Framework for Botnet Emulation and Analysis

Ph.D. Thesis Defense
Christopher Lee
Advisor: Dr. John A. Copeland

Please silence cell phones and be easy on the WiFi.
Presentation Outline

• Background
• Challenges
• Previous work
• Rubot
• Experiments
• Conclusion
Themes

- Studying botnets is tricky
- Current approaches are not scientific
- The Rubot Framework resolves these issues
1. Background

- History
- Criminal Uses
- Defenses
1.1 History

- The Good (Agents)
- The Bad (IRC-based bots)
- and The Ugly (HTTP/P2P-based bots)
1.2 Criminal Uses

- Expanding infrastructure
- Making money
1.3 Defenses

- Host-based
- Network-based
- Remediation
- Legal
Challenges

- Obtaining source code
- Sharing among researchers
- Legal issues
- Lack of ground truth
- Unreproducible
- Costly infrastructure
Techniques Used

- Review of source code
- Binary reverse engineering
- Live memory analysis
- Lightweight emulation
- Crawlers
3.1 Source Code

Ryan1918 Dot COM
Your number one source for everything!
Finding Open Dumps

<table>
<thead>
<tr>
<th>Name</th>
<th>Last modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent Directory</td>
<td>04-May-2005 19:35</td>
</tr>
<tr>
<td>agobot/</td>
<td>04-May-2005 19:35</td>
</tr>
<tr>
<td>dsnx/</td>
<td>04-May-2005 19:35</td>
</tr>
<tr>
<td>gt/</td>
<td>04-May-2005 19:35</td>
</tr>
</tbody>
</table>
// this is a nice all-in-one solution to encrypt
// recommended even for novice users.

void Crypt(unsigned char *inp, DWORD inplen, unsigned char *outp, DWORD outlen)
{
    // we will consider size of sbox 256 bytes
    // (extra byte are only to prevent any mishaps)
    unsigned char Sbox[257], Sbox2[257], temp = 0;
    unsigned long i = 0, j = 0, t = 0, x = 0;

    keylen = 0;

    // this unsecured key is to be used only where
    unsigned char OurUnSecuredKey[49]; // Change
    OurUnSecuredKey[0] = 10; OurUnSecuredKey[1] = 1;
Lots of features

![Bar chart showing various features with DDoS having the highest value at 10.0, followed by Proxy, Harvest, Download, Backdoor, Scan, and WebServer with the lowest value at 0.]

- DDoS
- Proxy
- Spam
- Harvest
- Download
- Backdoor
- Scan
- WebServer
3.2 Reverse Engineering

- Decompiling
- Tracing
3.3 Memory Inspection

- Searching through the dumps
  - Found peer table data structure
- Watching memory transactions
  - Understood encryption and obfuscation
3.4 WineBots

- WINE translates WIN32 to Linux and back
- Augmented the translation for socket calls
- Separated registries and IP stacks
- Now able to run as many bots as routable IPs == IP-bounded
3.4 WineBots (II)
3.5 Crawling

Got Peers?
3.5 Crawling (II)

Got Peers?
Rubot

• What’s NOT included by design
• Focus on network-centric functionality
• Goal was to promote research into:
  • taxonomies, detection, mitigation, and future design
So what is included?

- “Vulnerable” services
- Spamming
- Worms
- Packeting
- Scanning
- Proxy, Fast Flux, Updating
- Various control channels
Anything else?

- Various “supporting actors”
- Web server
- SMTP/POP3 server
- Authoritative DNS server
- Composite models: GTBot, Nugache, Storm
Exploits/Vulnerabilities

• There are no real exploits.

• Vulnerable services have a list of exploit payloads with appropriate callbacks

• Servers
  • TCPServer, UDPServer, WebServer

• Clients
  • WebBrowser, WebTrojan, POP3Trojan
Spam

Template

Addresses

Schedule

SPAM

SPAM

SPAM
Worms

TCP and UDP-based worms

Requires Vulnerable Service

Spreads with given exploit at given rate
Packeting (DDoS)

- TCP SYN Flood
- UDP Flood (with variable payload)
- ICMP Flood (defaults to Echo Request)
- IRC Clone Flood
Scanning

- Linear-range
- Random-range
- Subnet
- Random-without-repeat
Proxy & FastFlux

- Proxy arbitrary TCP streams and UDP packets
- FastFlux model accepts DNS records and serves them authoritatively
Updating

- Allows the bot to change behavior/versions
- Kills current bot and restarts it with a new configuration
Control Channels

- IRC
- HTTP
- TCP-based P2P
- UDP-based P2P
Experiments
IRC Bot

SAY HELLO.

Hello Hello Hello Hello Hello Hello
HTTP Bot
FluxNet

- Botmaster
  - Places records into fast flux node
  - Updates TLD for domains to point to nodes
  - Sends spam with domain
- Clients query TLD for domain and contact flux nodes acting as authoritatives
TCP P2P Bot

Bot A
Bot B
Bot C
Bot D
Bot E
Bot F
Bot G
Bot H
Bot I
Bot J
Bot K
Supernode A
Supernode B
Me
UDP P2P Bot

Worms
Nugache

- Keying and authentication
- Object-oriented command set
- Peer discovery
Storm

Master Proxy → Web Server

Sub Controller

Overnet Network

Super node

Sub node

Sub node

Sub node

HTTP

Victim

POP3

SMTP POP3
Contributions

• Rubot models and framework for botnet experimentation

• Large-scale, WINE-based botnet monitoring

• Evaluation of crawler vs. passive enumeration of P2P botnets

• Memory snapshot blind analysis technique

• Survey of botnet source code bases
Conclusion

• Botnets are used for crime
• Hard to study
• Tried a lot of techniques on Nugache and Storm
• Created Rubot to support botnet research

• Taxonomies, Detection, Mitigation, and Future Architectures
Future Work

- This framework opens up a world of possibilities of future work.

- Models that need to be included in the next version:
  - Testbed support
  - NS3 Integration
  - Instant messenger models
  - Sensor models
Selected Publications

- B. Kang, E. Chan-Tin, C. Lee, et al., Towards Complete Node Enumeration in a Peer-to-Peer, accepted to ASIACCS 2009
- D. Dagon, N. Provos, C. Lee, W. Lee, Corrupted DNS Resolution Paths: The Rise of a Malicious Resolution Authority, NDSS '08
- D. Dagon, G. Gu, C. Lee, W. Lee, A Taxonomy of Botnet Structures, ACSAC'07
- C. Lee and J. Copeland, Flowtag: a collaborative attack-analysis, reporting, and sharing tool for security researchers, VizSec 2006