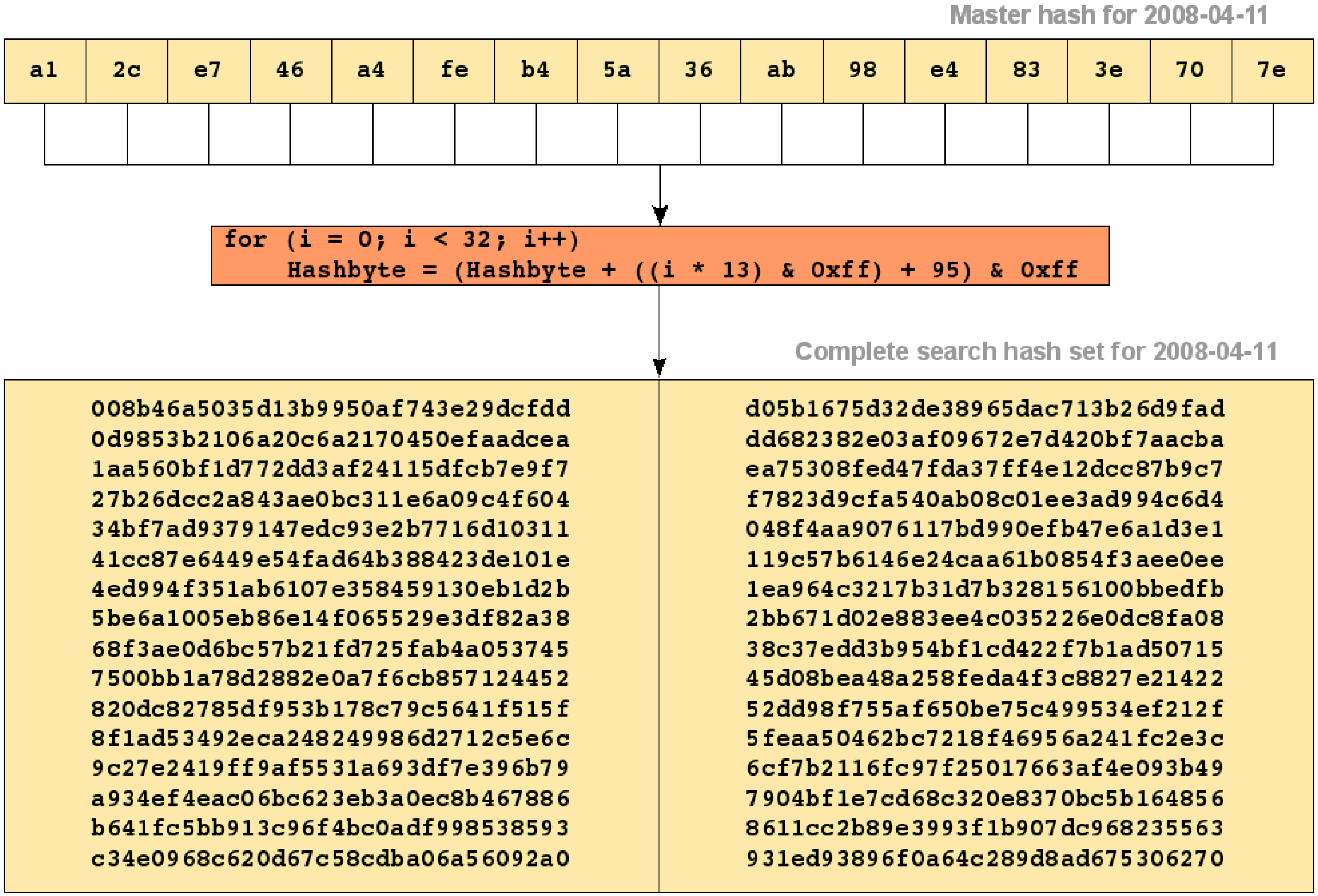
month | wday

XOR step

a1	2c	e7	46	a4	fe	b4	5a	36	ab	98	e4	83	3e	70	7e

XOR Key
74 df ec 43 a0 26 b3 5f 35 af 96 1e 8c 12 64 7b

Generating a search hash set (part 2)



