personal view on the future of
euro-day Worm Containment

Herbert Bos
Vrije Universiteit Amsterdam
herbertb_AT_cs.vu.nl
http://www.ist-lobster.org/
What is LOBSTER?

- FP6 Specific Support Activity (SSA)
- Duration: 09/2004 – 12/06
- Partners
  - FORTH
  - Vrije Universiteit Amsterdam
  - TNO ICT
  - CESNET
  - UNINETT
  - FORTHnet
  - ALCATEL
  - TERENA
  - Symantec?
What is LOBSTER?

- European Infrastructure for accurate network monitoring
- Allows one to perform pan-European monitoring
  - across organisations
- High-speed
  - specialised network cards
  - also: common NICs
- Why?
  - traffic classification
  - security
    - worms
    - DDoS
  - performance
  - billing
  - management

Herbert Bos, VU, http://www.cs.vu.nl/~herbert
Privacy

• a shared monitoring infrastructure
  ➔ what about privacy?!

Herbert Bos, VU, http://www.cs.vu.nl/~herbertb
What is LOBSTER?

- Data owners control
  - which users may access which data
  - very flexible
• **Intrusion Detection**
  – Are any of my computers compromised?
  – Is there any attacker trying to intrude into my network?

• **Large-scale Attack Detection – Detection of Epidemics**
  – DoS Attack detection (e.g., detect sharp increases in TCP/SYN packets)
  – Zero-day worm detection
    • e.g., detect lots of identical packets, never seen before, from several sources to several destinations
    • e.g., unusual no. of connections from a single port to unique destinations
    • e.g., detect worm characteristics
      – such as NOP sleds: long sequences of executable code

• **Network Telescopes**
  – monitor unused IP addresses
  – observe victims of DoS attacks
    • “back-scatter” traffic
  – observe infected hosts
  – port scans

*Herbert Bos, VU, http://www.cs.vu.nl/~herbertb*
Zero-day worm containment

• Why do we need it?
  – detect something new is on the loose
  – worms spread too fast for human intervention

• Different worms in different forms
  – fast ↔ slow
  – polymorphic ↔ immutable
  – wide spread ↔ narrow spread
  – stealth ↔ plain
  – multi-vector ↔ uni-vector

• Worm structure
  exploit  payload

Herbert Bos, VU, http://www.cs.vu.nl/~herbertb
Two tasks

• **Spot the bad guys**
  – network-based
    • content-based: EarlyBird
    • flow-based: VirusThrottling
  – host-based
    • honeypots
    • end-users (systrace)

• **Stop them!**
  – filters for networks
    • snort
    • VirusThrottle
  – filters for hosts
    • Self-Certifying Alerts

- can be fast (certainly flow-based)
- protects many hosts
- handles polymorphism
- can be very accurate (no false positives)
- may handle polymorphism
- handles polymorphism
- protects many hosts
- polymorphism
- few false positives
- some polymorphism?
Two tasks

- **Spot the bad guys**
  - network-based
    - content-based: EarlyBird
    - flow-based: VirusThrottling
  - host-based
    - honeypots
    - end-users (systrace)

- **Stop them!**
  - filters for networks
    - snort
    - VirusThrottle
  - filters for hosts
    - Self-Certifying Alerts

- false positives
- what to do with encryption?
- false positives
- slow
- needs a certain amount of luck
- need real services for accuracy
- false positives
- encryption/polymorphism will kill us
- false positives
- pretty slow
- can we rely on end-users?

Herbert Bos, VU, [http://www.cs.vu.nl/~herbertb](http://www.cs.vu.nl/~herbertb)
My conclusion (1/4)

- **detection**
  - **network-based**
    - behaviour-based
      - first indication
    - content-based:
      - weed out known and old threats
      - first indication for new threats
  - **host-based**
    - inaccurate behaviour based: first indication
    - accurate behaviour based:
      - zero-day detection
      - verification
    - should not handle full streams
My conclusion (2/4)

• **blocking**
  – **network-based**
    • behaviour-based:
      – no (unless exceptional circumstances)
    • content-based:
      – weed out known and old threats
      – first indication for new threats
  – **host-based**
    • good place for filtering, but scope of protection limited
    • end-host, so filtering should be fairly efficient
My conclusion (3/4)

- future of network-based content inspection for zero-day worm detection
• passive monitoring still needed, but role is changing
  – redirect traffic
  – sample traffic
  – first-pass detection
  – first-pass filtering
  – behaviour-based detection

• explore
  – multi-tier detection
  – multi-tier filtering
  – integrated approaches
  – cocktail-drugs for Internet diseases?
Fingerprinting zero-day attacks and using advertised honeypots

