BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation

Guofei Gu, Phillip Porras, Vinod Yegneswaran,
Martin Fong, Wenke Lee

USENIX Security Symposium (Security '07)

Presented by Nawanol Theera-Ampornpunt

Agenda

- **Bot infection process**
- **BotHunter**
- **Evaluation**
- **Conclusions**

PURDUE UNIVERSITY

Bot Infection Sequence (1)

- Bot's activities are viewed from the egress point
- Infection dialog sequence:
  - E1: External to Internal Inbound Scan
  - E2: External to Internal Inbound Exploit
  - E3: Internal to External Binary Acquisition (egg download)
  - E4: Internal to External C&C Communication
  - E5: Internal to External Outbound Infection Scanning

Bot Infection Sequence (2)

- Example: a variant of Phatbot (aka Gaobot)
  - E1: Probe an address range for exploitable network services or responses from existing Trojan backdoors
  - E2: Launch an exploit, or log in to the host using the backdoor
  - E3: The victim host is directed to download the full Phatbot binary
  - E4: Establish a connection to the C&C server established over an IRC channel
  - E5: Scan other external victims on behalf of the botnet

Bot Infection Sequence (3)

- The events might not happen in order
- Some of the events might not be detected
  - With mobile devices and VPNs, infection of an internal asset may take place outside the network perimeter
- In this paper, bot infection declaration requires a minimum of
  - Condition 1: Evidence of local host infection (E2) and evidence of outward bot coordination or attack propagation (E3-E5); or
  - Condition 2: At least two distinct signs of outward bot coordination or attack propagation (E3-E5)

BotHunter (1)

- Designed as a passive system based on three intrusion detecting systems (IDSs) monitoring inbound and outbound traffic flows
  - The alerts produced by these IDSs are intended as the input to BotHunter rather than an alert to be processed by administrators
- The three IDSs are:
  - Statistical sCan Anomaly Detection Engine (SCADE)
  - Statistical payLoad Anomaly Detection Engine (SLADE)
  - Signature Engine (Bot-specific heuristics)
- The IDSs are built on top of ***Snort***, an open-source network intrusion prevention and detection system