Master hash for 2008-04-11

a1	2c	e7	46	a4	fe	b4	5a	36	ab	98	e4	83	3e	70	7e

```
for (i = 0; i < 32; i++)
    Hashbyte = (Hashbyte + ((i * 13) & 0xff) + 95) & 0xff
```

Complete search hash set for 2008-04-11

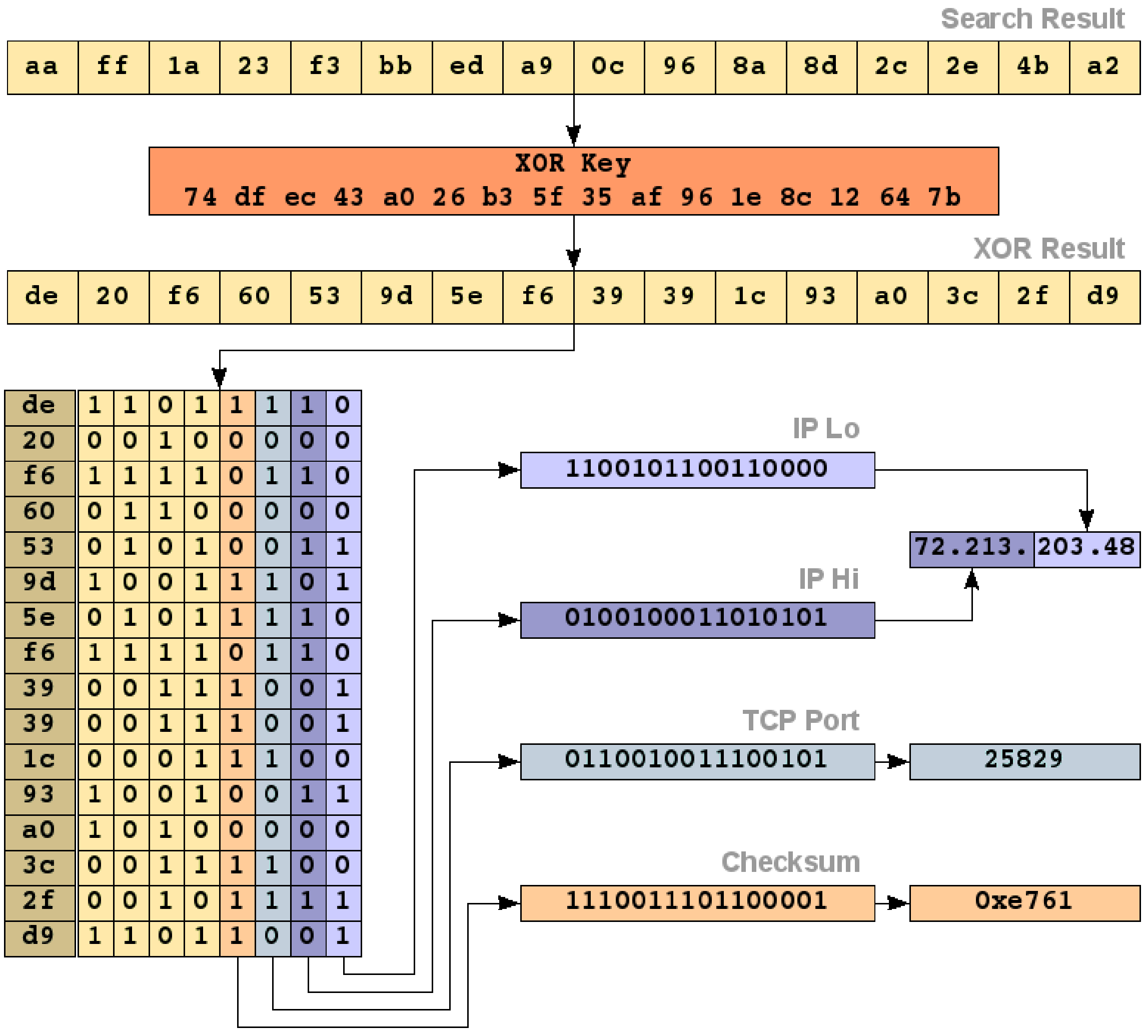
```
008b46a5035d13b9950af743e29dcfdd
0d9853b2106a20c6a2170450efaadcea
1aa560bf1d772dd3af24115dfcb7e9f7
27b26dcc2a843ae0bc311e6a09c4f604
34bf7ad9379147edc93e2b7716d10311
41cc87e6449e54fad64b388423de101e
4ed994f351ab6107e358459130eb1d2b
5be6a1005eb86e14f065529e3df82a38
68f3ae0d6bc57b21fd725fab4a053745
7500bb1a78d2882e0a7f6cb857124452
820dc82785df953b178c79c5641f515f
8f1ad53492eca248249986d2712c5e6c
9c27e2419ff9af5531a693df7e396b79
a934ef4eac06bc623eb3a0ec8b467886
b641fc5bb913c96f4bc0adf998538593
c34e0968c620d67c58cdba06a56092a0
```

```
d05b1675d32de38965dac713b26d9fad
dd682382e03af09672e7d420bf7aacba
ea75308fed47fda37ff4e12dcc87b9c7
f7823d9cfa540ab08c01ee3ad994c6d4
048f4aa9076117bd990efb47e6a1d3e1
119c57b6146e24caa61b0854f3aee0ee
1ea964c3217b31d7b328156100bbedfb
2bb671d02e883ee4c035226e0dc8fa08
38c37edd3b954bf1cd422f7b1ad50715
45d08bea48a258feda4f3c8827e21422
52dd98f755af650be75c499534ef212f
5feaa50462bc7218f46956a241fc2e3c
6cf7b2116fc97f25017663af4e093b49
7904bf1e7cd68c320e8370bc5b164856
8611cc2b89e3993f1b907dc968235563
931ed93896f0a64c289d8ad675306270
```

