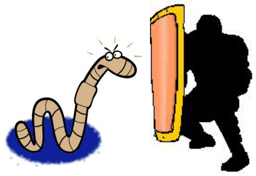


Shield
-- A First Line Worm Defense

Helen J. Wang, Chuanxiong Guo,
Dan Simon, and Alf Zugenmaier

Feb 25, 2004

Motivation

- Slammer, MSBlast, CodeRed, Nimda all exploiting ***known*!** Vulnerabilities whose patches are released ***months*!** before
- Software patching has ***not*** been an effective first line worm defense

Why don't people patch?

- ***Disruption:***
  - **Service or machine reboot**
- ***Unreliability***
  - **Software patching inherently hard to test**
- ***Irreversibility***
  - **Most patches are not designed to be easily reversible**
- ***Accident***
  - **Unaware of patch releases**

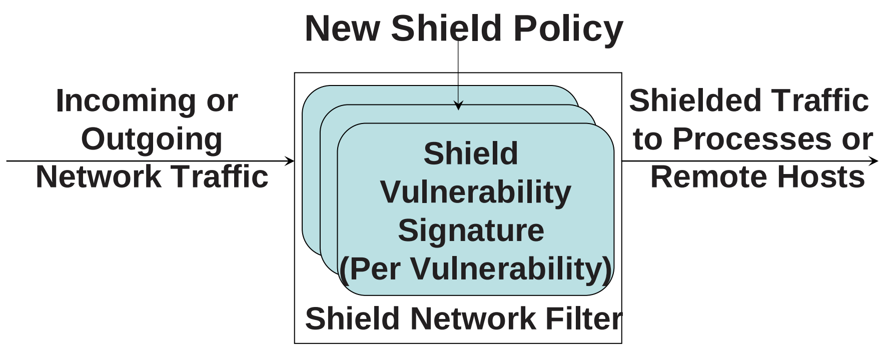
Microsoft Research

Our Vision: Shielding Before Patching

- Shield addresses the window between vulnerability disclosure and patch application.
- Shields: ***vulnerability-specific, exploit-generic*** network filters. Currently focus on end-host based shields.
- Patch is the ultimate fix of the vulnerability
  - Shield is removed upon patch application

Microsoft Research

Overview of Shield Usage



- Shield lies above the transport layer.

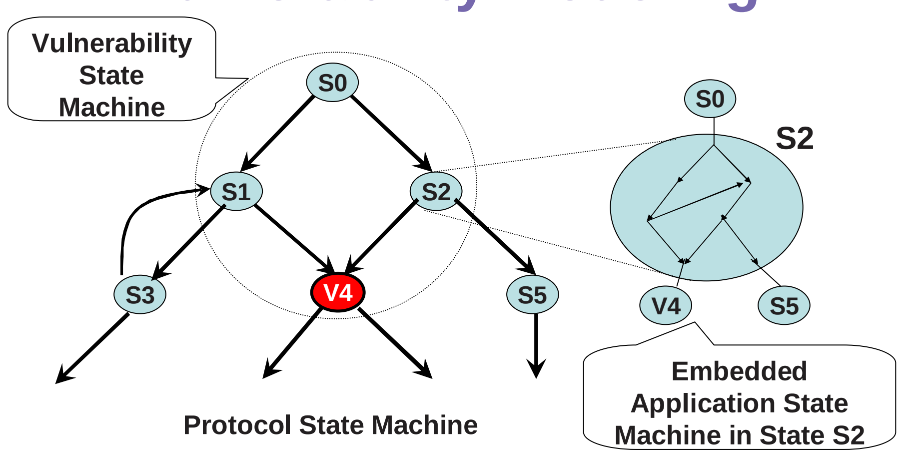
Why apply shields instead?