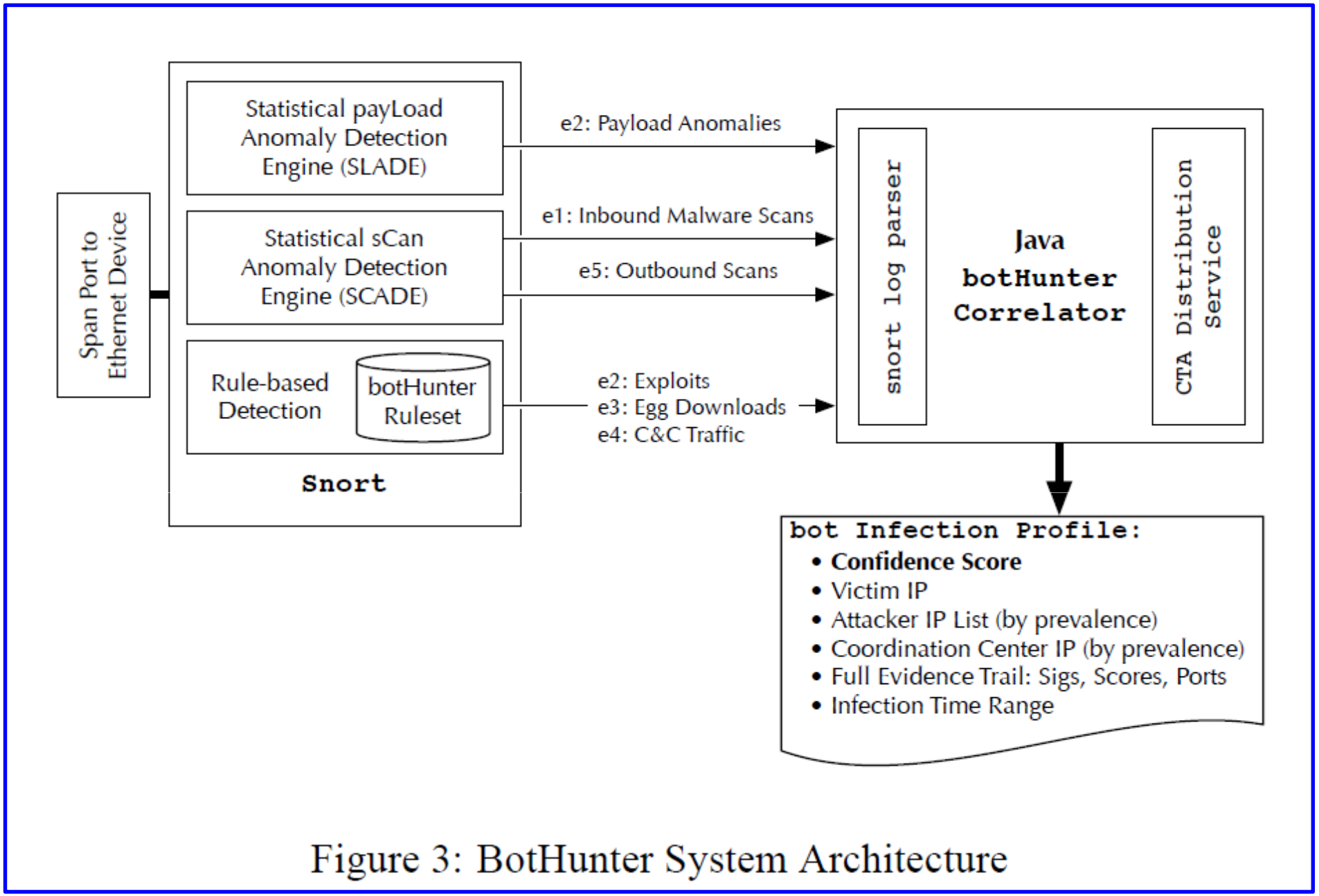

Figure 3: BotHunter System Architecture

BotHunter (2)

- SCADE is designed to detect inbound scans (E1) and outbound attack propagation (E5)
- Inbound scans:
  - Modern bots can take advantage of 15 exploits on average, to improve the success rate of exploitation
  - Depending on how the attack source scans for its potential target, we are likely to see some failed connection attempts prior to a successful infection
  - E1 scan detection is based on the number of failed connection attempts, with more weight put on ports often used by malware

BotHunter (3)

- Outbound scan detection is based on a voting scheme of three anomaly detection models that track all outbound connections for each internal host:
  - Outbound scan rate: detects internal hosts that conduct high-rate scans across large sets of external addresses
  - Outbound connection failure rate: detects abnormally high connection fail rates, with different weights for different ports
  - Normalized entropy of scan target distribution: calculates a Zipf (power-law) distribution of outbound address connection patterns. Normalized entropy = $\frac{H}{\ln(m)}$

BotHunter (4)

- SLADE examines the payload of every request packet and generates an E2 alert if its lossy n-gram frequency deviates from an established normal profile
  - A lossless n-gram model needs to store 256^n features, and this is impractical for high n-grams
  - To fix this problem, these features are stored in a lossy manner while still maintaining approximately the same accuracy as the original n-gram version

Table 1: Performance of 1-gram PAYL and SLADE

	DFP(%)	0.0	0.01	0.1	1.0	2.0	5.0	10.0
PAYL	RFP(%)	0.00022	0.01451	0.15275	0.92694	1.86263	5.69681	11.05049
	Detected Attacks	1	4	17	17	17	18	18
	Detection Rate(%)	0.8	17.5	69.1	72.2	72.2	73.8	78.6
SLADE	RFP(%)	0.0026	0.0189	0.2839	1.9987	3.3335	6.3064	11.0698
	Detected Attacks	3	13	17	18	18	18	18
	Detection Rate(%)	20.6	74.6	92.9	99.2	99.2	99.2	99.2