Publishing hashes

- Supernodes use Overnet publish packets to announce a location hash to peers
- More encoded data:
  - IP address
  - Storm control TCP port
- Other peers in the network cache location hashes and answer search queries
- Date hash in publish packet is state-dependent
  - Unactivated: current_date
  - Unactivated: current_date-1900 years

Decoding a Storm search result hash



Search Result

aa	ff	1a	23	f3	bb	ed	a9	0c	96	8a	8d	2c	2e	4b	a2

XOR Key
74 df ec 43 a0 26 b3 5f 35 af 96 1e 8c 12 64 7b

XOR Result

de	20	f6	60	53	9d	5e	f6	39	39	1c	93	a0	3c	2f	d9

de	1	1	0	1	1	1	1	0
20	0	0	1	0	0	0	0	0
f6	1	1	1	1	0	1	1	0
60	0	1	1	0	0	0	0	0
53	0	1	0	1	0	0	1	1
9d	1	0	0	1	1	1	0	1
5e	0	1	0	1	1	1	1	0
f6	1	1	1	1	0	1	1	0
39	0	0	1	1	1	0	0	1
39	0	0	1	1	1	0	0	1
1c	0	0	0	1	1	1	0	0
93	1	0	0	1	0	0	1	1
a0	1	0	1	0	0	0	0	0
3c	0	0	1	1	1	1	0	0
2f	0	0	1	0	1	1	1	1
d9	1	1	0	1	1	0	0	1