- *Non-intrusive*
  - No service or machine reboot
- *Easy testability -- Reliable*
  - Configuration independent, unlike patches – much fewer number of test cases
  - Simple testing through large trace replay or existing test suites for the protocol in question

Outline

- ✓ Motivation and overview
- Vulnerability Modeling
- Shield Architecture
- Shield Language
- Analysis
- Shield prototype implementations
- Initial evaluations
- Related Work
- Concluding Remarks

Vulnerability Modeling



Shield Vulnerability Signature:

Specifies vulnerability state machine and describes how to recognize exploits in the vulnerable event

Shield Architecture: Goals

- Minimize and limit the amount of state maintained by Shield
- Enough flexibility to support any application level protocols
- Defensive design

Microsoft Research

Flexibility: Separate Policy from Mechanism

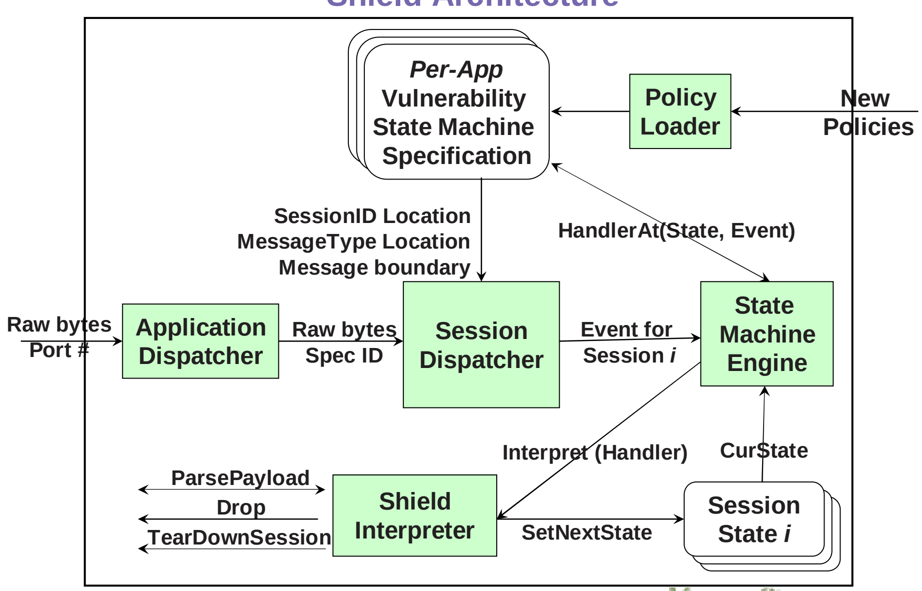
- ***Shield Mechanisms***: generic elements all application level protocols
  - All use finite state automaton for protocol operations
  - Event identification and session dispatching
  - Out-of-order datagram handling
  - Application level fragmentation handling
- ***Shield Policies:*** varying aspects of individual application level protocols
  - Application identification, event identification, session identification, vulnerability state machine specifications

Microsoft Research

Shield Architecture: Essential Data Structures

1. Per-app vulnerability state machine spec (***Spec***):
   - Transformed from Shield policy
   - Instructions for emulating vulnerability state machines in Shield at the runtime:
     - Application identification: ports, dynamic port registration
     - Vulnerability signature + reactions: states, events, handlers for recognizing and reacting to potential exploits
     - Event and session identification:
       - Location (offset, size) vector of event type and session ID in the app message. Unit: byte or "WORD" for text-based protocols
     - Message boundaries, e.g., CRLF CRLF for HTTP and SMTP
   - One state machine per application
     - Multiple vulnerability state machines are merged into one
2. Session State: current state and session context for exploit-checking

Shield Architecture



Scattered Arrivals of an Application Message

- *An application message* is the smallest interpretable unit by the application
- Why scattered arrivals?
  - Congestion control or application-specific message handling
- Copying: save then pass on
- What to save (*parsing state*): the name of the current incomplete field, the value of the current incomplete field only if the value is needed by Shield later
  - Per application message
- How to differentiate parsing state belonging to multiple sessions:
  - Safe to use socket here because only one socket should be used for delivering a complete application level message despite the M-M relationship between sockets and sessions.
- Pre-session copying: before the session info arrives
  - The parsing state is associated with the socket only
- In-session copying: after the session info arrives
  - The parsing state becomes part of the session state

Microsoft Research

Out-of-Order Application Datagrams

- Save out-of-order datagrams
- What is the max? Same as the application
- Additional info needed in Shield policy: seq num location, max number of saved datagrams

Microsoft Research

Application Level Fragmentation

- Over TCP: same treatment as scattered arrivals of a single application level message
- Over UDP: ordered copies of the fragments are treated the same as scattered arrivals
- Additional information needed in Shield policy: frag ID location

