

CS574

Computer Security

San Diego State University

Spring 2008

Lecture #9

Today's Structure

- Administrivia
- Questions
- Recent News
- Lecture

Administrivia

- Exam next wednesday

Questions?

Recent News

- Vulnerability Disclosure & Purchasing
- Cracking a “Crypto Hard Drive”
- MD5 collisions
- Harmful Search Results
- Quantum Computers and P & NP
- Social Engineering
- Privacy & Anonymity

Lecture

More Malicious Code

- Rootkits
- Web Bugs
- Salami Attacks
- Keyloggers
- Man-in-the-Middle attacks
- Covert Channels

Rootkits

- A *rootkit* is a program or set of programs that attempt to change the user's environment so as to remain undiscovered.
- Intercepts commands and/or system calls, and filters output to remove evidence.
- Example: Sony's XCP and the \$sys\$ prefix.

Web Bugs

- Any small/invisible web-page component, traditionally 1x1 pixel transparent GIFs.
- May set a cookie on the system, or just provide a unique URL for every user.
- Used to track web-usage patterns.

Salami Attack

- Really thin slices won't be missed, but can be aggregated into something useful.
- Traditional example: the fractional-cent roundoff.
- Relies on most folks trusting another's math more than their own.

Key Loggers

- Any program or device that can intercept and record user input.
- May be software or hardware.
- Workarounds depend on where the keylogger is intercepting the data.

Man in the Middle

- Adds another entity in the chain of communication.
- Very difficult to defend against.
- General concept, shows up at many levels
 - hardware
 - software
 - network

Covert Channels

- A secret way of communicating without being noticed, using a non-obvious communication channel (“banging on the walls”).
- From an insider (rogue programmer, trojan horse, etc.) to an untrusted outsider.
- Difficult to trace data flow.
- A big concern on multi-classification shared systems.

Software Flaws

Software Flaws

- Incorrect assumptions.
- Buffer overflows.
- Incomplete mediation.
- Synchronization / Serialization
(“time of check to time of use”)

Assumptions

- “This wouldn’t ever happen.”
- Programming for Satan’s computer.
- Example (day one, program one, error one)

Buffer Overflows

- May *corrupt* data.
- May *crash* the program
- May *inject* code.

Incomplete Mediation

- May provide a means to induce a *buffer overflow*.
- May *leak* information.
- May *corrupt* data.
- *Client-side validation* is a type of incomplete mediation.

Synchronization / Serialization

- Also “*time of check to time of use error*” and “*race condition attack*”.
- A result of not all operations being *atomic*.
- May need to run hundreds or thousands of times to find the right *window*.
- If the system gets very slow, the window for the attack gets larger.

Synchronization Attack Techniques

- Filesystem attacks with
 - access(“filename”, mode)
 - checking for existence
 - symbolic links
- Input validation
 - pass-by-reference variables
 - buffers

Software Design Flaws

Some Stupid System Design Principles:

- no “least privilege”
- no protected memory
- no access control
- no authentication
- no integrity checking

Software Controls

- Good software practices
 - best practices
 - modularity
 - peer reviews
 - testing
- Pick the right language for the task at hand.
- Penetrate-and-patch uses and downfalls.
- Input mediation (aka validation).

Software Controls

- Redundancy (compare results).
- Active/Passive fault detection.
- Fault tolerance:
 - sandboxes (e.g., JVM, Zones, Xen, ...)
 - noexec data and stack segments
- Testing
 - unit/component/system/functional
 - performance/acceptance/regression
 - black-box/white-box or clear-box

Design Principles (1/3)

- Principle of Economy of Mechanism
Security mechanisms should be as simple as possible.
- Principle of Complete Mediation
Require all accesses to objects to be checked to ensure that they are allowed.

Design Principles (2/3)

- Principle of Open Design
The security of a mechanism should not depend on the secrecy of its design or implementation.
- Principle of Separation of Privilege
A system should not grant permission based on a single condition.

Design Principles (3/3)

- Principle of Least Common Mechanism
Mechanisms used to access resources shall not be shared.
- Principle of Psychological Acceptability
Security mechanisms should not make the resource more difficult to access than if the security mechanisms were not present.

Lecture References

- Pfleeger, *Security in Computing*, 3rd & 4th edition
- Bishop, *Introduction to Computer Security*

End Lecture

Reading

- Review Chapters 1, 2, 3, and 8
- <http://cr.yp.to/qmail/qmailsec-20071101.pdf>

Finis