

SOFTWARE SECURITY

**Information Security in Systems & Networks
Public Development Program**

Sanjay Goel

University at Albany, SUNY

Fall 2006

Software Vulnerabilities

Learning Objectives

- Students should be able to:
 - Ascertain what software security should include.
 - Identify software-related vulnerabilities.
 - Recognize various software-related attacks (including buffer overflow attack, tunneling attack, and back door).
 - Determine appropriate controls for software-based attacks.

Software Vulnerabilities

Software Security

- The root of most security problems is software that fails in unexpected ways. Perfect security does not exist.
 - Security vulnerabilities are the result of violating an assumption about the software (or, more generally the entire system).
 - Assumptions are necessary while creating software
 - As long as assumptions are made vulnerabilities exist
 - Vulnerabilities can be exploited by an attack.
- Example: Buffer overflows
 - Assumption (by programmer) is that the data will fit in the buffer.
 - This leads to a vulnerability: Supply data that is too big for the buffer (thereby violating the assumptions)

Software Vulnerabilities

Software Security: Context

- Security is a function of tradeoffs
 - Performance
 - Cost
 - Usability
 - Functionality
- The level of security required depends on the context of the problem
 - What are you are trying to protect?
 - How valuable is it?
 - In what way is it valuable?

Software Vulnerabilities

Software Security: Context

- Information may be important only to one person
 - Private Email/Passwords
 - Nuclear Secrets
- Some information important due to accuracy & reliability
 - Bank's Accounting Information
- System can be important because of
 - Critical services it provides
 - Someone can use it to cause physical damage
 - e.g. SCADA systems (dams, power plants, etc.)

Software Vulnerabilities

Software Security: Goals (CIA³)

- **Data Confidentiality**
 - Keep data and communication secret
 - Privacy of personal financial/health records etc
 - Military and commercial relevance
- **Data Integrity**
 - Protect reliability of data against tampering
- **Authentication**
 - Authenticity of the source & content of information
- **Availability**
 - Data/resources should be accessible when needed
 - Protection against denial of service attacks
- **Access Control**
 - Only authorized people have access to the data

Software Vulnerabilities

Buffer Overflow

- Definition:
 - Attacker tries to store more information on the stack than size of the buffer & manipulates memory stack to execute malicious code
- Typical Behaviors:
 - Varied attack and can be used for obtaining privileges on a machine or for denial-of-service.
- Vulnerabilities:
 - Takes advantage of the way information is stored by computer programs.
 - Programs which do not have a rigorous memory check in the code are vulnerable to this attack

Buffer Overflow

Incidents

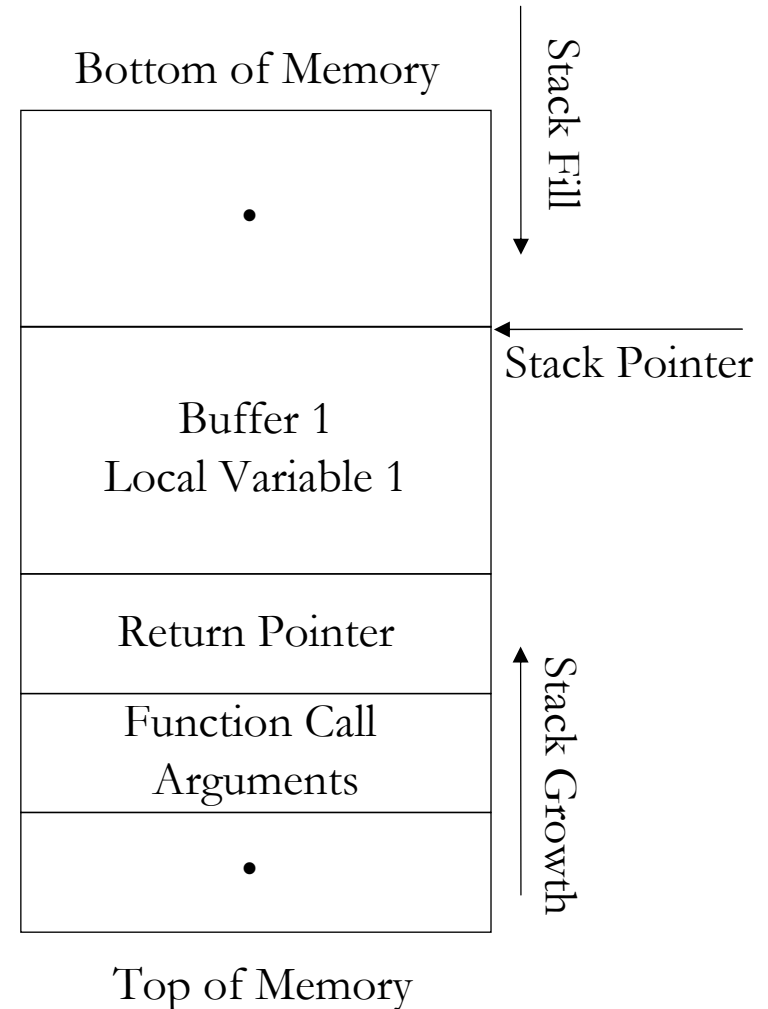
- Effectiveness of this attack has been common knowledge since the 1980's:
 - Used by the Internet Worm used in 1988 to gain unauthorized access to networks and systems.
 - Accounts for approximately half of all security vulnerabilities.
- According to one survey MS Blaster worm caused:
 - Remediation cost \$475,000 per company (median average - including hard, soft and productivity costs) with larger node-count companies reporting losses up to \$4,228,000.
 - Entered company networks most often through infected laptops, then through VPNs, and finally through misconfigured firewalls or routers.

