Code (in)Security

Where do code vulnerabilities come from.

Shachar Shemesh
http://www.shemesh.biz

What's on the Plan

- General background on the security arena.
- Introduce coding practices that often lead to vulnerabilities.
  - More details on Buffer Overruns.
  - Format strings will be discussed another time.
  - Step by step demonstration of exploit code – next time.
- Show how these vulnerabilities are exploited.

A Few Terms

- Vulnerability - a software bug that enables an attacker to gain undesired capabilities.
- Exploit - the steps required to gain the undesired capabilities from the vulnerability.
- Arbitrary code execution - the highest form of exploit, allowing an attacker to inject arbitrary code into the vulnerable program, and have that code executed.

A Few More Terms

- Denial of Service (DoS) - an attack that only allows the attacker to inhibit a service, without gaining anything else.
- "Owning" a machine - achieving the same (or higher) level of control over an attacked machine as the legitimate administrator of the machine.
- Root kit - a set of files and utilities the attacker leaves on the cracked machine, to allow easy re-entry, or to collect information not otherwise immediately available.

The People

- Hacker - A person with curiosity for making things work outside their intended envelope.
- Cracker - A person who exploits vulnerabilities in order to gain unauthorized capabilities.
- Script Kiddy - A cracker who does not understand, and often is also not interested, in the mechanics behind the attack tools he/she uses. These people use tools made by others in order to attack machines.

Types of Attacks

- Worms and viruses - the attack is performed by an automatic utility, usually not aware of who it is attacking.
- Sweep attacks - the attacker (usually a script kiddy) is interested in owning as many as possible.
- Targeted attacks - an attacker targets a specific entity, due to principal, political or financial reasons.

Most attacks today are done by insiders

Types of Vulnerabilities

It is close to impossible to list them all.

Buffer Overruns

- Two major types.
  - Stack overruns.
  - Heap overruns.
- It is almost impossible to write a C program that does not have one.
  - BIND, sendmail, Windows NT Kernel, tcpdump, etc.
- Arbitrary code execution is relatively easy, and becoming easier as new techniques are found.
  - Recently - also for heap overruns.

Format Strings

- Stems from passing untrusted buffer as the format string for "printf" like functions.
- Easy to find during an audit, easy to fix.
- Easy to find in the binary, easy to exploit.
  - A format string vulnerability that echoes the result to the attacker is like giving the attacker a debugger into the application.
  - Arbitrary code execution exploitation is relatively easy.

Incorrect Error Handling

- Not checking a function's return code is not always harmless.
  - DoS as a result of a disk full, or no memory free.
  - WinNuke – A TCP connection to SMB with OOB data would cause BSOD.
- Sometimes this can lead to more serious problems.
  - ICQ long password login problem
- Sometimes it can even lead to arbitrary code execution.
  - Double free in zlib and many others.

Lack of Input Validation

- Most common among Web applications.
- This can often lead to serious breaches in the security model.
  - XSS
  - Allowing arbitrary queries into the backend database.

Rogue Messages (Win32)

- Interactive services – receive messages from unprivileged processes.
- All of those messages affect the execution flow.
- Some of those messages copy buffers from unprivileged to privileged space.
- Some of those messages (WM_TIMER) contain pointers that are immediately executed.

Race Conditions

- Temporary file creation.
  - Most common case - creating a temporary file in a location both known, and with access, to an attacker.
  - Allows bypassing of the security model, changing internal program data structures and, in some cases, arbitrary code execution.
- Network related races
  - ARP poisoning.
  - DNS poisoning.

Evolution of a Security Exploit

1) Someone finds a bug.
2) Someone (usually same someone) writes a PoC exploit.
3) Someone standardizes the exploit.
4) Script kiddies can now use the exploit to break in.
5) A worm can be written to automatically exploit.

But They Don't Have the Source…

- *One* person talented enough to find the bug is enough.
- The "Copy Protection" wars of the 80's show that no program is above reverse engineering.
- CSS, GSM, RC4, Word passwords, SecureID.

Window of Exposure

- A graph describing how likely for a given machine to be cracked using a given vulnerability.
- Increases slowly the more time the vulnerability is there.
- Increases quickly the more time passes from the publication.
- Decreases once a patch is available.
- Greatly decreases once a worm is released for that vulnerability.

Full Disclosure

- In the (distant) past, people who found vulnerabilities reported them discretely to the vendors.
  - No fixes at all, or not in a timely manner.
  - The "Black Hat" community still knew of problems.
  - Problems were never patched.
- The “Full Disclosure” movement.
  - Vulnerabilities are reported, but then disclosed to the public.
  - The negative PR usually forces the vendor to patch.
- Master key vulnerability – modern example.

What to Do?

- Write bug-free code ☺.
- Fix problems as soon as possible.
- Create easy to install reliable patches.
- Never threaten the revealer of the information with legal actions.
  - It is amazing how many companies fail this simple advice.
  - Doesn’t work, but creates a backlash.

What Else to Do?

- Security audits.
- Code audits.
- Design reviews.
- Code comments.
- Code reuse.
- Careful design.
- Error handling.
- All the other things we all know and never do.

End of Part I

Questions?