Detecting Bots with Automatically Generated Network Signatures

Peter Wurzinger, Leyla Bilge, Thorsten Holz,
Jan Goebel, Christopher Kruegel, Engin Kirda

International Secure Systems Lab,
Vienna University of Technology, {pw,tho}@seclab.tuwien.ac.at
Institute Eurecom, France, {bilge,kirda}@eurecom.fr
University of Mannheim, goebel@informatik.uni-mannheim.de
University of California, Santa Barbara, chris@cs.ucsb.edu

Outline

- Introduction
- Detection models - overview
- Generating detection models
- Analysis and evaluation

The Botnet Threat

- Tool of choice for Internet criminals
- Useful for many purposes:
  - Spam
  - DDoS
  - Fast Flux
- Extremely powerful
- Simple to deploy and maintain

The Botnet Threat

- Network of compromised computers
- Remotely operated by botmaster
- Command and control channel (C&C)
  - IRC: classic, Agobot
  - HTTP: more stealthy, Bobax
  - P2P: robust, Storm worm

Botnet Counter Measures

- Host-based
  - Anti-virus software
  - Relies on binary signature database (polymorphism)
  - Host installation required
- Network-based
  - Intrusion detection
  - No requirements from end-user
  - Relies on (hand-crafted) network signatures

Goal of our Work

- Network-based botnet detection
  - Deployed on gateway
  - Transparent to the user
- Automatically generated signatures
  - No costly work has to be performed by human experts
  - Signatures for new botnets can be added easily
- C&C protocol agnostic
  - Signatures can be generated regardless of C&C protocol
  - No expert knowledge about a specific botnet is required

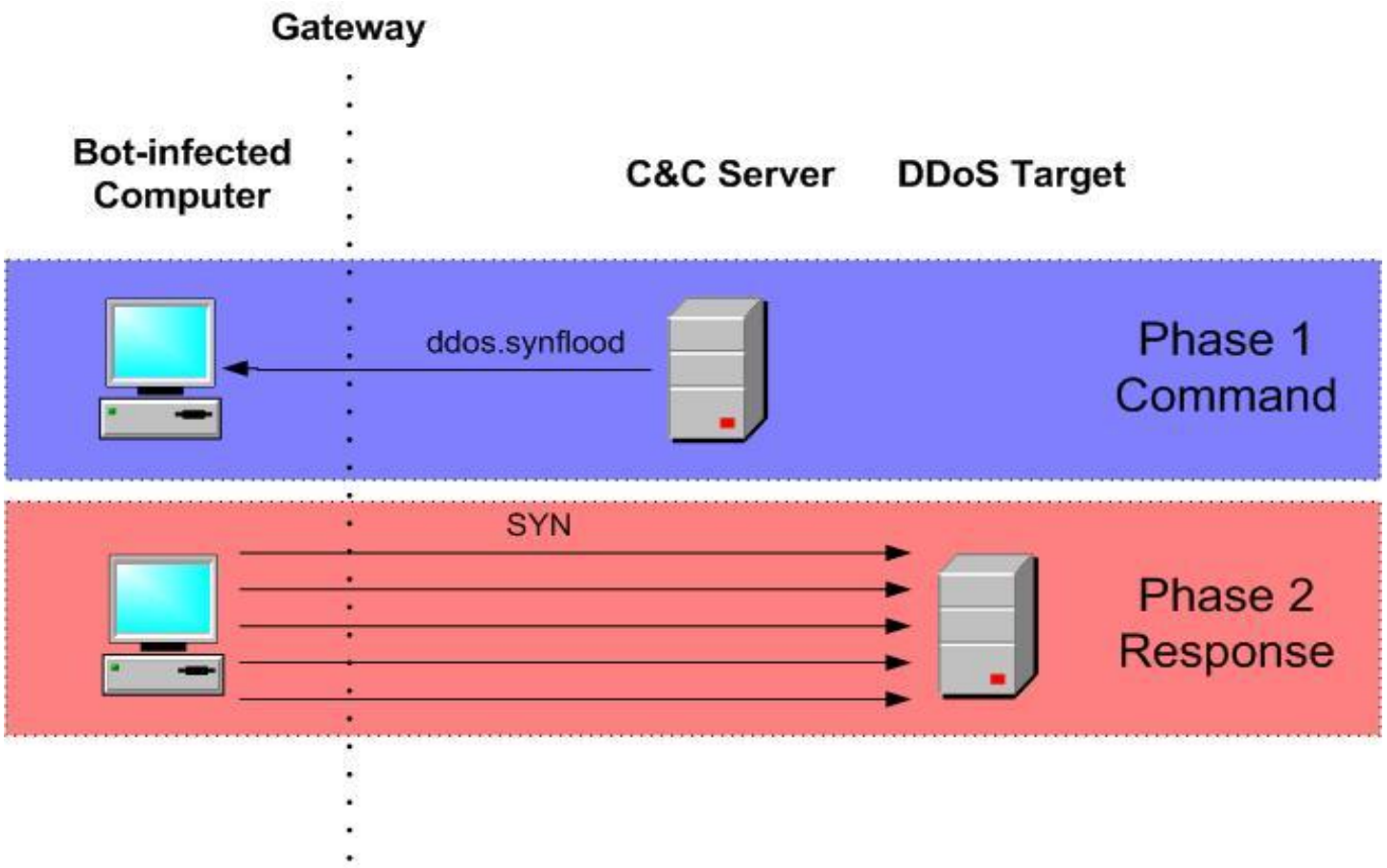
Detection Models

Detection Models

- Characterisation of bot traffic using two phases
  - Phase 1: Bot receives command
  - Phase 2: Bot executes command
- Both phases are visible in network traffic

