The NoAH approach to zero-day worm detection

Asia Slowinska
Vrije Universiteit, Amsterdam
What is NoAH?

~ NoAH is a Specific Support Action in the Sixth Framework Programme of the European Union

~ Start: 1\textsuperscript{st} April 2005

~ End: 31\textsuperscript{st} March 2008

~ Homepage: http://www.fp6-noah.org/
Why?

- Too many vulnerabilities
- New worm attacks
- Human intervention too slow
- Current solutions are not perfect
- Time consuming
- Inaccurate

CERT/CC Reported Vulnerabilities

Asia Slowinska, Vrije Universiteit
Why?

~ Too many vulnerabilities
~ New worm attacks
~ Human intervention too slow
~ Current solutions are not perfect
  ~ Time consuming
  ~ Inaccurate
Why?

∼ Too many vulnerabilities
∼ New worm attacks
∼ Human intervention too slow
∼ Current solutions are not perfect
∼ Time consuming
∼ Inaccurate
Why?

~ Too many vulnerabilities
~ New worm attacks
~ Human intervention too slow
~ Current solutions are not perfect
  ~ Time consuming
  ~ Inaccurate
Goals

~ Design and develop infrastructure for security monitoring based on honeypots technology

~ Detect most common attack vectors
  ~ Detect worms in early stage of spreading

~ Gather information about attacks

~ Generate signatures
Honeypots

~ A computer system acting as a decoy
~ does not provide regular services
~ lures in potential hackers in order to study their activities
~ honeypots in NoAH listen to unused IP address space, called further dark space

~ Two basic types of honeypots
~ low interaction (LI) – emulate services
~ high interaction (HI) – run real applications
NoAH architecture
NoAH architecture

Internet

Participating Organization

Funnel

Tunnel

NoAH Core

LI honeypot

HI honeypot

Asia Slowinska, Vrije Universiteit
Core organizations

~ host NoAH honeypots

~ problem: wide dark address space we could monitor vs limited number of honeypots
Core organizations

- host NoAH honeypots
- problem: wide dark address space we could monitor vs limited number of honeypots
- solution: funelling
Core organizations

~ host NoAH honeypots

~ problem: wide dark address space we could monitor vs limited number of honeypots

~ solution: funelling
Core organizations

~ host NoAH honeypots

~ problem: wide dark address space we could monitor vs limited number of honeypots

~ solution: funnelling
Core organizations

- host NoAH honeypots
- problem: wide dark address space we could monitor vs limited number of honeypots
- solution: funelling
Cooperating organizations

- do not maintain NoAH honeypots
- traffic arriving at the dark space is redirected to the NoAH core
- install and run funnel component only
Honey@home

~ homes or small offices

~ a honeypot daemon running in the background

~ easy to install

~ dark space

~ unused IP addresses

~ unused TCP/UDP ports (or a subset of them)

~ forwards all traffic for the dark space to the NoAH core via an anonymous path

Asia Slowinska, Vrije Universiteit
Cooperation between LI and HI honeypots

Attacker

Low-interaction honeypots
honeyd instance

honeyd instance

Attacker

High-interaction honeypots
Windows XP
IIS v2.0

RedHat Linux 9.0
SSH server v1.7
SMTP server

RedHat Linux 9.0
SSH server v2.4
Apache v1.3

Windows 2000
Oracle 10g
High-interaction honeypots in NoAH – Argos

- based on Qemu, an emulator
- protects multiple OSs and all applications without modification
- employs dynamic taint analysis
- detects attacks that divert conventional control flow, e.g., exploits for buffer overflows, format strings, and double-free vulnerabilities
Argos design

Applications

Guest OS

Argos Emulator

Host OS

Asia Slowinska, Vrije Universiteit
Argos design

- Log
- Host OS
- Argos Emulator
- Guest OS
- Applications

Asia Slowinska, Vrije Universiteit
Argos design

Applications

Guest OS

Argos Emulator

Host OS

Log

Asia Slowinska, Vrije Universiteit
**Dynamic taint analysis:**
- tag data coming from the network as tainted
- track tainted data during execution
- identify and prevent usage of tainted data
Argos design

Applications

Guest OS

Argos Emulator

Host OS

Log

Asia Slowinska, Vrije Universiteit
Argos design

Applications

Guest OS

Argos Emulator

Host OS

Log

Asia Slowinska, Vrije Universiteit
Argos design

Applications

Guest OS

Argos Emulator

Host OS

Log

Signature and information about attack

Asia Slowinska, Vrije Universiteit
Forensics in Argos

Applications

Guest OS

Argos Emulator

Host OS

Log

Asia Slowinska, Vrije Universiteit
Forensics in Argos

Forensics

Guest OS

Argos Emulator

Applications

Host OS

Log

Signature, information about attack and additional information about the vulnerable application

Asia Slowinska, Vrije Universiteit
Attack detection

~ redirect control flow

~ program counter must be loaded with a tainted value
~ keep track of call, jmp and ret instructions
~ check that the value loaded in program counter is not tainted

