Botnets: Analysis of Command and Control Channels & Signature Generation
Malware Bot(net) Operation

1. **Bot (trojan) delivery** (e.g. drive-by via web browser, network service exploit, infection via USB drive)

2. **Bot setup** - turns host to “zombie”, connect to Botnet (persist, conceal and communicate)

3. **Stand-by mode** (optional)

4. **Execute commands** (e.g. spamming, DoS, credential theft)
How Botnets Work

1. Botmaster exploits the vulnerability on the victim.
2. The victim downloads the actual bot binary.
3. Bot contact the IRC server address in the executable, including resolving the DNS name.
4. The bot joins an IRC channel.
5. The botmaster sends out commands via IRC channel.
Bot C&C channel Requirements

- **Stealthy**
  - Avoid custom ports
  - Merge normal traffic

- **Availability**
  - Common (public) protocol/service
  - Legitimate access to channel (Windows API)

- **Usability**
  - Quality of service (application level protocol)
  - Command multicasting/broadcasting (control scalability)
  - Centralized vs P2P

*Bandwidth is not important*
Bot C&C channels
Modern Techniques Trend

Invisibility

DNS
Social Net
P2P
HTTP
IRC

Time

Conficker (10.5 Mil bots)
TDL4 (4.5 Mil bots, 2010)
BredoLab (30 Mil bots, 2009)
Zeus (3.6 Mil)
Agobot, Sdbot, m-IRC (<1 mil, 2004)
Zeus (3.6 Mil)
The Threats from Botnets

- Types of attack
  - DDoS attacks.
  - Spam.
  - Clickfraud.
  - Spreading new malware.
  - Cracking.
  - Manipulating online polls
Botnet Detection

- IRC botnet
  - IRC port, may be on non-standard port
  - Monitor IRC payload for known command
- Behavioral characteristics
  - Response
    - Constant response time, fast join
  - Long standing connection
  - Bots are not talkative
- Machine learning techniques
  - Using labeled data to build classifier
- Track the botnet by honeypot
  - Use honeypot to get infected
Why IRC?

Original motivation

- Internet Relay Chat (standard protocol)
- IRC is not firewalled by most users
- IRC traffic load is predictable (between 2-3 pm, e.g. office usage pattern, do not cause anomaly)
- IRC endpoints are 100% legal (known IRC servers)
- C&C commands blend in normal data traffic

Practically easy to avoid detection (network-based IDS, e.g. Snort)
IRC Bot operation (server independence)

Source: http://www.windowsecurity.com/
Fighting with IRC C&C
Practical approaches

Prevention:
- Block IRC (port 6667)
- Allow only certain IRC channels
- IRC channel remap via IRC proxy (potential)

Detection:
- Anomaly in IRC usage pattern (e.g. night time IRC connections from office computers, previously unknown IRC channel, strange IRC server)
- IRC message correlation (e.g. consistent command/replay pattern)

Can black hats strike back?

Smart IRC C&C:
- Randomize IRC command and replay (generate unique communication protocol at each host)
- Collect usage pattern and blend in (utilize/piggyback user’s IRC channels)
Network Intrusion Detection System (NIDS)
BotNet C&C detection

- Network traffic (packet) capture: libpcap, WinPcap
- Traffic analysis: Wireshark, tcpdump, ngrep, Ettercap*
- NIDS: Snort, Bro
Bot generators (SpyEye)

Screenshots were obtained from public web-sites
Analyzing IRC Bot

Tasks:

1. Reverse Engineer the image
   - IRC server ID, channel and user names (generator)
   - C&C Commands
2. Set up an environment (fake IRC server, IRC client)
3. Trigger bots commands (IRC command messages and observe malware’s behavior)
Demo: bBot (rBot based)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IRC session:</strong></td>
<td></td>
</tr>
<tr>
<td>IRC server name</td>
<td>botirc.net (static)</td>
</tr>
<tr>
<td>IRC channel name</td>
<td>#test</td>
</tr>
<tr>
<td>IRC user name</td>
<td>[M]bBot</td>
</tr>
<tr>
<td><strong>C&amp;C commands:</strong></td>
<td></td>
</tr>
<tr>
<td>Log keys</td>
<td>keylog file</td>
</tr>
<tr>
<td>Remote access to host</td>
<td>httpserver</td>
</tr>
<tr>
<td>Download a file</td>
<td>do\wnl04d</td>
</tr>
</tbody>
</table>
Control and Command Infrastructure

- IRC-based
- P2P botnet
  - Servent bots
  - Client bots
  - Communicate via peer list.
    - Only servent bots are in peer list
  - Avoid single point of failure.
Control and Command Infrastructure

- **HTTP botnet**
  - Bot contact C&C server with its info embedded in URL.
  - Attackers send commands via HTTP response.
  - Communicating in between HTTP's noise
  - Bobax Trojan

```
localhost MySQL: (from getcmd.php)
SELECT cmd FROM bots WHERE uid=A1
SELECT cmd FROM bots WHERE uid=A2
SELECT cmd FROM bots WHERE uid=A3
```

GET /getcmd.php?uid=A1
GET /getcmd.php?uid=A2
GET /getcmd.php?uid=A3

http://botnet.org/

send("cmd") to all bots!

POST /botnet-admin.php

authorization packet

h a c k e r

infected machine A1

infected machine A2

infected machine A3
Control and Command infrastructure

- IM botnet
  - Send command to all bots on contact list, via instant message.
  - The botmaster only need to send to a small number of message.
    - Each recipients can then spread to their own contacts, and then go on.
  - Sdbot, spreading via AIM.

* SCMD -- send command
* CLB  -- contact list of infected bots
* AUTH -- authorization on icq.com
Dr. Arnur Tokhttabayev,
Center for Secure Information Systems
George Mason University
Computer Science Department
Research I, Rm 435

E-mail:
- atokhtab@gmu.edu
- arnur78@gmail.com