IP Lo
1100101100110000

72.213. 203.48

IP Hi
0100100011010101

TCP Port
0110010011100101 → 25829

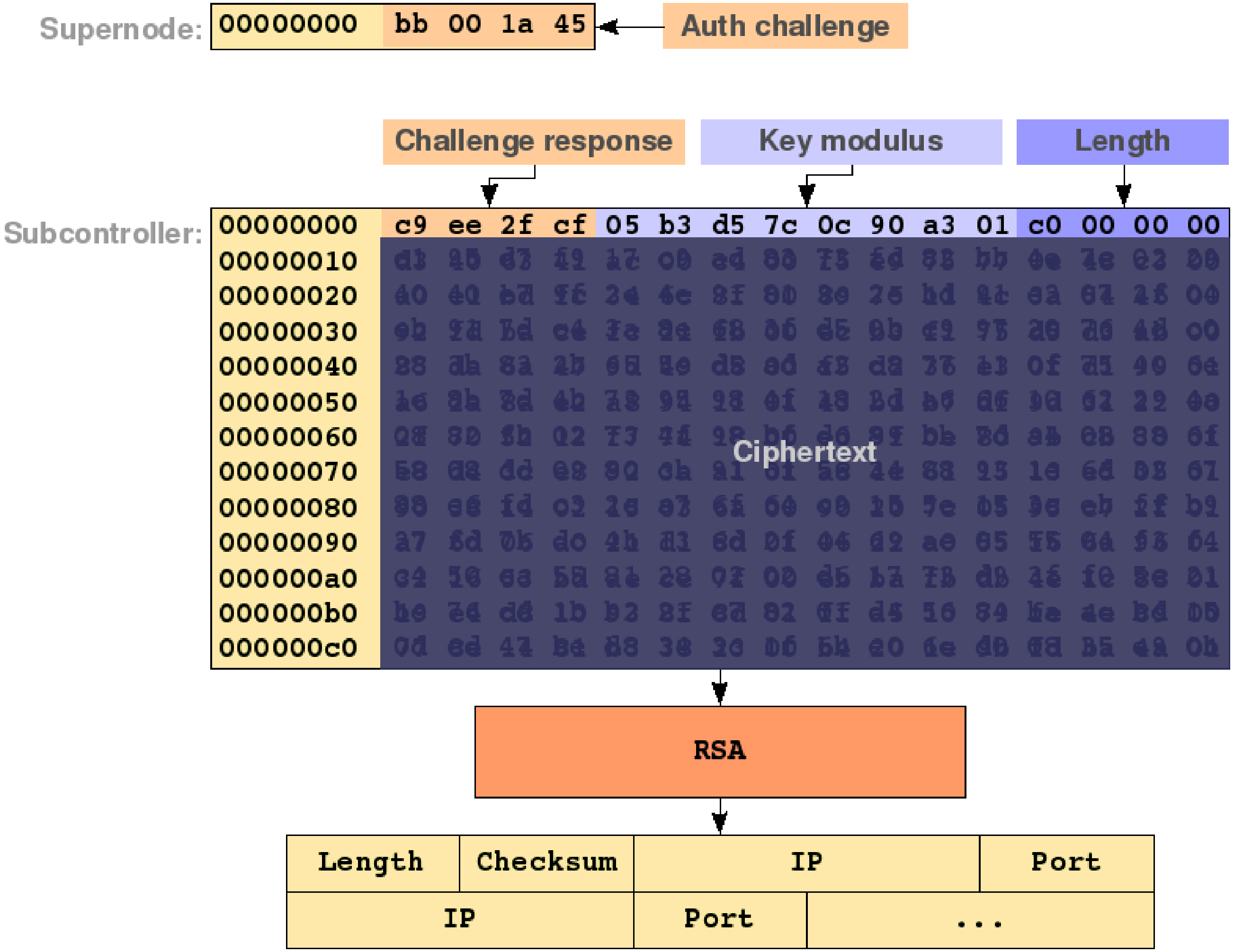
Checksum
1110011101100001 → 0xe761

The Information Security Experts

Activation of supernodes

- Subcontrollers search for a published current date hash on the network (unactivated Supernode)
- Subcontroller sends encrypted activation packet (BoL) to IP/TCP port found in published hash
- Supernode decodes list of subcontroller hosts it can to relay traffic to – it is now considered active
- Supernode begins to publish current_date-1900 years hash so subnodes can locate them and connect to the control port

Activation packet decryption



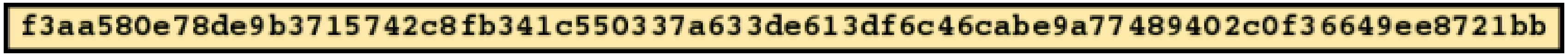
```
Supernode:      00000000  bb 00 1a 45

Subcontroller:  00000000  c9 ee 2f cf 05 b3 d5 7c 0c 90 a3 01 c0 00 00 00
                00000010  [illegible]
                00000020  [illegible]
                00000030  [illegible]
                00000040  [illegible]
                00000050  [illegible]
                00000060  [illegible]
                00000070  [illegible]
                00000080  [illegible]
                00000090  [illegible]
                000000a0  [illegible]
                000000b0  [illegible]
                000000c0  [illegible]
```

Length	Checksum	IP	Port
IP		Port	...