BotHunter (5)

- Signature Engine: Bot-Specific Heuristics
  - secondary source for direct exploit detection (E2)
  - primary source for binary downloading (E3) and C&C communications (E4)
- Number of rules (total=1,383 rules):
  - 1,046 E2 rules
  - 71 E3 rules
  - 246 E4 rules
  - 20 E5 rules

BotHunter (6)

- Based on the three IDS *sensors*, BotHunter tracks the sequences of dialog warnings for each local host over a temporal window

Int. Host	Timer	E1	E2	E3	E4	E5
192.168.12.1		$A_a \ldots A_b$				
192.168.10.45			$A_c \ldots A_d$		$A_e \ldots A_f$	
192.168.10.66			A_g			
192.168.12.46					$A_h \ldots A_i$	$A_j \ldots A_k$
:						
192.168.11.123		A_l	$A_m \ldots A_n$	A_o		

Figure 4: BotHunter Network Dialog Correlation Matrix

BotHunter (6)

- For each local host, the weights for all events are added up to compute a score

	Coefficients	Standard Error
E1	0.09375	0.100518632
E2 rulebase	0.28125	0.075984943
E2 slade	0.09375	0.075984943
E3	0.34375	0.075984943
E4	0.34375	0.075984943
E5	0.34375	0.075984943

Table 2: Initial Weighting

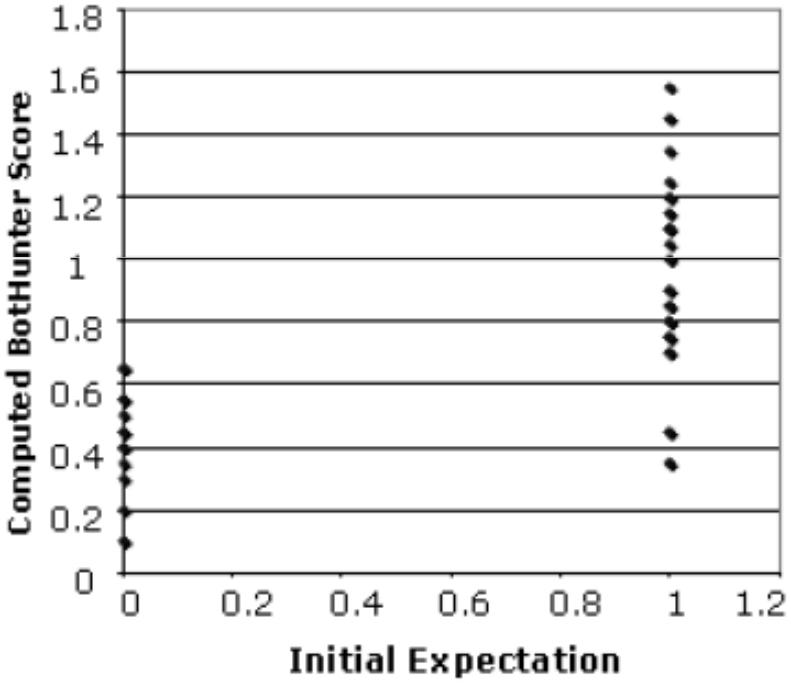

Figure 5: Scoring Plot from Expectation Table

BotHunter (7)

- Once a bot is detected, BotHunter produces a *bot profile* based on the dialogs, recording details such as:
  - IP address of the infected target
  - Infector list
  - Possible C&C server

Evaluation (1)

- Experiment 1: *In situ* virtual network composed of three Vmware guest systems:
  - Linux machine with IRC server installed, used as the C&C server
  - two Windows 2000 systems, one as the initial infected host, and the other as the next victim

