Some Anti-Worm Efforts at Microsoft

Helen J. Wang
System and Networking Research Group
Microsoft Research
Oct 29, 2004

1

Acknowledgements

- Matt Braverman, Opher Dubrovsky, John Dunagan, Louis Lafreniere, Steve Lipner, Jon Pincus, Pat Stemen, Yi-Min Wang

2

Outline

- Product side:
  - Software Development Life cycle (SDL)
  - Compile-time solutions:
    - /GS compiler option
    - Static checking
  - Windows XP SP2
- Research side:
  - Shielding before patching (Shield, research)
  - System management research (Strider)

3

New MS Software Development Life cycle

- Training
- Requirement
  - Security at outset; security advisor, security milestone, exit criteria
- Design
  - Identify trusted base, minimize/document attack surface, secure default setting
- Development
  - Static checking, code review
- Verification
  - Beta, regression testing, code review, penetration testing, auto tool check,
- Release:
  - Final security review: 2-6 months before; go back to previous phases if necessary; additional (external) penetration testing
- Response:
  - Microsoft Security Response Center
  - Sustain Engineering Teams
  - Patch Management

4

Outline

- Product side:
  - Software Development Life cycle (SDL)
  - Compile-time solutions:
    - /GS compiler option
    - Static checking
  - Windows XP SP2
- Research side:
  - Shielding before patching (Shield, research)
  - System management research (Strider)

5

/GS Compiler Option

- Goal: defeat return address hijacking
- /GS
  - insert a cookie between the locally declared buffer and the return address
  - test cookie for corruption before using return address
  - If test fails, terminate the process
- Various challenges
  - Exception handler function pointer hijacking
  - User installable function pointer hijacking
  - Pointer subterfuge
    - hijacking local pointers or function parameters
    - Global cookie hijacking

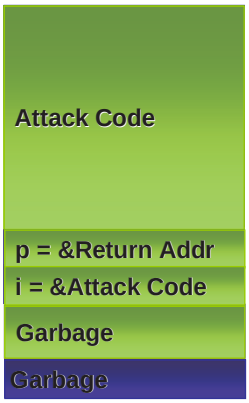
6

/GS Compiler Option: Trampoline (Pointer subterfuge)

2 stages attack

```
void vulnerable(
  char* buf, int cb)
{
  char name[8];
  int *p = &G;
  int i = value();

  memcpy(name, buf, cb);
  *p = i;
}
```



7

/GS Compiler Option, Cont.

- Mitigations
  - Reorder local variables to avoid local pointer hijacking
  - Shadow parameters as local variables to avoid function parameter hijacking
  - Safe Exception Handling (SEH):
    - OS detects invalid exception handlers
    - CRT detects corrupted SEH info table
  - Cookie protection:
    - Hiding the local cookie to mitigate global cookie hijacking: XOR (ESP, cookie)
    - Leading 0's for cookie to prevent "strcpy" buffer overruns
- Arms race

8

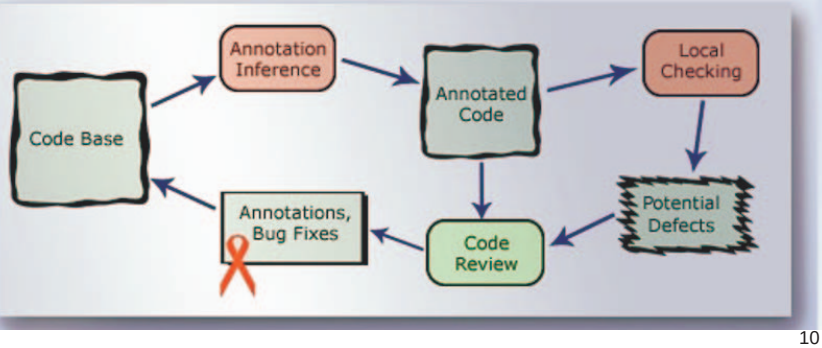
Static Checking

- MSR PPRC → MS CSE
  - Static checking for software defects such as buffer overflows, un-initialized data, resource leakage, etc.
  - Tools: espX
    - Use code annotation to enable effective local data flow and control flow analysis