Storm's private overnet

- Overnet is filled with poison peers, nosy botnet researchers :)
- Storm's answer – keep the protocol, but encrypt the packets
  - Creates a new network – only Storm nodes can talk to each other
  - Encryption is simple XOR by embedded key
  - Could be used to segment botnets in case Storm author feels like selling turn-key spam system

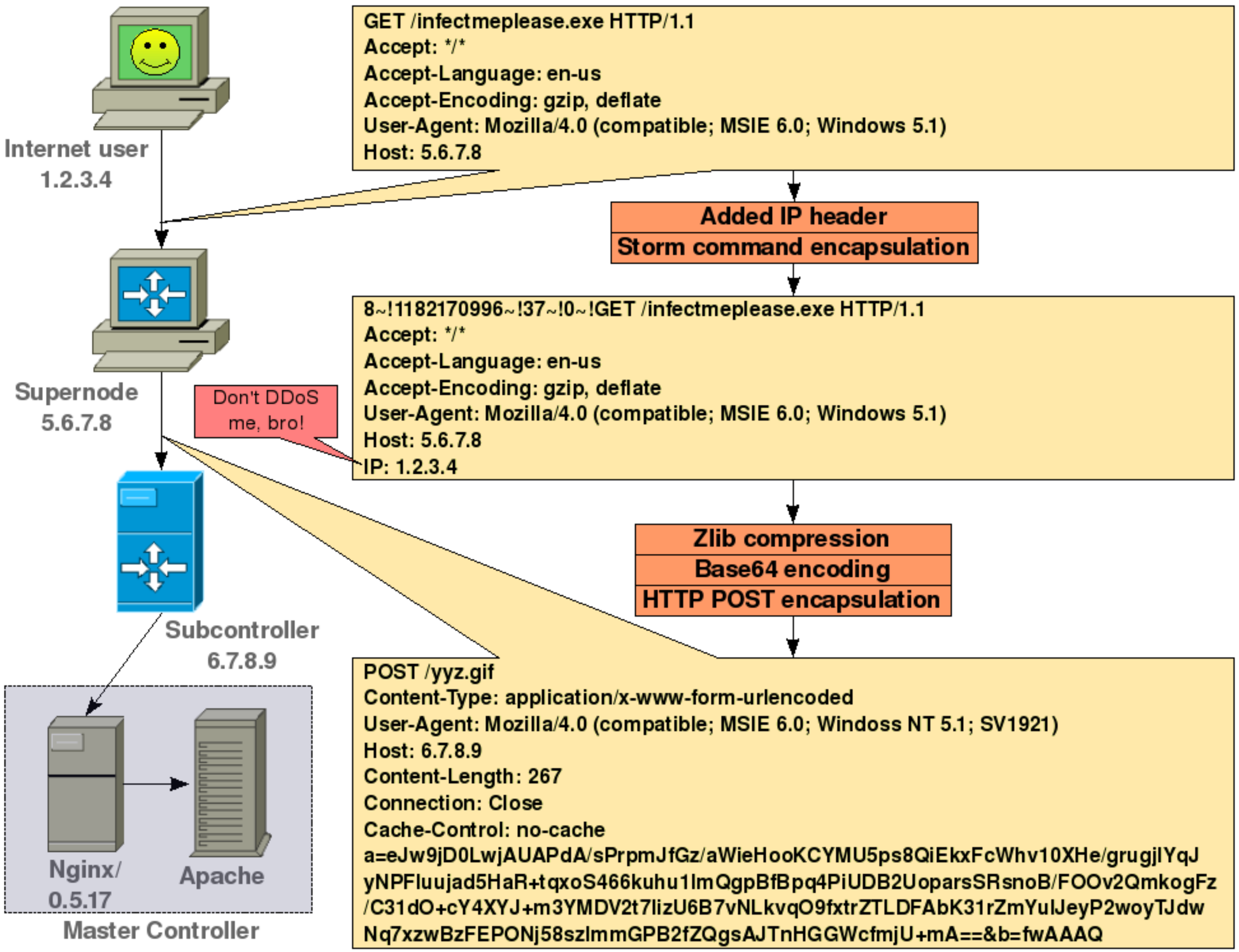


```
f3aa580e78de9b3715742c8fb341c550337a633de613df6c46cabe9a77489402c0f36649ee8721bb
```

