Beyond Patch and Pray: Security by Design

• Adam Shostack
• Presented to the Security Leadership Conference Series
• Arlington, TX
• Oct 19 2004

Goal

• Much of today’s security seems to cycle through:
  – Penetrate or otherwise find vulnerabilities in deployed systems
  – Fix the issues
  – Pray that you do it before the bad guys or the worms
So?

- This is very expensive
- Fixing deployed systems risks downtime
  - Could deploy “patch management” sw
- Or we could look to fix problem from root causes

Bug (and software) Development
How To Move?

• It’s actually worse than that
• That’s a graph for a single program
• You deploy lots of programs
How To Get There

• Better software tools
  – Internal, external
• Better Deployment tools
  – Security
  – Operations
Where The Tools Fit

Software Improvement

- Static checkers
- Dynamic Checkers
- Languages
- Limits of software improvement
Static Checkers

- Work with source code
- Lots of different languages
- Results generally easier to fix
  - They’re associated with lines of code
- High false positive rates
- Find “sins of commission” like strcat()
- Fast

Free Static checkers

- RATS
- ITS4
- Flawfinder
Static Checkers: Slicers

• Compiler-like technology to see what variable could be touched where
  – Perl’s taint mode
• Clever techniques to deal with pointers
• Can be perfect on small code (20kloc)
• Much research

Static Checkers: Parsers

• Analyze variables and typing because C doesn’t
• Can deal with integer issues well
• Slower
• SPLINT is a free example
Static Checkers: Compilers

- Compile code, and analyze on the way
- Code is not always compiled to your processor
  - Target a VM that has security features
- MOPS
- Dawson Engler’s group @ Stanford
- GCC -Wall is not complete

Dynamic Checkers

- Work on binary code
  - Never wonder if the optimizer was too clever
- Find “Sins of Omission” like SQL injection
- Slow! (Can be hours or days)
Dynamic Tools: Fuzzers

- Fuzz, Spike, libwhisker
- Mangleme http fuzzer (added after talk)
  - http://lcamtuf.coredump.cx/soft/mangleme.tgz
- Feed noise to the target see if it breaks
- And you’re surprised this is slow?

Dynamic: Attack Simulation

- “Second Gen fuzzers”
- Attack tool libraries
- CORE Impact, Metasploit
- Require skilled driver
- Nikto
  - Less powerful, easy to use
Dynamic Tools: Decompilers

- Turn byte code/machine code into something resembling C
- Useful for closed source apps you need
- Need to analyze the decompiled source

Dynamic: Binary Differs

- Not a dynamic tool as much as a static tool for machine code
- Best for finding why a patch happened
- Attack/exploit creation
- Vendor verification:
  - Is this patch effective?
  - Are they being upfront about what’s in it?
Language Selection

• Some languages seem to be more prone to security flaws
  – C, PHP
• We may not have found the classes of flaws in Java, C#
• New classes keep showing up (integer underflows, etc)

Things Hard to Measure

• Security design goodness
• Attack surface
  – nmap not enough
  – port 25 seems to have a large surface
  – port 137 does too.
Adding Resilience to Code

- How to
  - deploy
  - operate
- Buggy code *more* securely

Free UNIX techniques

- chroot/jail
- Unprivileged daemon accounts
  - Painful if you need fast code on port 80
- Free security enhanced OSes:
  - OpenBSD, SELinux
Techniques

• Harden the system:
  – Control Attack surface
  – Limit effect of an attack
• Can entail high operational cost for questionable benefit
  – Need to evaluate what happens

More advanced tools

• OS hardening tools
  – Immunix subdomain
  – Sana kernel enhancements
• Application hardening
  – Stackguard & company
  – (Recompile vs kernel modules)
Issues with Hardening Tools

- How to measure their effectiveness
- Configuration effort
- Costs (perceived and real)
  - Cash up front
  - Speed
  - Supportability + Vendor finger pointing

Network Intrusion Prevention

- Throwing Ducks at Balloons
  - Paper by Ptacek and Newsham, 1998
  - Showed how to evade IDSs, IPSs
- The Covert Channel problem
Firewalls Move (back) Up

• Application Firewalls vs packet filters
• Inspection
• Snort Inline

Process Resilience Tools

• How to fail gracefully
  – detect, respond, improve
• Measuring your process
• Architecture and Forensics
  – An ounce of prevention
Selling Your Boss

• Or, Security folks are from Mars, businesspeople are from Wheaton

How You Buy Software

• Functionality, supportability, price
• Can you get security in there?
• Probably requires being able to get lots of complexity into a 1-5 score (or somesuch)
• The above can be used for that
Sample Scoring

- 0-1 point for a good language
- 0-1 point for documented use of tools to check code
- 0-1 point for unprivileged, chroot install
- 0-1 point for logging
- 0-1 point for local analysis

Deployment Budgets

- Cash for wires, hubs, power, air
- Where does security fit?
- What’s the real cost of a failure?
  - (Hint, its not $1m, unless you’re a large bank)
Deployment Business Cases

• Cost of operations with and without tool X
• Cost of special events:
  – Patching
  – Breakins
  – Worms
• Frequency of special events

Conclusions

• Way back to patching
• Learned how to cut # of patches
  – Better SW
  – Better operations
  – Better sales to management