Argos Emulator

Georgios Portokalidis
Asia Slowinska
Herbert Bos

Vrije Universiteit Amsterdam
Why?

- Too many vulnerabilities
- New worm attacks
- Human intervention too slow
- Current solutions are problematic
  - Time consuming
  - Inaccurate
Goals

- Platform for next generation honeypots
- Protect entire OS
- Detect most common attack vectors
- Accuracy
It Works!

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

Herbert Bos  VU Amsterdam
Argos Overview

- Forensics
- Argos Emulator
- Guest OS
- Applications
- Snitch
- Host OS
- Log
- Signature
- Post-Processing Sub-system

Herbert Bos

VU Amsterdam
Network Data Tracking

Register = network_read


Memory(A) = Reg. A

Reg. B = Reg. A / 156.345
Capturing Attacks

- Diverting control flow
- Executing arbitrary instructions
- Overwriting system call arguments

Diagram:
- Tainted Register Operands
- Tainted Memory
- JMP CALL
- RET
- SYSCALL

Herbert Bos
VU Amsterdam
Forensics

Virtual Address Space

Applications
- Process name
- Linked Libraries
- Open Ports

Guest OS

Argos Emulator

Registers

RAM

Herbert Bos VU Amsterdam
Signature Generation
Multiple Signature Generators
Signature Generation I

Logged Network Flows

Argos Memory Log

Critical Exploit Bytes
(e.g. value loaded on EIP)

New Signature

Similar Signatures

Generalised Signature
Multiple fields may have contributed to the overflow
Signature Generator II

- handles polymorphic attacks
- easy to check
- great when considering false positives
Emulator Performance

Overhead ($y$ times slower)

- Vanilla Qemu
- Argos

Herbert Bos

VU Amsterdam
Signature Generation Performance

Time to generate signature (sec)

Tcpdump trace size (MB)
Future Work

- Replaying attacks
- Integration with nepenthes honeypot
- Increase data tracking precision
- Protocol aware signature generation
- Generate self certifying alerts
On The Web

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

Argos: an Emulator for Capturing Zero-Day Attacks

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

We have extended QEMU to enable it to detect remote attempts to compromise the emulated guest operating system. Using dynamic taint analysis, Argos tracks network data throughout the processor's execution and detects any attempts to use them in a malicious way. When an attack is detected the memory footprint of the attack is logged and the emulators exits.

Argos is the first step to create a framework that will use next generation honeyhops to automatically identify and produce remedies for zero-day worms, and other similar attacks. Most generation honeyhops should require that the honeyhop's IP address remains un-advertised. On the contrary, it should attempt to publicise its services and even actively generate traffic. In former honeyhops this was often impossible, because malicious and benevolent traffic could not be distinguished. Since Argos is explicitly signaling each possibly successful exploit attempt, we are now able to differentiate malicious attacks and innocuous traffic.

Argos has moved to the local gForge. You can pickup the code and documentation there.
Network Data Tracking

- Tag network data as “tainted”
Network Data Tracking

- Tag network data as “tainted”
- Track “tainted” data propagation
  - Arithmetic, logical operations
  - Memory operations
Network Data Tracking

- Tag network data as “tainted”
- Track “tainted” data propagation
  - Arithmetic, logical operations
  - Memory operations
- Sanitise data
  - Floating point, SSE
Identifying Attacks

- Jumps
- Function calls
- Function returns
- System calls

```
EAX  EBX  ECX  EDX  RAM
JMP EAX
CALL EAX
RET
JMP A
INT 0x80
```
SweetBait Design
## Logs Format

<table>
<thead>
<tr>
<th>Format</th>
<th>Type</th>
<th>RID</th>
<th>Timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Register values</td>
<td></td>
<td></td>
<td>Register tags</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EIP value</td>
<td>EIP origin</td>
<td>EFLAGS</td>
<td></td>
</tr>
<tr>
<td>Memory Block Contents</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Format</th>
<th>Tainted Flag</th>
<th>Size</th>
<th>P. Address</th>
<th>V. Address</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Herbert Bos

VU Amsterdam
Forensics Shellcode Injection

- Lookup process’s read-only pages
- Inject code at last text segment page
- Point EIP to shellcode

Process Address Space

Herbert Bos  VU Amsterdam
Forensics – The Snitch

- Pid = getpid()
- Rid [injected by Argos]
- Connect(localhost)
- Send(pid & rid)

- Listen()
- Accept()
- Read(pid & rid)
- Exec(Netstat or OpenPorts)
- Connect(argos host)
- Send(info)