Information Security: Application Development Vulnerabilities

Ghodrat Moghadampour, PhD
mg@vamk.fi
Principal Lecturer
Vaasa University of Applied Sciences
Vaasa
Finland

Outline

- Information Security
- Application Vulnerabilities
  - Buffer Overflow
  - Race Conditions
  - Input Validation Attacks
  - Authentication Attacks
  - Authorization Attacks
  - Cryptographic Attacks
  - Web Security
    - Client-Side Attacks
      - Cross-Site Scripting
      - Cross-Site Request Forgery

Outline cont.

- Clickjacking
- Defense
- Server-Side Attacks
  - Lack of Input Validation
  - Improper or Inadequate Permissions
  - Extraneous Files

Information Security

- Information Security: protecting information and information systems against unauthorized access, use, disclosure, disruption, modification or destruction
- In general sense, security means protecting our assets, i.e. protecting them from atatckers invading our networks, natural disasters, adverse environmental conditions, theft or vandalism or other undesirable states

Application Security

- Application security can be threatened by a number of common development misconducts
- The main categories of software development vulnerabilities include:

  - **Buffer overflows**
  - **Race conditions**
  - **Input validation attacks**
  - **Authentication attacks**
  - **Authorization attacks**
  - **Cryptographic attacks**

Application Security: Buffer Overflows

- Buffer overflows (*buffer overruns*) occur when the size of data is not limited when requiring input into the application
- Typically, most programming languages will require that we specify the amount of data we expect to receive, and reserve storage for that data
- If the amount of data taken by the program is not limited (bounds checking), we may receive much more input data than expected:

  Example: 100 chars instead of 8 chars

Application Security: Buffer Overflows

- In case of excess data, extra characters may be written over other areas in the memory that are in use by other applications, or the operating system itself
- Proper bounds checking can nullify this type of attack entirely. Bound checking may be implemented automatically by the programming language like in Java and C#

Application Security: Race Conditions

- Race condition occurs when multiple processes or multiple threads within a process control or share access to a particular resource, and the correct handling of that resource depends on the proper ordering or timing of transactions
- For example, if we are making a 60e withdrawal from bank account via an ATM, the process might go as follows:

Application Security: Race Conditions

- Step 1: check the account balance (1000e)
- Step 2: withdraw funds (60e)
- Step 3: update the account balance (940e)
- If someone else starts the same process around the same time and tries to make a 40e withdrawal, we might end up with a bit of problem:

Race Conditions

- User 1, step 1: Check the account balance (1000 e)
- User 2 step 1: Check the account balance (1000 e)
- User 1 step 2: Withdraw funds (60 e)
- User 2 step 2: Withdraw funds (40 e)
- User 1 step 3: Update the account balance (940e)
- User 2 step 3: Update the account balance (960e)

Application Security: Race Conditions

- In this case, two users race to access the shared resource (bank account) and undesirable conditions occur: we end up with a balance of 960e being recorded, where we should see only 900e
- Race conditions can be very difficult to detect in existing software as they are hard to reproduce
- Careful handling of shared resources can generally help avoid such issues

Application Security: Input Validation Attacks

- If input data is not validated, we may find ourselves in trouble depending on the particular environment and language being used
- Format string attack is a good example of such attacks
- Certain print functions within a programming language can be used to manipulate or view the internal memory of an application
- In C and C++ languages we can use format string such as %f, %n, %p and %x to apply formatting to the data we are printing to the screen

Application Security: Input Validation Attacks

- Using format strings the attacker could execute code, read the stack, or cause a segmentation fault in the running application, causing new behaviors that could compromise the security or the stability of the system.
- If a format String parameter, like %x, is inserted into the input data, the string is parsed by the Format Function (like printf, fprintf) and the conversion specified in the parameters is executed

Application Security: Input Validation Attacks

- If arguments are not supplied, the function could read or write the stack
- Examples

```
printf ("Entered argument: %s" , argv[1] ) ;
…
./prog "Today %x %x"
./prog %s%s%s%s%s%s%s%s%s%s%s
```

Application Security: Authentication Attacks

- Strong authentication mechanisms will help react in a reasonable manner in the face of attacks
- There are a number of common actions which we can take in order to make them stronger
- If we put a requirement for strong passwords into applications when we are doing password authentication, this will help a lot in keeping attacks away

Application Security: Aauthentication Attack

- An eight-character all-lowercase password, like *todaysun*, can be broken by a fairly powerful machine in a matter of seconds: p= (1/25)^8
- An eight-character, mixed case password that includes also a number and a special character like *To5aysu/*, can require up to two years to be broken
- p=(1/128)^8
- Applications should not use hard-coded (built-in, unchangeable) passwords
- Authentication on the client side will also put the application under more threat

Application Security: Authorization Attacks

- Placing authorization mechanisms on the client side is a bad idea and is almost guaranteed to be a security issue at some point
- When we are authorizing a user for some activity, we should do so using the principle of least privilege both for users and processes
- Whenever a user or process attempts an activity that requires particular privileges, we should always check again to ensure that the user is indeed authorized for the activity in question, each time it is attempted

Application Security: Authorization Attacks

- If a user by accident or intentionally gains access to restricted portions of the application, we should have measures in place that will not allow the user to proceed