9

espX Usage

(1) espX infers annotations, i.e., contracts between functions
(2) espX checks each function for buffer overruns
(3) Developers review reported defects



10

Outline

- Product side:
  - Software Development Life cycle (SDL)
  - Compile-time solutions:
    - /GS compiler option
    - Static checking
  - Windows XP SP2
- Research side:
  - Shielding before patching (Shield, research)
  - System management research (Strider)

11

Windows XP SP2: Securing the Network

- Windows firewall (ICF)
  - On by default
  - Stateful: automatically matching inbound traffic with outgoing requests
  - Boot time security
  - Limit the number of half open TCP connections to 10
  - Application affected: those listen for unsolicited traffic (e.g., file/printer sharing, uPnP, remote desktop, remote admin, ICMP options)
- RPC/DCOM
  - Reduce attack surface
  - Make it easier to restrict RPC interfaces to local machine
  - Block unauthenticated calls to DCOM and RPC services
- Attachments:
  - Unsafe attachments not trusted by default
  - Block/Prompt/Allow determined by combination of file type & zone
    - Dangerous file type + Restricted Zone = Block
    - Dangerous file type + Internet Zone = Prompt

12

Windows XP SP2: Memory Protection

- /GS:
  - Most critical components that take network or untrusted input have been recompiled
- NX:
  - Prevents execution of injected code
  - Leverages processor technology
    - Marks memory regions as non-executable
    - Processor raises exception when injected code is executed
  - Supported on 64-bit extensions processors
    - SP2 runs in 32-bit compatibility mode with NX support
  - On by default only for system components
    - User applications can be opted in
  - Some app compatibility issues

13

Outline

- Product side:
  - Software Development Life cycle (SDL)
  - Compile-time solutions:
    - /GS compiler option
    - Static checking
  - Windows XP SP2
- Research side:
  - Shielding before patching (Shield, research)
  - System management research (Strider)

14

Software patching *not* an effective first line worm defense

- Sasser, MSBlast, CodeRed, Slammer, Nimda, Slapper all exploited ***known*** vulnerabilities whose patches were released ***months or weeks*** before
- 90+% of worm attacks exploit known vulnerabilities [Arbaugh2002]
- People don't patch immediately

15

Why don't people patch?

- ***Disruption***
  - **Service or machine reboot**
- ***Unreliability***
  - **Software patches inherently hard to test**
- ***Irreversibility***
  - **Cannot always undo a patch**
- ***Unawareness***
  - **Automatic patch installation not possible**

16

Firewall also *not* an effective first line worm defense

- Traditional firewalls
  - Course-grained
    - High false positive rate
  - Typically in the network
    - One-size-fits-all solution, lack application-awareness, miss end-to-end encrypted traffic
- Exploit-driven firewalls
  - Filter according to exploit (attack) signatures
    - Attack code obfuscation, e.g., polymorphism, metamorphism, can evade the firewall
  - Worms spread fast (in minutes or seconds!)
    - Real-time signature generation and distribution difficult

