Program Security and Vulnerabilities

Week 3
In this lecture

- Nonmalicious Program Errors
- Buffer Overflow
- SQL Injection Attack
- Incomplete Mediation
- Time-of-Check to Time-of-Use Errors
- Malicious Code
Ethical Use of Security Information

- We discuss *vulnerabilities* and *attacks*
  - Most vulnerabilities have been fixed
  - Some attacks may still cause harm
  - Do not try these at home or anywhere else

- Purpose of this class
  - Learn to prevent malicious attacks
  - Use knowledge for good purposes
Law Enforcement

- David Smith
  - *Melissa virus*
  - 5 years in prison, $150K fine
- Ehud Tenenbaum ("The Analyzer")
  - *Broke into US DoD computers*
  - 6 month service, suspended prison, $18K fine
- Dmitry Sklyarov
  - *Broke Adobe ebooks*
  - Prosecuted under DMCA
SQL in Web Pages

- SQL can be used to display data on a web page
- Web users can input their own search values
- Dynamically change SQL statements to provide the user with selected data:
- Example (Server side code):
  - txtUserId = getRequestString("UserId");
  - txtSQL = "SELECT * FROM Users WHERE UserId = " + txtUserId + ";
  - txtSQL is a select statement
  - Fetch data from “Users” database for “txtUserId” to web the page
SQL Injection Attack

- Technique where malicious users can inject SQL commands into an SQL statement, via web page input.
- Injected SQL commands can alter SQL statement and compromise the security of a web application.
SQL Injection Attack Type 1

- 1=1 is always true
- txtUserId = getRequestString("UserId");
  txtSQL = "SELECT UserId, Name, Password FROM Users WHERE UserId = " + txtUserId + ";

- Malicious user can enter smart (but wrong) input as txtUserId

<table>
<thead>
<tr>
<th>UserId:</th>
</tr>
</thead>
<tbody>
<tr>
<td>105 or 1=1</td>
</tr>
</tbody>
</table>

- Server code:
  - SELECT UserId, Name, Password FROM Users WHERE UserId = 105 or 1=1
  - Valid: will return all rows from the table “Users”
SQL Injection Attack Type 2

- SQL Injection Based on "='' is Always True
- Server code:

uName = getRequestString("UserName");
uPass = getRequestString("UserPass");

sql = "SELECT * FROM Users WHERE Name ='' + uName + '' AND Pass ='' + uPass + ''"
SQL Injection Attack Type 2 (cont’d)

- SQL Injection Based on ""="" is Always True
- Server code:
  ```python
  uName = getRequestString("UserName");
  uPass = getRequestString("UserPass");
  sql = "SELECT * FROM Users WHERE Name ="" + uName + "" AND Pass ="" + uPass + """;
  ```

- Attacker can insert " or "=" into the name and password box
- Server code becomes
  ```sql
  SELECT * FROM Users WHERE Name ="" or ""="" AND Pass ="" or ""="""
  ```
SQL Injection Attack Type 3

- SQL Injection Based on Batched SQL Statements
- Batched SQL statements: separated by semicolon
  - SELECT * FROM Users; DROP TABLE Suppliers
  - Return all rows in the Users table, then delete the table called Suppliers
SQL Injection Attack Type 3 (cont’d)

- SQL Injection Based on Batched SQL Statements

  - Server code

  ```
  txtUserId = getRequestString("UserId");
  txtSQL = "SELECT * FROM Users WHERE UserId = " + txtUserId;
  ```

  Server code becomes

  SELECT * FROM Users WHERE UserId = 105; DROP TABLE Suppliers

User id:

```
105; DROP TABLE Suppliers
```
In this lecture

- Nonmalicious Program Errors
- Buffer Overflow
- SQL Injection Attack
- Incomplete Mediation
- Time-of-Check to Time-of-Use Errors
- Malicious Code
Incomplete Mediation

- What if par2 is
  - 1800Jan01 (outside of range)
  - 2000Feb30 (non-existent)
  - 2048Min32 (undefined)
  - 1Aardvark2Many ?!
- How to fix such errors?
  - Have client side code to verify input correctness
  - Restrict choices to only possible ones, e.g., drop-down menus ...
Incomplete Mediation (cont’d)