Application Security: Cryptographic Attacks

- Cryptography is easy to implement poorly, and this can give us a false sense of security
- One of the most common temptation is to develop a cryptographic scheme of our own devising
- The major cryptographic algorithms in use today, such as Advanced Encryption Standard (AES) and RSA, have been developed and tested by thousands of experts
- Additionally, such algorithms are in general use because they have been able to stand the test of time without serious compromise

Application Security: Cryptographic Attacks

- While designing the software, we should also allow for the use of different algorithms or make sure that changing algorithms is not impossible
- We should also plan for changing the encryption keys the software uses, in case our keys break or become exposed

Web Security

- Web pages ad applications are prevalent nowadays and attackers can use an enormous variety of techniques to compromise our machines, steal sensitive information, and trick us into carrying out activities without our knowledge
- These types of attacks divide into two main categories:

  - Client-side Attacks
  - Server-side Attacks

Web Security: Client-Side Attacks

- Client-side attacks take advantage of weaknesses in the software on the client side, or use social engineering to trick us into going along with the attack
- The major types of such attacks are:

  - Cross-Site Scripting
  - Cross-Site Request Forgery
  - Clickjacking
  - Defense

Client-Side Attacks: Cross-Site Scripting (XSS)

- Cross-site scripting (XSS) is carried out by placing code in the form of a scripting language into a Web page, or other media, that is interpreted by a client browser, including Adobe Flash animation and some types of video files
- When others view the same Web page, the attack code will get executed automatically
- The attacker can leave a comment containing the attack script on a blog window and every user reading the command on the browser would execute the attack

Client-Side Attacks: Cross-Site Request Forgery

- Cross-Site Request Forgery (XSRF) is carried out by placing links on a Web page to initiate a particular activity or another Web page or application where the user is currently authenticated
- For example, such a link might cause the browser to add items to the shopping cart on an online shop, or transfer money from one bank account to another
- If we are browsing several pages and are still authenticated to the same page, the attack is intended for, we might execute the attack in the background and never know it

Client-Side Attacks: Clickjacking

- Clickjacking takes advantage of the graphical display capabilities of our browser to trick us into clicking on something we might not otherwise
- Clickjacking attacks work by placing another layer over the page, or portion of the page, in order to obscure what we are actually clicking
- For example, the attacker might hide a button that says "order" under another layer with a button that says "more information"

Client-Side Attacks: Defense

- Defense: in many cases, new attack vectors are simply variations of old attacks
- There are numerous vulnerable clients running on outdated or unpatched software that are still vulnerable to attacks that are years old
- It is very important to keep up with the most recent browser versions and updates, as the vendors that produce them regularly update their protections

Client-Side Attacks: Defense

- For some browsers, we can apply additional tools in order to protect us from client-side attacks
- One of the well-known of these tools is NoScript which blocks most Web page scripts by default and allows only those that we specifically enable

Server-Side Attacks

- The server-side threats and vulnerabilities can vary widely depending o the operating system, Web server software, various software versions, scripting languages and many other factors
- These attacks are typically related to the following issues:
  - Lack of input validation
  - Improper or inadequate permissions
  - Extraneous files

Server-Side Attacks: Lack of Input Validation

- Lack of input validation is a general security issue also with Web applications and some of the most common server-side Web attacks use this weakness
- SQL injection is a strong example of what might happen if we do not properly validate the input of Web applications
- In case our Web application communicates with SQL database, entering SQL queries or query parts can produce results not anticipated by the application developers

Server-Side Attacks: Lack of Input Validation

- SELECT * FROM Users WHERE UserId = uID;

  Hacking input: uID = 1000 OR 1=1;

- SELECT * FROM Users WHERE Username =""" + uname """ AND Password=""" + pwd + """;

  Hacking input:

  Username: " or ""="

  Password: " or ""="

- Most databases support batched SQL statement (a group of two or more SQL statements)

- SELECT * FROM Users WHERE UserId =""" + uID + """

  Hacking input: uID=1000; DROP TABLE Suppliers

Server-Side Attacks: Lack of Input Validation

- If the input into Web applications is validated and problematic characters are filtered out, we can often fend off such attacks before it even begins
- In many cases, filtering out special characters such as * % ' ; / will defeat such attacks entirely
- We can also use SQL parameters to protect a web site from SQL injection

Server-Side Attacks: Improper or Inadequate Permissions

- Inadequate permissions can often cause problems with Web applications
- Often there are sensitive files (like database credentials) and directories with sensitive files that will cause security issues if they are exposed to general users
- One area that might cause trouble is the exposure of configuration files

Server-Side Attacks: Extraneous Files

- When moving a Web server from development into production, one of the tasks often missed in the process is that of cleaning up any files not directly related to running the site or application, or that might be artifacts of the development or build process
- If the application source code files or backup copies, or text files containing notes or credentials, or any such related files are left, we may be handing the material an attacker is waiting for

Server-Side Attacks: Extraneous Files

- To protect ourselves against such threat we can make sure that all such files are cleaned up, or moved elsewhere if they are still needed
- As a good periodic check we can also ensure that, in the course of troubleshooting or upgrading, these items have not been left behind where they are visible to the general public

Thank you!

☺