Vigilante and Potemkin

The Wild Wild Web

Presenter: Ýmir Vigfússon
Based in part on slide sets from Mahesh Balakrishnan and Raghavan Sriraman.

Security vulnerabilities

- Lazy programmers
- Bad programmers
- BIND, Sendmail, WU-FTP
- Buffer overflows
- Format string attacks
- Integer overflows
- Race conditions
- Command injection
- …

Simple buffer overflow

Parameters
Return address
Stack Frame Pointer
Local variables

Stack Frame Pointer
SP
Parameters
Return address
Local variables

Code on a server (written by a lazy programmer):

```c
void func(char *str) {
    char buf[128];
    strcpy(buf, str);
    do something(buf);
}
```

When the function is invoked the stack looks like:

```
buf  do  ret addr  top of stack
```

What if *str is 136 bytes long? After `strcpy()`:

```
*str  ret  addr  top of stack
```

Simple buffer overflow

```assembly
31 c0 b0 3f cd 80 31 db 89 d9
b1 02 31 c0 b0 3f cd 80 eb 1f
5e 89 76 08 31 c0 88 46 07 89
46 0c b0 0b 89 f3 8d 4e 08 8d
56 0c cd 80 31 db 89 d8 40 cd
80 e8 dc ff ff /sbin/eject
exec("*/sbi n/ej ect");
```

Worms

11/1988
Cornell grad student Robert Morris writes the Internet Worm
Worms

Cornell grad student Robert Morris writes the Internet Worm

- CodeRed – MS IIS
- Nimda – MS IIS+email
- Slammer – MS SQL
- Blaster – MS Win RPC
- Sasser – MS Win LSASS
- Zotob – MS Win Plug-n-Play

(more to come)

Worms

- CodeRed – MS IIS
- Slammer – MS SQL
- Blaster – MS Win RPC

Infected 2x

- CodeRed 360,000 37m
- Slammer 75,000 8.5s
- Blaster 500,000 37m

Vigilante

- Manuel Costa, Jon Crowcroft, R. IgalCabri, Antony Rowstron, Lidong Zhou, Lintao Zhang, Paul Barham

- Automates worm defense
  - Run heavily instrumented versions of software on detector machines
  - Uses collaborative infrastructure to detect worms

Overview

SCA: Self-Certifying Alert

Dynamic dataflow analysis

Dirty data loaded into PC → Execution control vuln.

Dirty data to be executed → Code execution vuln.

Critical fn. argument dirty → Function argument vuln.

- Specific to C/ C++ vulnerabilities
- E-mail? Format string attacks?

SCA generation

buf | dsp | ret addr | top of stack
SCA generation

Example: The Slammer Worm

AHA!

SCA verification

- Hosts run same software with identical configuration within sandbox
- Replace code/address in SCA with a call to verified()

No trust required
Fast, simple and generic verification
No false positives

SCA distribution

- Flooding over secure Pastry overlay
- Denial-of-Service attacks? (DoS)
  - Don't forward already blocked SCAs
  - Forward only after verification
  - Rate-limit SCAs from each neighbor
  - Use super-peers so worms can’t learn the topology
Local response

- Verify SCA
- Data and Control Flow Analysis
- Generate filters – conjunctions of conditions on single messages
  - Two levels:
    - General filter with false positives
    - Specific filter with no false positives

Evaluation: SCA generation

- SCA Generation Time
- SCA Sizes

Evaluation: SCA verification

- Verification is fast. The sandbox VM is always running.

Simulation on real worms

- Simulate worm epidemic on 500,000 nodes, 1000 super-peers
- Includes worm-induced congestion
- DoS: Each host sends fake SCAs to all neighbors

Internet Dangers

- Honeypots
  - A honeypot is a network-connected system that is carefully monitored so that intrusions can be easily detected and precisely analyzed.

Honeypots

- Scalability
- Fidelity
- Containment
Honeypots

Low interaction
- High scaling
- Low fidelity

High interaction
- Low scaling
- High fidelity

Containment means that compromised honeypots should not be able to attack third-party systems.

A honeyfarm is a set of honeypots.

Potemkin Honeyfarm

"Dynamically bind physical resources to external requests only for the short periods of time necessary to emulate the execution behavior of dedicated hosts."

Potemkin: a honeyfarm system that exploits:
- Virtual machines
- Aggressive memory sharing
- Late binding of resources

Kijong Dong, N-Korea

(a modern Potemkin village)

Potemkin Honeyfarm

Potemkin

Catherine II

Potemkin Honeyfarm

"Dynamically bind physical resources to external requests only for the short periods of time necessary to emulate the execution behavior of dedicated hosts."

Potemkin: a honeyfarm system that exploits:
- Virtual machines
- Aggressive memory sharing
- Late binding of resources

Potemkin Architecture

Virtual Machine Monitors (VMMs)
- Easy to manage. Physical resources not a major restriction.
- Each IP address spawns a new VM.

Problem: Expensive

Observation: Targets are homogenous

Solution: clone a VM from a reference image, change IP (etc.), accept packets.
**Flash Cloning**

- Domain-0 Network Stack
  - Xen Data traffic
  - Internal swap
- Xen Management Daemon
  - "Clone VM"
  - "Reconfigure IP address to A"
- Cloned VM
  - Forwarded to Cloned VM
- Cloned VM’s response traffic

**Delta Virtualization**

- Guest A
  - Guest Physical Address
  - Machine Memory
- Guest B
  - Frame Table

**Potemkin Architecture**

- What if VMs are compromised?
- Gateway router policy:
  - Isolate the HoneyFarm, only send outgoing packets in response to incoming ones.
  - Other packets are internally reflected. Infections spread within HoneyFarm.
  - Universal identifier captures causal relationship of communication.
- Directs incoming traffic, contains outgoing traffic, resource management, user interface

**Figure 4:** Required number of VMs active in response to all measured traffic from a /10 network, when VMs are aggressively recycled after 200 milliseconds of inactivity. Traffic is from the one-hour period starting Monday, March 21, 2005 04:05 GMT.
Evaluation: Scan filter

Food for thought

- How can HoneyFarms attract traffic?
- HoneyPot detection
- DoS attacks

Questions?