Microsoft Research

Outline

✓ Motivation and overview
✓ Vulnerability Modeling
✓ Shield Architecture
- Shield Language
- Analysis
- Shield prototype implementations
- Initial evaluations
- Related Work
- Concluding Remarks

Microsoft Research

Shield Policy Language

```
SHIELD (MSBlast, TCP, (135, 139,445))

SESSION_ID_LOCATION = (12, 4);
MSG_TYPE_LOCATION = (2, 1);

INITIAL_STATE S_WaitForRPCBind;
FINAL_STATE   S_Final;
STATE          S_WaitForRPCBindAck;
STATE          S_WaitForRPCAlterContextResponse;
STATE          S_WaitForRPCRequest;
STATE          S_WaitForSessionTearDown;

# Event types
EVENT E_RPCBind =       (0x0B, INCOMING);
EVENT E_RPCBindAck = (0x0C, OUTGOING);
EVENT E_RPCRequest = (0x0,  INCOMING);
…

STATE_MACHINE = {
(S_WaitForRPCBind, E_RPCBind, H_RPCBind),
(S_WaitForRPCBindAck, E_RPCBindAck, H_RPCBindAck),
(S_WaitForRPCBindAck, E_RPCBindNak, H_RPCBindNak),
(S_WaitForRPCBindAck, E_RPCCancel,  H_RPCCancel),
(S_WaitForRPCRequest, E_RPCRequest, H_RPCRequest),
…
};
# Payload
PAYLOAD_STRUCT {
  SKIP BYTES(2)          pContextID,
  BYTES(1)               numTransferContexts
  SKIP BYTES(1)          dummy1,
  BYTES(16)              UUID_RemoteActivation,
  SKIP BYTES(4)          version,
  SKIP BYTES(numTransferContexts * 20)
                         allTransferContexts,
} P_Context;

PAYLOAD_STRUCT {
  SKIP BYTES(12)          dummy1,
  BYTES(4)                callID,
  SKIP BYTES(8)           dummy2,
  BYTES(1)                numContexts,
  SKIP BYTES(3)                         dummy3,
  P_Context[numContexts] contexts,
  SKIP BYTES(REST)              dummy4,
} P_RPCBind;
…

HANDLER H_S_RPCBind (P_RPCBind)
{
  IF (>>P_RPCBind.contexts[0] ==
      0xB84A9F4D1C7DCF11861E0020AF6E7C57)
    RETURN (S_WaitForRPCBindAck);
  FI
  RETURN (S_Final);
};

HANDLER H_RPCRequest (P_RPCRequest)
{
  IF (>>P_RPCRequest.bufferSize > 1023)
    TEARDOWN_SESSION;
    PRINT ("MSBlast!");
    RETURN (S_Final);
  FI
  RETURN (S_WaitForSessionTearDown);
};
```

Microsoft Research

Shield Policy Language: Cont.

- High specialized for Shield's purpose
- Part 1: Vulnerability state machine specification and generic application level protocol info such as ports used, the locations of the event type, session ID, message boundary, etc.
- Part 2: Handler and payload parsing specifications for run-time interpretation
  - Handler specification:
    - Variable types: BOOL, COUNTER, BYTES, WORDS
    - Two scopes: local or session
    - Statements: assignment, IF, special-purpose FOR-loop
  - Payload specification:
    - Skippable fields of BYTES, WORDS, BOOL, or arrays of PAYLOAD_STRUCTs
- Coping with scattered arrivals:
  - handler continuation – part of the session state consisting of statement ID queue, parsing state
  - Stream-based built-in length functions or regular expression functions: e.g., "COUNTER c = MSG_LEN (legalLimit);" c = legalLimit + 1 if msg exceeds "stopCount" number of bytes

Microsoft Research

Outline

- ✓ Motivation and overview
- ✓ Vulnerability Modeling
- ✓ Shield Architecture
- ✓ Shield Language
- Analysis
- Shield prototype implementations
- Initial evaluations
- Related Work
- Concluding Remarks

Microsoft Research

Analysis: Scalability

- Scalability with Number of Vulnerabilities
  - # of shields doesn't grow indefinitely – upon successful patching, the corresponding shields are removed
  - N shields for N apps ⇔ 1 shield
  - Multiple vulnerabilities of a single app can compound if they share paths on the vulnerability state machine – not significant because no more than 3 worm-exploitable vulnerabilities seen in a single application in 2003
  - Application throughput is at worst halved, traffic processed once in Shield and once in the application