- *Still vulnerable!*
  - The results of the verification are accessible in the URL
  - The (malicious) user can access and modify fields
  - Only then send to the server
  - The server cannot tell if URL came directly from the user browser or from malicious user
In this lecture

- Nonmalicious Program Errors
- Buffer Overflow
- Incomplete Mediation
- Time-of-Check to Time-of-Use Errors (TOCTTOU)
- Malicious Code
TOCTTOU Errors

- Concurrency issue
  - Successive instructions may not execute serially
  - Other processes may be given control
- Access control
  - Only users with *rights* can access objects
- TOCTTOU: control is given to other process *between* access control check and access operation
TOCTTOU Example

```c
int openfile(char *path) {
    struct stat s;
    if (stat(path, &s) < 0)
        return -1;
    if (!S_ISRREG(s.st_mode)) {
        error("only allowed to regular files");
        return -1;
    }
    return open(path, O_RDONLY);
}
```

- **Path to file**
- **Extract file meta-data**
  - Between check and open, attacker can change `path`
  - Initial `path` is regular file
  - Later `path` is not
  - Adversary by-passes security
- **Open file**

No symlink, directory, special file
TOCTTOU: How an Attack Works

- openfile is being run within the kernel (at the OS)
- At the user space level, there is a program P
  - Controlled by adversary
  - Program P defines \texttt{path} variable
- Program P also launches two threads T1 and T2
  - T1 and T2 share the \texttt{path} variable
  - If T2 changes path, T1 also sees the change
- T1 runs openfile where path is set to a file
- T2 sets path to a directory
TOCTTOU Prevention

1. Ensure critical parameters are not exposed during pre-emption
   ▪ openfile “owns” path

2. Ensure serial integrity
   ▪ openfile is atomic
   ▪ No pre-emption during its execution

3. Validate critical parameters
   ▪ Compute checksum of path before pre-emption
   ▪ Compare to checksum of path after ...
In this lecture

- Nonmalicious Program Errors
- Buffer Overflow
- Incomplete Mediation
- Time-of-Check to Time-of-Use Errors (TOCTTOU)
- Malicious Code
What is Malicious Code

- Malicious Code (Malware) is a set of instructions that run on your computer and make your system do something that an attacker wants it to do.
- We previously studied program vulnerabilities
- Malicious code: exploits them
What Is It Good For?

- Steal personal information
- Delete files
- Click fraud
  - Pay-per-click online advertising
  - Person, automated script or computer program imitates a legitimate user of a web browser clicking on an ad, for the purpose of generating a charge per click without having actual interest in the target of the ad's link. [Source: wikipedia]
- Steal software serial numbers
- Use your computer as relay
Types of Malicious Code

- Virus
  - Transient
  - Resident
- Trojan horse
- Logic/time bomb
- Worm
- Etc ...
What is a Virus?

- Program that can infect other programs by modifying them to include a, possibly evolved, version of itself

- Fred Cohen 1983
Some Virus Type

- Polymorphic: uses a polymorphic engine to mutate while keeping the original algorithm intact (packer)
- Methamorphic: Change after each infection
What is a Trojan

- **Class of malware that appears to perform a desirable function but in fact performs undisclosed malicious functions that allow unauthorized access to the victim computer** [Source: Wikipedia]

- Trojans are not viruses since they do not replicate, but Trojan horse programs can be just as destructive.
What is a Logic Bomb

- Type of Trojan Horse that executes when specific conditions occur:
  - Change in a file
  - Particular series of keystrokes
  - Specific time or date.
What is a Worm

- Reproducing programs that run independently and travel across network connections
- Virus vs. Worm - Reproduction Differences:
  - A virus is dependent upon a host file or boot sector, and the transfer of files between machines to spread.
  - A worm can run completely independently and spread of its own will through network connections.
What is Spyware/Adware

- Collects small pieces of information about users without their knowledge.
- The presence of spyware is typically hidden from the user, and can be difficult to detect
  - Secretly installed on the user's PC
  - By the user of a shared computer to secretly monitor other users
Almost 30 years of Malware
Malware Repartition

- Trojan: 74%
- Worm: 9%
- Adware: 13%
- Other: 3%
- Spyware: 1%