(or: guarding the heifer without falling asleep)
Argos Overview

• Platform for next generation honeypots
  – High-interaction, advertised, safe

• Detection of most common vulnerabilities
  – Control, code injection, function argument attacks

• Emulate + protect entire PC systems
  – OS agnostic, run on commodity hardware

• Generate host and network intrusion prevention signatures
  – Protect even uncooperative users

• Joint development with Dutch DeWorm project (VU)

Herbert Bos, VU, http://www.cs.vu.nl/~herbertb
Argos Overview

Applications

Forensics

Guest OS

Argos emulator

Detect attack and log state

Host OS

Signature
correlation

correlation

NIC

Log

Correlate data

Signature post-processing

Herbert Bos, VU, http://www.cs.vu.nl/~herbertb
Argos Overview

- Applications
  - Forensics
  - Guest OS
  - Argos emulator

- NIC
- Detect attack and log state
- Log
- Correlate data
- Signature
- Signature post-processing

Herbert Bos, VU, http://www.cs.vu.nl/~herbertb
Development up to Present

- Based on the Qemu emulator
- Track network data throughout execution
- Detect illegal uses of network data
  - Jump targets, function pointers, instructions, system call arguments
- Forensics to generate signatures
  - Export emulator state, inject “forensics” shellcode

Herbert Bos, VU, http://www.cs.vu.nl/~herbertb
• Tagging network data as “tainted”
Network Data Tracking

- Tagging network data as “tainted”
- Tracking “tainted” data
  - ALU operations

```
ADD EAX, EBX
```
Network Data Tracking

- Tagging network data as “tainted”
- Tracking “tainted” data
  - ALU operations

```
ADD EAX, EBX
XOR EBX, EBX
```
Network Data Tracking

- Tagging network data as “tainted”
- Tracking “tainted” data
  - ALU operations
  - MMU operations

```
ADD EAX, EBX
XOR EBX, EBX
ST A, EAX
```
Identifying Attacks

- Jump targets

![Diagram showing EAX, EBX, ECX, EDX registers and ALERT symbol]
Identifying Attacks

- Jump targets
- Function calls

EAX  EBX  ECX  EDX
JMP EAX
CALL EAX

Alert

Herbert Bos, VU, http://www.cs.vu.nl/~herbertb
Identifying Attacks

- Jump targets
- Function calls
- Returns

EAX  EBX  ECX  EDX
JMP EAX
CALL EAX
RET

RAM

STACK

 ALERT

Herbert Bos, VU, http://www.cs.vu.nl/~herbertb
Identifying Attacks

- Jump targets
- Function calls
- Returns
- Code injection

EAX  EBX  ECX  EDX
JMP EAX
CALL EAX
RET
JMP A

RAM
STACK

ALERT

Herbert Bos, VU, http://www.cs.vu.nl/~herbertb
Identifying Attacks

- Jump targets
- Function calls
- Returns
- Code injection
- System calls

EAX | EBX | ECX | EDX
---|---|---|---
JMP EAX
CALL EAX
RET
JMP A
INT 0x80

RAM
STACK

ALERT

Herbert Bos, VU, http://www.cs.vu.nl/~herbertb
Forensics

• Emulator state (registers, “tainted” memory)
• Injected shellcode data
  – Process information (e.g. PID)
  – Extraction of probable target port PID → Name → Port
• Network trace
Signature Generation

Alert value  Forensics data  Network streams

Origin of value in memory

“tainted” memory data

Signature

Herbert Bos, VU, http://www.cs.vu.nl/~herbertb
Argos: an Emulator for Capturing Zero-Day Attacks

Argos is a full system emulator that implements secure extensions, which protect it from being compromised by most of the known vulnerabilities. It is based on the QEMU open source emulator, which employs dynamic translation to achieve attractive emulation speed.

The incentive behind Argos is to create a framework for honeypots that is both secure and robust, to identify zero-day attacks of worms and other similar malicious software. Eventually, we aim to produce a system that will automatically produce remedies for such attacks by generating appropriate vaccines (e.g. intrusion prevention signatures).

To identify attacks we employ dynamic taint analysis. This involves tracking data originating from the network during execution, and raising an alarm whenever an illegal use of such data is detected. For example the use of network data as a jump target, instruction or critical system call argument.

Currently, Argos can be run in any little-endian CPU and any OS supported by QEMU. Emulated systems can be either x86 or x86_64. We are looking for people that would be interested in deploying Argos in their network, so if you are one of them do not hesitate to contact us.

Copyright © 2005 by Georgios Portokalidis.

http://www.few.vu.nl/argos