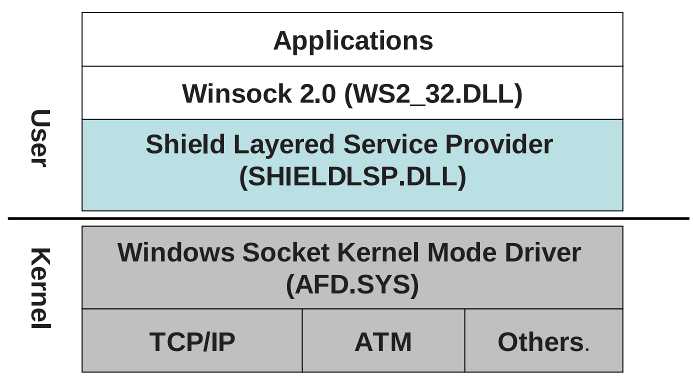
Microsoft Research

Analysis: False Positives

- Low false positives by nature
- Two sources:
  - Misunderstanding of protocol and payload spec – can be debugged with large traffic trace or test suites
  - Differential treatment of a certain network event: could be an exploit in one runtime setting, and yet completely legal in another

Microsoft Research

Shield Prototype Implementation



- 10,702 line C++ code;
- Experimented with 15 vulnerabilities and 7 application level protocols, such as RPC, HTTP, SMTP, FTP, SMB

Microsoft Research

Outline

- ✓ Motivation and overview
- ✓ Vulnerability Modeling
- ✓ Shield Architecture
- ✓ Shield Language
- ✓ Analysis
- ✓ Shield prototype implementations
- Initial evaluations
- Related Work
- Concluding Remarks

Microsoft Research

Evaluation: Shield-ability

- What are hard to shield:
  - Virus
    - *vulnerability-driven* anti-virus software would be a better alternative
  - Vulnerabilities that could be embedded in HTML scripting
  - Application-specific encrypted traffic – may be hard to get the key.
    - But for SSL/TLS, an SSL-based shield framework can potentially be built on top of SSL

Microsoft Research

Evaluation: Shield-ability, Cont.

# of vul.	*Nature*	*Worm-able*	*Shield-able*
6	Local	No	No
24	Client	No	Hard
12	Server input validation	Yes	Easy
3	Cross-site scripting	No	Hard
3	Server DoS	No	Hard

Study of 49 vulnerabilities from MS Security bulletin board in 2003

Microsoft Research

Evaluation: Throughput

- Clients and a server use RPC/TCP. Server sends 100 MB of data back to initiating clients. Every byte is accessed by Shield on the server
- Both have P4 2.8GHz and 512 MB of RAM, connected by 100Mbps Ethernet switch.

Microsoft Research

Evaluation: Throughput

# of clients	w/o Shield (Mbps)	w/ Sheld (Mbps)
10	86.51	86.20
15	86.57	86.36
50	86.66	86.20
100	86.48	85.86
150	86.67	86.24
200	86.06	81.70
500	84.27	82.29
1000	66.29	57.56

Microsoft Research

Evaluation: False Positives

- Evaluate on shield for Slammer.
- Used an SSRP stress test suite obtained from a MS test group: 32 test cases for 12 message types
- No false positives observed.

Microsoft Research

Related Work

- Threats of Internet worms:
  - 0wn Internet, CodeRed study, Inside Slammer, Internet quarantine, Warhol
- Insufficiency of patches:
  - Timing patching, CodeRed study,
- Firewall
  - More coarse-grained, high-false positive solution
  - Will be much improved by fast exploit-signature generation schemes such as "early bird"
- NIDS (such as Bro), traffic normalizers
  - Different layers and different purposes from Shield

Microsoft Research

Concluding Remarks

- Shield: vulnerability-specifc, exploit generic network filters for preventing exploits against known vulnerabilities.
- Initial prototyping and evaluation results are encouraging

Microsoft Research

Ongoing Work

- Gaining experience and evolving our language and architecture design
- Shield policies more difficult to write, but can be potentially easy to automate the difficult part of it
- Shield at firewall or edge router.
- Shield testing
- Vulnerabilities easier to reverse-engineer with Shield – need secure, reliable and expeditious distribution
- Apply Shield principle to anti-virus – scalability a key challenge.

Microsoft Research