Table 3: Dialog Summary of Virtual Network Infections

	E1	E2[rb]	E2[sl]	E3	E4	E5
agobot3-priv4	Yes(2/2)	Yes(9/8)	Yes(6/6)	Yes(5)	Yes(38/8)	Yes(4/1)
phat-alpha5	Yes(14/4)	Yes(5,785/5,721)	Yes(6/2)	Yes(3/3)	Yes(28/26)	Yes(4/2)
phatbot-rls	Yes(11/3)	Yes(2,834/46)	Yes(6/2)	Yes(8/8)	Yes(69/20)	Yes(6/2)
rbot0.6.6	No(0)	Yes(2/1)	Yes(2/1)	Yes(2/2)	Yes(65/24)	Yes(2/1)
rxbot7.5	No(0)	Yes(2/2)	Yes(2/2)	Yes(2/2)	Yes(70/27)	Yes(2/1)
rx-asn-2-re-workedv2	No(0)	Yes(4/3)	Yes(3/2)	Yes(2/2)	Yes(59/18)	Yes(2/1)
Rxbot-ak-0.7-Modded.by.Uncanny	No(0)	Yes(3/2)	Yes(3/2)	Yes(2/2)	Yes(73/26)	Yes(2/1)
sxtbot6.5	No(0)	Yes(3/2)	Yes(3/2)	Yes(2/2)	Yes(65/24)	Yes(2/1)
Urx-Special-Ed-UltrA-2005	No(0)	Yes(3/2)	Yes(3/2)	Yes(2/2)	Yes(68/22)	Yes(2/1)
gt-with-dcom-profile1	No(1/0)	Yes(5/3)	Yes(6/2)	No(0)	Yes(221/1)	No(4/0)
gt-with-dcom-profile2	No(1/0)	No(5/0)	No(6/0)	No(0)	Yes(221/44)	Yes(4/2)
gt-with-dcom-10min-profile	No(1/0)	Yes(5/3)	Yes(6/3)	No(0)	Yes(221/51)	Yes(4/2)

Evaluation (2)

- Experiment 2: SRI Honeynet
  - During a 3-week period, BotHunter detected a total of 1,920 of 2,019 successful bot infections, representing 95.1% detection rate
- The authors manually examined the remaining 4.9% (false negative) and classified the reasons into three categories:
  - Infection failures: Infections that lead to instability and eventual failure in the infected host
  - Honeynet setup and policy failures: The NAT mechanism did not correctly translate application-level address requests
  - Data corruption failures: This is the main reason (86% of the failed traces) in preventing the BotHunter's sensors from producing dialog warnings

Evaluation (3)

- Experiment 3: An example detection in a live deployment
  - BotHunter is deployed in Georgia Tech network
  - In Feb 07, BotHunter detected a bot infection that produced E1, E4 and E5 dialog warnings
  - Although the actual infection event (E2) was not detected, the target of the E4 warning (C&C server) was an address that was blacklisted as a known C&C server

Evaluation (4)

- Experiment 4: University Campus Network
  - BotHunter is deployed in Georgia Tech's College of Computing network
  - The monitored link has a traffic of over 100 Mbps during the day, composing of diverse protocols which share similarities with infection dialog
  - Bot traffic is injected into the network
    - BotHunter detected all 10 of 10 injected infections
  - A longer-term (4 month) evaluation is also run, to give an upper bound to the number of false positives
    - 98 profiles were generated, representing less than one false alarm per day

Evaluation (5)

- Experiment 5: Institutional Laboratory
  - BotHunter is deployed in a small well-administered production network
  - The goal of this experiment is to obtain the false positive rate
  - During the 10-day test, BotHunter sensors produced 5,501 dialog warnings. However, BotHunter correlator only produced one bot profile
  - The session that produced the false positive was actually a 1.6 GB multifile FTP transfer, with some of the files' content matching buffer overflow detection patterns

Conclusions

- BotHunter provides a real-time monitoring system to detect bots, with good detection and false positive rates
- Adversaries can evade BotHunter's detection scheme
  - using encrypted communication channels for C&C,
  - using more stealthy scanning techniques,
  - lying dormant for some time before moving to the next step, etc.
- However, many of the existing systems remain unprotected. The adversaries do not need to innovate yet