CS 493
Secure Software Systems

Ch 6 Programming languages

Dr. Williams
Central Connecticut State University

Objectives

- Common threats in the programming language environment
- The risk of unrestricted user input
- The need to own information where context is understood
- Secure handling of arrays
- Mitigation techniques for common programming language attacks, like buffer overflow
- Risks and best practices in the use of external APIs and libraries

Goals

- Identify common attacks against programming languages.
- Identify mitigation techniques to prevent malicious input.
- Determine the highest risks to the use of an API or library.
- Identify the threats to the most common programming languages.
- Conduct an investigation into the programming languages used in your system.

Language Barriers

- **Programming languages** are convenience structures that keep a programmer from needing to speak the native language of the machine; they are written in a combination of variables and near human terms that are similar in meaning to human language.
- Common functions are needed for most programs and are available within libraries for different languages.

Language Barriers

- **Application programming interfaces (APIs)** have been created to allow a system to call existing functionality in another module or system through a specified interface.
- **Compiling** is the process of translating the high-level language into native machine Code.
- An **interpreted language** is one that has a lower layer of machine code that dynamically reads and interprets commands from a higher-level language.

Buffer Bashing

- A **buffer overflow**, in its most general sense, is when more information is written to a location than the location can hold.
- Most buffer overflows in programming result from a lack of constraint on the amount of input that is allowed or the mishandling of pointer variables.

Buffer Bashing

There are defenses against buffer overflow, though none of them are perfect:

- **Array bounding**
- **Pointer handler indirection**
- **Data canaries**
- **Strict read and write limits**

Stack Overflow

Two methods to prevent smashing the stack:

1. **Stack canaries:** A stack canary is a randomly chosen small integer value that is placed in the stack adjacent to the return address for code execution.
2. **Nonexecutable stacks:** This is a system-level protection wherein some portions of memory are not allowed to contain executable code.

Good Input

- **Input validation** is a common tool used in developing a system. This asserts that the information entered by a user is of proper format for the system to process.
- **Input validation** at the front end of the application is wonderful for getting legitimate users to line up correctly, but the attacker can circumvent the front end. If the same validation is not done on the back end of the system, your initial input validation is meaningless in terms of attack.

JIT Systems

There are several steadfast rules that should be employed with any system, but most of all with JIT systems:

- **Never execute your input.**
- **Keep a layer between your code and your input.**
- **Map your exceptions.**

Good Output

Output scrubbing is different than input scrubbing because you do not need to worry about format and injection unless you are passing user input through your system into another external component.

- **Encrypt information that is secret.**
- **Include only what is necessary for the external process.**
- **Do not assume trust for the external system.**

Inherent Inheritance and Overdoing Overloads

Two consideration that need to be made when dealing with secure coding:

1. **Inheritance** is the use of a base parent class and defining specific extensions of it.
2. An **overload** is a redefinition of an existing operation for a new class.

The Threatdown

The following list identifies some of the highest-reported vulnerabilities for these languages and what you can do to mitigate their effects.

C: has its main vulnerabilities in susceptibility to buffer overflow and data type mismatch.

C++: One of the main languages used in infrastructure along with C, C++ falls victim to buffer overflow as easily as C. No inherent bound checking occurs in either language. One of the other significant areas of vulnerability for C++ is the boundary conditions on floating point values.

The Threatdown

Java: One of the most popular and powerful languages, Java was designed to have the highest interoperability of any language in existence. The highest risk to Java is the ability to call out external executions and inject OS commands.

C#: is Microsoft's answer to Java. It exists on the .NET platform from Microsoft, which insulates it from most of the usual suspects when it comes to exploits. The most significant risk in using any .NET application is specifying an unsafe block of code.

The Threatdown

Visual Basic (VB): Although it has lost out for most thick client development, VB has found a new life in Active Server Pages (ASP). Remote code execution is the big item on the list for VB.

PERL: PERL is often called the “Duct Tape of the Internet” because of its unequaled power and ability to compile on demand within a server environment. It is vulnerable to command injection.

The Threatdown

- **PHP:** PHP is another common language used for web applications. In particular, sending malicious-form data could actually corrupt the internal data of a PHP application and run arbitrary code or simply cause a system crash.

Deployment Issues

- There is generally no way to get rid of the system calls that allow for compromise in the myriad network of libraries and system function calls that have been constructed.
- If you are using any portion of an external system, make sure it is running the latest version with all of the security patches in place.
- However, too much information given in an error log on a deployed system can let attackers know just what state they need to trigger for a compromise.

Summary

- Programming languages are tools, and they can be used well or misused just like any other tool.
- Knowing what you are using will help you avoid the pitfalls that can allow system compromise.
- Scrubbing user input wherever it exists is the main item that should be done for any programming language in any platform.

In class exercises revisited

Bob’s Pizza Shack

In groups consider this scenario and identify in general what data elements are going to have security concerns

- Bob is a small business owner of Bob’s Pizza Shack and wants to create a website to allow online credit card delivery orders

Alice’s Online Bank

In groups consider this scenario and identify in general what data elements are going to have security concerns

- Alice opens Alice’s Online Bank (AOB)
- What are Alice’s security concerns from a network perspective?
  - Consider ones in own environment
  - Consider ones interoperating with another bank
  - Consider ones interacting with customer
    - Web
    - Mobile app

Location based social media app

In groups consider this scenario and identify in general what data elements are going to have security concerns

- Open source group wants to create a mobile app to allow groups to communicate/find each other in public demonstrations/protests
  - Communication internet, as well as, P2P (WiFi/Bluetooth) in case internet cut off – so if person you want to contact is on other side of crowd and no internet as long as P2P network can be established with app can reach somebody outside your immediate vicinity via the P2P network
  - Should be able to communicate securely messages and images to people you identify within your group (group as whole or direct)
  - Should be able to share GPS location with people in group (group as whole or direct)
- Broader scope – Who are potential threats and associated ways could attack/weaken system?