Encapsulated HTTP relay

- Supernode listens on port 80 for HTTP requests
- Supernode relays zlib-compressed traffic to subcontroller via base64-encoded HTTP postdata
  - Request is encapsulated inside Storm command
  - Host IP HTTP header is added
- Nginx server passes HTTP request to master Apache server
- Reply from master is base64-encoded zlib-compressed data
- Reply is decoded and sent back to requestor

HTTP relay flow



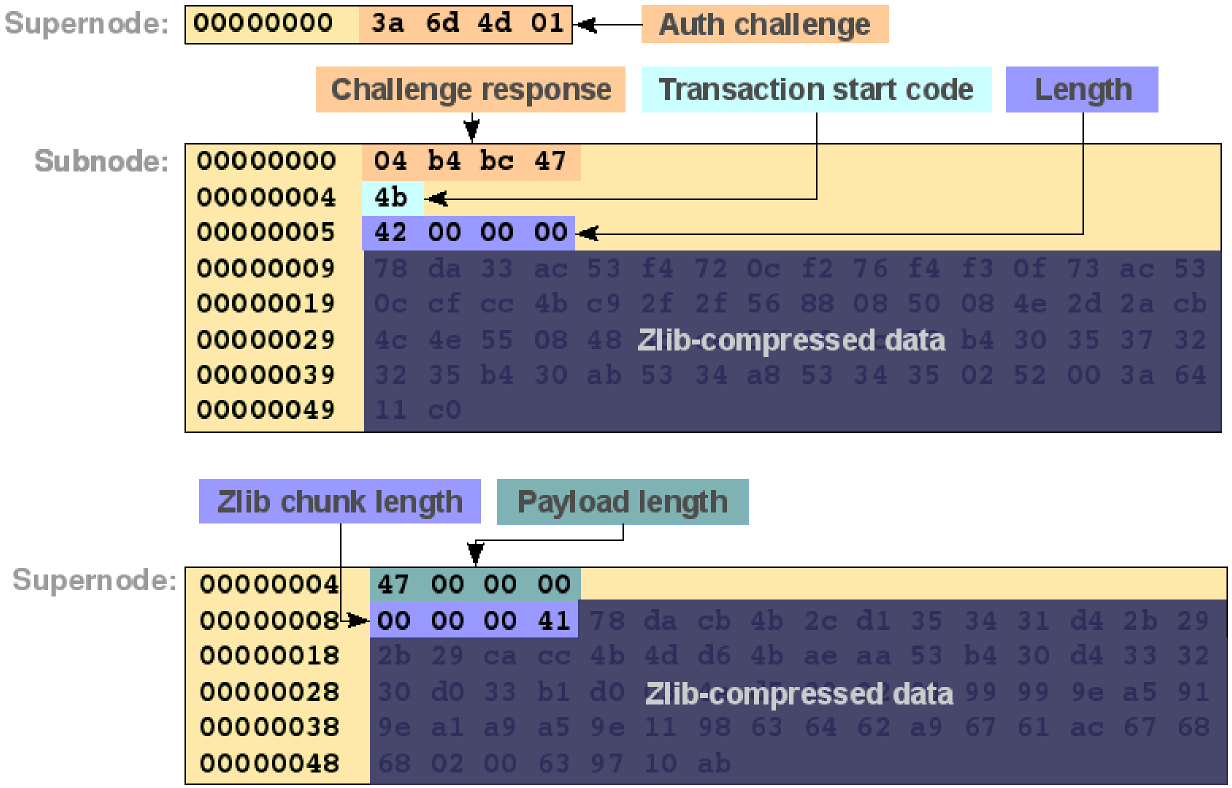
```
GET /infectmeplease.exe HTTP/1.1
Accept: */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows 5.1)
Host: 5.6.7.8
```

```
8~!1182170996~!37~!0~!GET /infectmeplease.exe HTTP/1.1
Accept: */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows 5.1)
Host: 5.6.7.8
IP: 1.2.3.4
```

```
POST /yyz.gif
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windoss NT 5.1; SV1921)
Host: 6.7.8.9
Content-Length: 267
Connection: Close
Cache-Control: no-cache
a=eJw9jD0LwjAUAPdA/sPrpmJfGz/aWieHooKCYMU5ps8QiEkxFcWhv10XHe/grugjlYqJ
yNPFluujad5HaR+tqxoS466kuhu1lmQgpBfBpq4PiUDB2UoparsSRsnoB/FOOv2QmkogFz
/C31dO+cY4XYJ+m3YMDV2t7lizU6B7vNLkvqO9fxtrZTLDFAbK31rZmYulJeyP2woyTJdw
Nq7xzwBzFEPONj58szlmmGPB2fZQgsAJTnHGGWcfmjU+mA==&b=fwAAAQ
```