~ code-injection attacks

~ format string attacks do not overwrite program counter with a tainted value
~ check that the memory pointed by the value loaded in program counter is not tainted
Argos: simple example

int main(int argc, char **argv)
{
    if (argc > 1) read_url(argv[1]);
    return 1;
}

int read_url(char *request)
{
    char url[100];
    if (!strncmp(request, "GET ", 4))
        strcpy(url, request + 4);
    return 1;
}
Argos: simple example

```c
int main(int argc, char **argv)
{
    if (argc > 1) read_url(argv[1]);
    return 1;
}

int read_url(char *request)
{
    char url[100];
    if (!strncmp(request, "GET ", 4))
        strcpy(url, request + 4);
    return 1;
}

./test GET A....AAAAA<nasty_address>\0
100 chars
```
int main(int argc, char **argv)
{
    if (argc > 1) read_url(argv[1]);
    return 1;
}

int read_url(char *request)
{
    char url[100];
    if (!strncmp(request, "GET ", 4))
        strcpy(url, request + 4);
    return 1;
}

./test GET A....AAAAA<nasty_address>\0
100 chars

Asia Slowinska, Vrije Universiteit
int main(int argc, char **argv)
{
    if (argc > 1) read_url(argv[1]);
    return 1;
}

int read_url(char *request)
{
    char url[100];
    if (!strncmp(request, "GET ", 4))
        strcpy(url, request + 4);
    return 1;
}

./test GET A....AAAAA<nasty_address>
100 chars
int main(int argc, char **argv)
{
    if (argc > 1) read_url(argv[1]);
    return 1;
}

int read_url(char *request)
{
    char url[100];
    if (!strncmp(request, "GET ", 4))
        strcpy(url, request + 4);
    return 1;
}

./test GET A....AAAAA<nasty_address>\0
  100 chars
int main(int argc, char **argv)
{
    if (argc > 1) read_url(argv[1]);
    return 1;
}

int read_url(char *request)
{
    char url[100];
    if (!strncmp(request, "GET ", 4))
        strcpy(url, request + 4);
    return 1;
}

./test GET A....AAAAA<nasty_address>\0
100 chars

Asia Slowinska, Vrije Universiteit
**Argos: simple example**

```c
int main(int argc, char **argv)
{
    if (argc > 1) read_url(argv[1]);
    return 1;
}

int read_url(char *request)
{
    char url[100];
    if (!strncmp(request, "GET ", 4))
    {
        strcpy(url, request + 4);
        return 1;
    }
}
```

```
./test GET A....AAAAA<nasty_address>
100 chars
```

Asia Slowinska, Vrije Universiteit
int main(int argc, char **argv)
{
    if (argc > 1) read_url(argv[1]);
    return 1;
}

int read_url(char *request)
{
    char url[100];
    if (!strncmp(request, "GET ", 4))
        strcpy(url, request + 4);
    return 1;
}

./test GET A....AAAAA<nasty_address>\0

100 chars
int main(int argc, char **argv) {
    if (argc > 1) read_url(argv[1]);
    return 1;
}

int read_url(char *request) {
    char url[100];
    if (!strncmp(request, "GET ", 4))
        strcpy(url, request + 4);
    return 1;
}

./test GET A....AAAAA<nasty_address>\0
100 chars
int main(int argc, char **argv)
{
    if (argc > 1) read_url(argv[1]);
    return 1;
}

int read_url(char *request)
{
    char url[100];
    if (!strncmp(request, "GET ", 4))
        strcpy(url, request + 4);
    return 1;
}

./test GET A....AAAAA<nasty_address>
100 chars
# Security evaluation

<table>
<thead>
<tr>
<th>Vulnerability</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache chunked encoding overflow</td>
<td></td>
</tr>
<tr>
<td>IIS ISAPI .printer host header overflow</td>
<td></td>
</tr>
<tr>
<td>WebDav ntdll.dll overflow</td>
<td></td>
</tr>
<tr>
<td>FrontPage Server Extensions Debug Overflow</td>
<td></td>
</tr>
<tr>
<td>War-FTP overflow</td>
<td></td>
</tr>
<tr>
<td>ASN.1 Library Bitstring Heap Overflow</td>
<td></td>
</tr>
<tr>
<td>Windows Message Queueing Remote Overflow</td>
<td></td>
</tr>
<tr>
<td>RPC DCOM Interface overflow</td>
<td></td>
</tr>
<tr>
<td>LSASS Overflow</td>
<td></td>
</tr>
<tr>
<td>Windows PnP Service Remote Overflow</td>
<td></td>
</tr>
<tr>
<td>nbSMTP remote format string exploit</td>
<td></td>
</tr>
<tr>
<td>WMF exploit</td>
<td></td>
</tr>
<tr>
<td>Microsoft net API overflow MS06-040</td>
<td></td>
</tr>
</tbody>
</table>
Performance

![Bar chart comparing QEMU and Argos performance across nbench integer, nbench float, and nbench memory benchmarks.](image)

Asia Slowinska, Vrije Universiteit
Welcome to NoAH

NoAH is a Specific Support Action in the Sixth Framework Programme of the European Union.