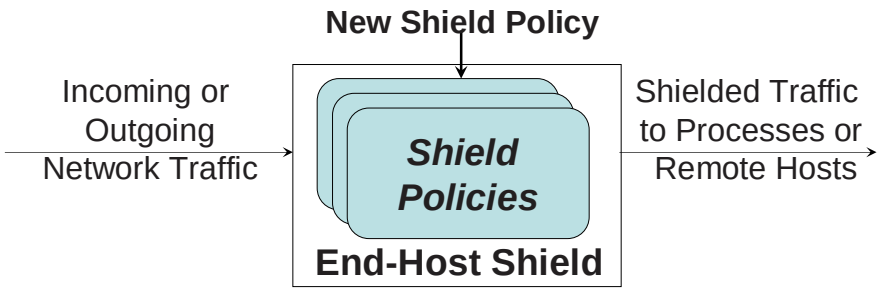
17

Shields: End-host Vulnerability-Driven Network Filters

- Goal: Protect the time window between ***vulnerability disclosure*** and ***patch application*****.**
- Approach: Characterize the vulnerability instead of its exploits and use the vulnerability signature for end-host firewalling
- Shields combine the best features of
  - Patches: ***vulnerability-specific***, code level, executable
  - Firewall: exploit-specific, ***network level***, ***data-driven***
- Advantages of Shield:
  - Protection as good as patches (resilient to attack variations), unlike exploit-driven firewalls
  - Easier to test and deploy, more reliable than patches

18

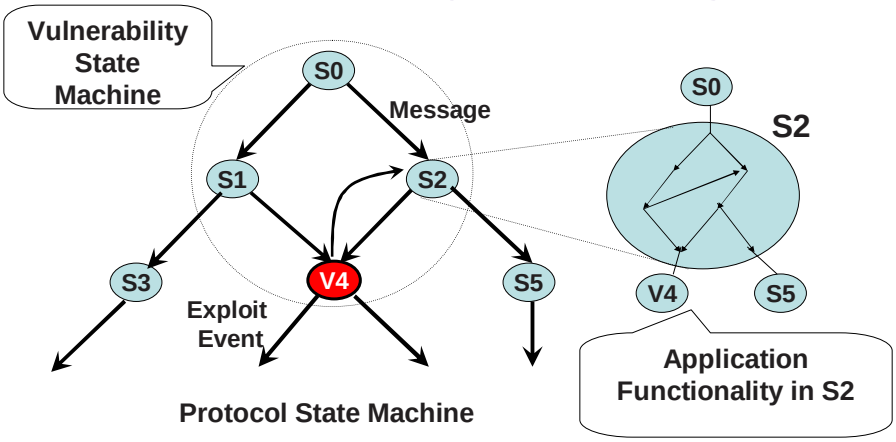
Overview of Shield Usage



- Shield intercepts vulnerable application traffic above the transport layer.
- Policy distribution very much like anti-virus signature model – automatic, non-disruptive, reversible

19

Vulnerability Modeling



Shield Policy (Vulnerability Signature):

Vulnerability state machine + how to recognize and react to exploits in the vulnerable state

20

Shield Implementation and Evaluation

- Prototype implemented as Windows Layered Service Provider (LSP)
  - Uses Generic Protocol Analyzer
  - Working shields for vulnerabilities behind Blaster, Slammer, and CodeRed
  - Performance and scalability results promising:
    - Negligible overhead for end user machines
    - 14-30% throughput overhead for an artificial scenario stressing Shield
- MSRC 2003 Bulletin study
  - All 12 worm-able vulnerabilities are easily shield-able
  - Some of the other 37 may also be shield-able

21

Ongoing Work

- Generic protocol analyzer (GPA):
  - Implements common elements of application protocol functions
    - State machine operations, event dispatching, …
  - Policy language specifies variations of individual protocols
    - State machine transitions, payload format, …
  - Key advantage: Minimize efforts for releasing new shields
- ShieldPot:
  - Distributed shield-equipped honeypots
  - Detect (stealthy) unknown attacks against known vulnerabilities

22

Outline

- Product side:
  - Software Development Life cycle (SDL)
  - Compile-time solutions:
    - /GS compiler option
    - Static checking
  - Windows XP SP2
- Research side:
  - Shielding before patching (Shield, research)
  - System management (Strider, research)

23

Strider: Patch Management

- The challenge of software patches: testing
- Patch Impact Analysis
  - Use file and registry tracing to quickly narrow down the set of apps that need to be tested

24

Strider: Security Access Check Tracer

- Problem: user-level app runs with Admin privilege – compromise of user-level app is a system compromise
- Security Access Check Tracing
  - A developer tool for identifying every access that would fail for a non-admin, along with helpful debugging information
    - Kernel-mode tracing around security subsystem
  - Most admin dependencies are easy to remove once pinpointed

25

Questions?

26