Command packet exchange

- Subnode connects to TCP port of supernode
  - Zlib-compressed protocol relayed to controller via encapsulated HTTP requests as before



Supernode:

```
00000000  3a 6d 4d 01
```

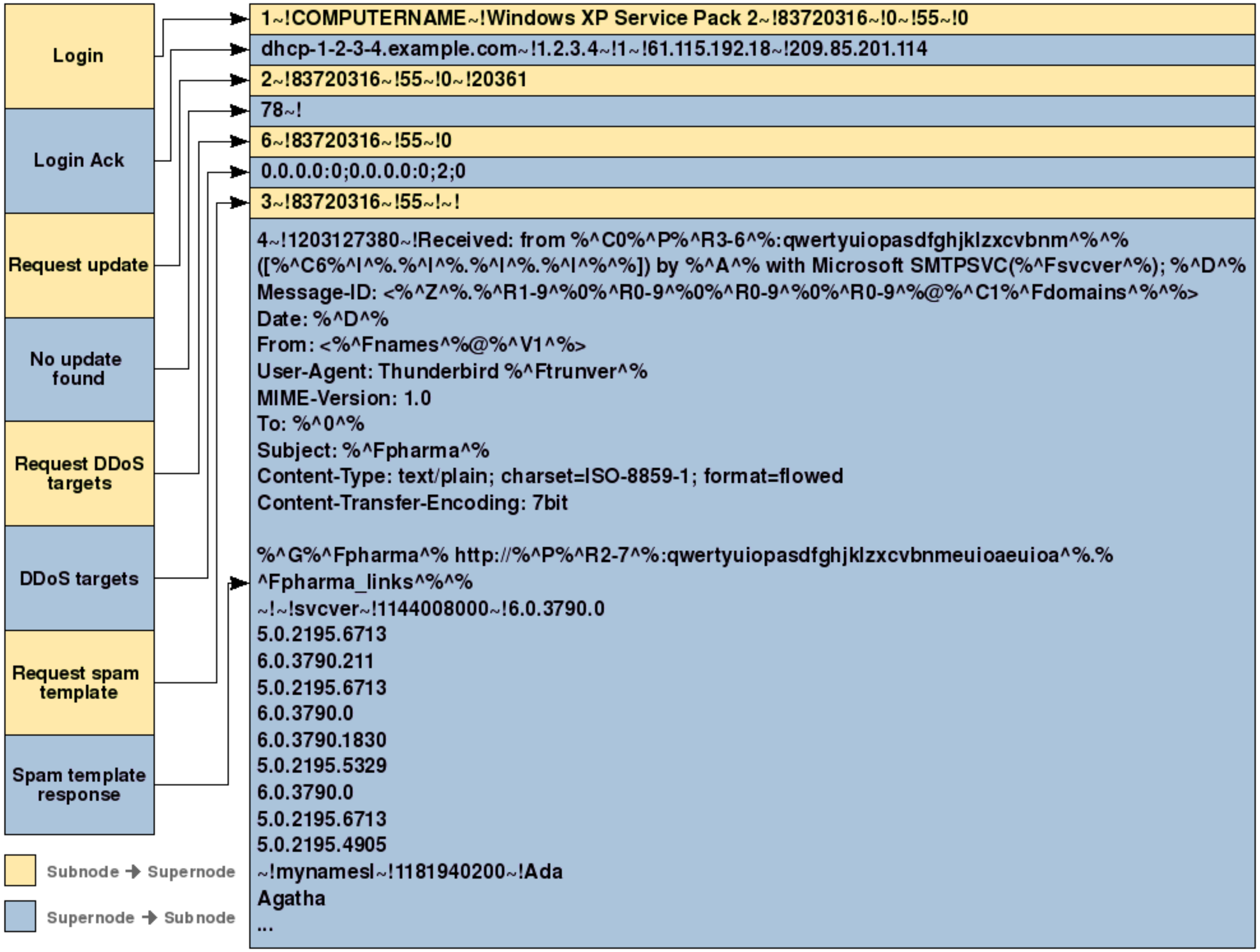
Subnode:

```
00000000  04 b4 bc 47
00000004  4b
00000005  42 00 00 00
00000009  78 da 33 ac 53 f4 72 0c f2 76 f4 f3 0f 73 ac 53
00000019  0c cf cc 4b c9 2f 2f 56 88 08 50 08 4e 2d 2a cb
00000029  4c 4e 55 08 48 Zlib-compressed data b4 30 35 37 32
00000039  32 35 b4 30 ab 53 34 a8 53 34 35 02 52 00 3a 64
00000049  11 c0
```