- Example:
  - Phase 1 (command): string „advscan“ is transmitted to host X
  - Phase 2 (response): X transmits many SYN packets to different recipients

Detection Models



Generating Detection Models

Generating Detection Models - Overview

- Input: Network traces of similar bot programs
- Find sudden changes in the bot‘s network behavior
- These changes are most likely due to a previously received command!
- Characterize traffic content before the change -> command model (phase 1)
- Characterize network behavior after the change -> response model (phase 2)

Obtaining Bot Network Traces

- Assemble a „bot family“
  - Set of similar sample bot programs
  - Similar C&C mechanism
  - Not necessarily from same botnet
- Execute samples in a controlled environment
  - Internet access open, so C&C communication works
  - Run-time: several days
  - Goal: collect command/reponse pairs

Locating Bot Behavior Changes

- Identify points in time where a sudden change of the bot‘s network behavior has occurred
- Assumption
  - Change is due to a previously received command
  - New network behavior is a manifestation of a bot response
  - Command (data that is directly related to the bot‘s action) was received within a restricted time interval before the change

Locating Bot Behavior Changes

- Time-series
- Partition into **discretization intervals** of equal length
- Set of low-level **network features** each interval is inspected for:
  - Number of packets
  - Cumulative size of packets
  - Number of different IPs contacted
  - Number of different ports contacted
  - Number of non-ASCII bytes in payload
  - Number of UDP packets
  - Number of HTTP packets (Port 80)
  - Number of SMTP packets (Port 25)

Locating Bot Behavior Changes

- Change point detection
- Modified variant of CUSUM algorithm

- We know the interesting points in time now!
- → command in traffic before
- → response in traffic after

Response Model (Phase 2)

- Generalisation steps:
  1. Description of network behavior in one discretization interval
  2. Description of network behavior of the discretization intervals that form one bot response
  3. Description of a class of bot responses

- We already have 1. → network features

Response Model (Phase 2)

- Generalization to describe sequence of discretization intervals that form one bot response

- Each period between two detected change points exhibits consistent bot network behavior
- This consistent behavior represents one bot response
- **Behavior profile:** average values of the network features per discretization interval

Response Model (Phase 2)

- Generalization to describe a class of bot responses
- Clustering of similar bot responses based on behavior profiles
- Each cluster represents one type of bot behavior
- **The response model (phase 2) is the average of all behavior profiles of a cluster**

Command Model (Phase 1)

- We have response models, now what are the corresponding command models?

- Reuse clusters of similar bot responses
- Inspect traffic that precedes responses in same cluster
- Extract similarities

Command Model (Phase 1)

- Find **token sequences** in the network traffic that are characteristic for triggering the observed response
- Tokens can consist of:
  - the command itself
  - frequently used parameters
  - artefacts from the surrounding C&C protocol

Detection Model Summary

- Phase 1 – command
  - Token sequence
  - Network content that is characteristic to show up before a certain bot response begins
- Phase 2 – response
  - Description of the response using network features
  - Network-level characterization of a type of bot response

Evaluation

Evaluation

- Generated detection models for
  - various IRC bots
  - Bobax
  - Storm worm
- Translated them into Bro NIDS policy script

Example


```
signature irc {
    dst-ip == local_nets
    payload /.* PRIVMSG #.* :\.asc .*5 0 .*/
}

#DIFFERENT IPS > 20
    (within 50 seconds)
```

Evaluation – Detection Performance

- Evaluation of our generated signatures using cross-validation on bot network traces
- Detection rate: 88%

Evaluation - Preciseness

- Real-world deployment on well maintained networks
- No bot infections expected
- Students residential homes network
  - /21 range, densely populated
  - observation period: 55 days
  - no false positives
- University network (/20, 3 months)
  - /20 range, medium populated
  - observation period: 102 days
  - only 11 IPs falsely raised an alert

Conclusion

- 2 Phases: Command/Response
- Our system produces botnet detection models
  - for network-based detection
  - without expert knowledge about specific botnets
  - automatically
- Deployment on gateway, end-user not involved
- Effective detection with few false positives

Publication

This work is presented also at **ESORICS 2009**.

„Automatically Generating Models for Botnet Detection“

Check out the paper at http://www.iseclab.org

Questions?

Thank you for your attention!
I'd be happy to answer all of your questions!