The project aims to gather and analyse information about the nature of Internet cyberattacks. It will also develop an infrastructure to detect and provide early warning of such attacks, so that appropriate countermeasures may be taken to combat them.

Latest News

- The 1st NoAH workshop organised at the TNC 2006 conference in Catania, Italy.
- First NoAH workshop on honeypots at TNC 2006.
-分布式Honeypot Project survey reaffirms NoAH conclusions.
- Argos secure system emulator source code available.
- Argos survey on honeypot requirements summarised.
- Use of Shadow Honeypots presentation available.

What is Argos?

Argos is a full and secure system emulator designed for use in honeypots. It is based on Qemu, an open source emulator that uses dynamic translation to achieve a fairly good emulation speed.

Argos extends Qemu to enable it to detect remote attempts to compromise the emulated guest operating system. Using dynamic taint analysis it tracks network data throughout execution and detects any attempts to use them in an illegal way. When an attack is detected the memory footprint of the attack is logged.

Argos is the first step to create a framework that will use next generation honeypots to automatically identify and produce remedies for zero-day worms, and other similar attacks. Next generation honeypots should not require the honeypots IP remains unadvertised. On the contrary, it should attempt to publicise its service and even actively generate traffic. In formal honeypots, this was often impossible, because malevolent and benevolent traffic could not be distinguished. Since Argos is explicitly signalling each possibly successful exploit attempt, we are now able to differentiate malicious from innocuous traffic.

GET ARGOS

Download Argos
NoAH partners

~ Foundation of Research and Technology (FORTH), Heraklion, Greece – coordinator
~ Vrije Universiteit, Amsterdam, The Netherlands
~ ETH, Zurich, Switzerland
~ TERENA, Amsterdam, The Netherlands
~ FORTHnet SA, Heraklion, Greece
~ DFN-CERT, Hamburg, Germany
~ Virtual Trip Limited, Greece
~ ALCATEL, France
Backup slides
Funneling

~ arpd to collect IP addresses

~ user-space daemon that responds to ARP requests arriving to the network interface of the honeypot

~ honeyd handles traffic arriving at honeypots

~ funneling has no overhead

~ we tested emulating /24, /16, and /8 subnets without any noticeable difference in performance
Tunneling

- OpenVPN 2.0 as tunnel software
- Encrypted channel, supports packet compression
Honey@home - challenges

- We cannot trust clients
  - Anyone will be able to set up honey@home

- Clients must not know the address of honeypot
  - Honeypots may become victims of flooding

- Address of client must also remain hidden
  - Attacker can use their black space for flooding

- Computer-based mass installation of honey@home mockup client should be prevented
Network Data Tracking

Reg B = network_read

Registers
Network Data Tracking

Reg B = network_read

Registers
Network Data Tracking

Reg B = network_read

Reg A = Reg A + Reg B
Network Data Tracking

Reg B = network_read

Reg A = Reg A + Reg B
Network Data Tracking

Reg B = network_read

Reg A = Reg A + Reg B

Memory(B) = Reg B
Reg B = network_read

Reg A = Reg A + Reg B

Memory(B) = Reg B
Network Data Tracking

Reg B = network_read

Reg A = Reg A + Reg B

Memory(B) = Reg B

Reg B = Reg A / 12.34 (Sanitise data)
Network Data Tracking

Reg B = network_read

Reg A = Reg A + Reg B

Memory(B) = Reg B

Reg B = Reg A / 12.34 (Sanitise data)
Guest forensics

Virtual address space

Applications

Guest OS

Argos Emulator

Snitch

Process name
Linked libraries
Open Ports

Virtual address space

Registers

Memory

Asia Slowinska, Vrije Universiteit
Forensics shellcode injection

- lookup process's read-only pages
- inject code at last text segment page
- point EIP to shellcode
<table>
<thead>
<tr>
<th>Forensics</th>
<th>Snitch</th>
</tr>
</thead>
<tbody>
<tr>
<td>~ pid = getpid()</td>
<td>~ listen()</td>
</tr>
<tr>
<td>~ connect(localhost)</td>
<td>~ accept()</td>
</tr>
<tr>
<td>~ send(pid)</td>
<td>~ read(pid)</td>
</tr>
<tr>
<td></td>
<td>~ exec(netstat or OpenPorts)</td>
</tr>
<tr>
<td></td>
<td>~ connect(argos host)</td>
</tr>
<tr>
<td></td>
<td>~ send(info)</td>
</tr>
</tbody>
</table>
Network tracking

RAM

Reg A
Reg B

Network trace

Network flows

Asia Slowinska, Vrije Universiteit