Supernode:

```
00000004  47 00 00 00
00000008  00 00 00 41 78 da cb 4b 2c d1 35 34 31 d4 2b 29
00000018  2b 29 ca cc 4b 4d d6 4b ae aa 53 b4 30 d4 33 32
00000028  30 d0 33 b1 d0 Zlib-compressed data 99 99 9e a5 91
00000038  9e a1 a9 a5 9e 11 98 63 64 62 a9 67 61 ac 67 68
00000048  68 02 00 63 97 10 ab
```

Decoded protocol exchange



```
1~!COMPUTERNAME~!Windows XP Service Pack 2~!83720316~!0~!55~!0
dhcp-1-2-3-4.example.com~!1.2.3.4~!1~!61.115.192.18~!209.85.201.114
2~!83720316~!55~!0~!20361
78~!
6~!83720316~!55~!0
0.0.0.0:0;0.0.0.0:0;2;0
3~!83720316~!55~!~!
4~!1203127380~!Received: from %^C0%^P%^R3-6^%:qwertyuiopasdfghjklzxcvbnm^%^%
([%^C6%^I^%.%^I^%.%^I^%.%^I^%^%]) by %^A^% with Microsoft SMTPSVC(%^Fsvcver^%); %^D^%
Message-ID: <%^Z^%.%^R1-9^%0%^R0-9^%0%^R0-9^%0%^R0-9^%@%^C1%^Fdomains^%^%>
Date: %^D^%
From: <%^Fnames^%@%^V1^%>
User-Agent: Thunderbird %^Ftrunver^%
MIME-Version: 1.0
To: %^0^%
Subject: %^Fpharma^%
Content-Type: text/plain; charset=ISO-8859-1; format=flowed
Content-Transfer-Encoding: 7bit

%^G%^Fpharma^% http://%^P%^R2-7^%:qwertyuiopasdfghjklzxcvbnmeuioaeuioa^%.%
^Fpharma_links^%^%
~!~!svcver~!1144008000~!6.0.3790.0
5.0.2195.6713
6.0.3790.211
5.0.2195.6713
6.0.3790.0
6.0.3790.1830
5.0.2195.5329
6.0.3790.0
5.0.2195.6713
5.0.2195.4905
~!mynamesl~!1181940200~!Ada
Agatha
...
```

Enumerating Storm

- Because overnet peers are constantly broadcasting their presence, one can walk all nodes in the network and get a count of the botnet size
- Brandon Enright at UCSD has done extensive work in mapping Storm nodes on both the public and private Overnet
- More information:

http://noh.ucsd.edu/~bmenrigh/exposing_storm.ppt

Questions?

Thank You!

- And a special thanks to GW and Brandon Enright for their insights during the development of this presentation