Source: TruSecure / ICSA Labs, 29 August 2003.

Buffer Overflow

Creating Execution Stack

- Four bulk operations are performed to call a function in a conventional architecture:
 - The function's parameters are saved onto the stack
 - The return address is saved onto stack
 - Execution is transferred to the called function.
- Once the function completes its task, it jumps back to the return address saved on the stack
- Note that the string grows towards the return address

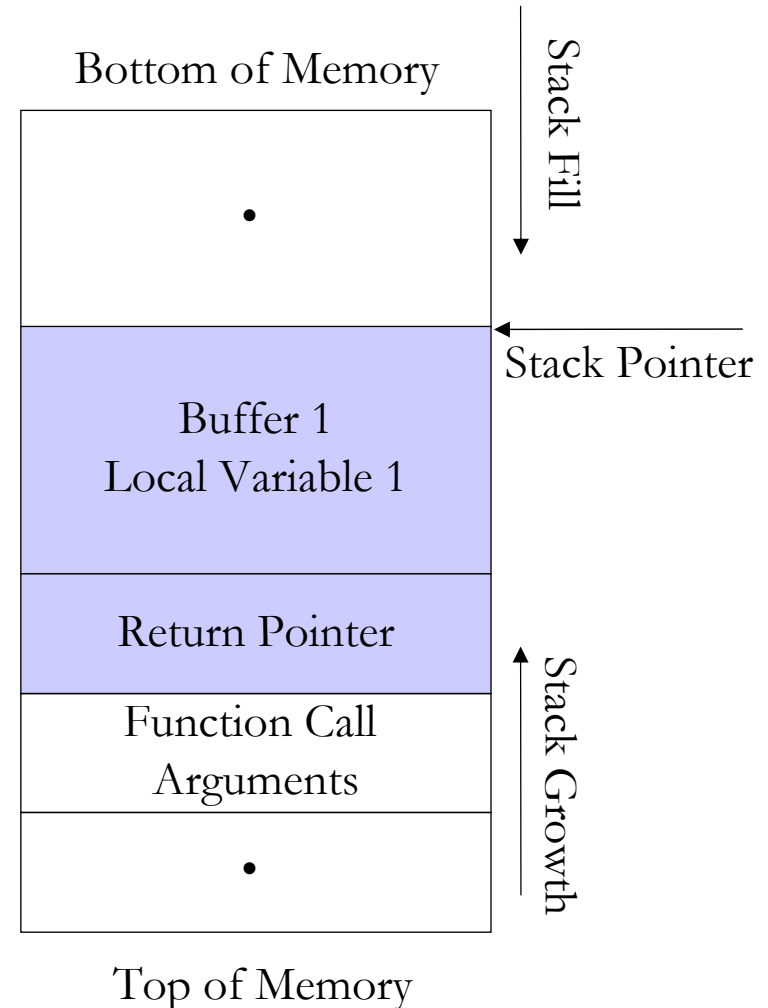


Normal Stack

Buffer Overflow

Vulnerability

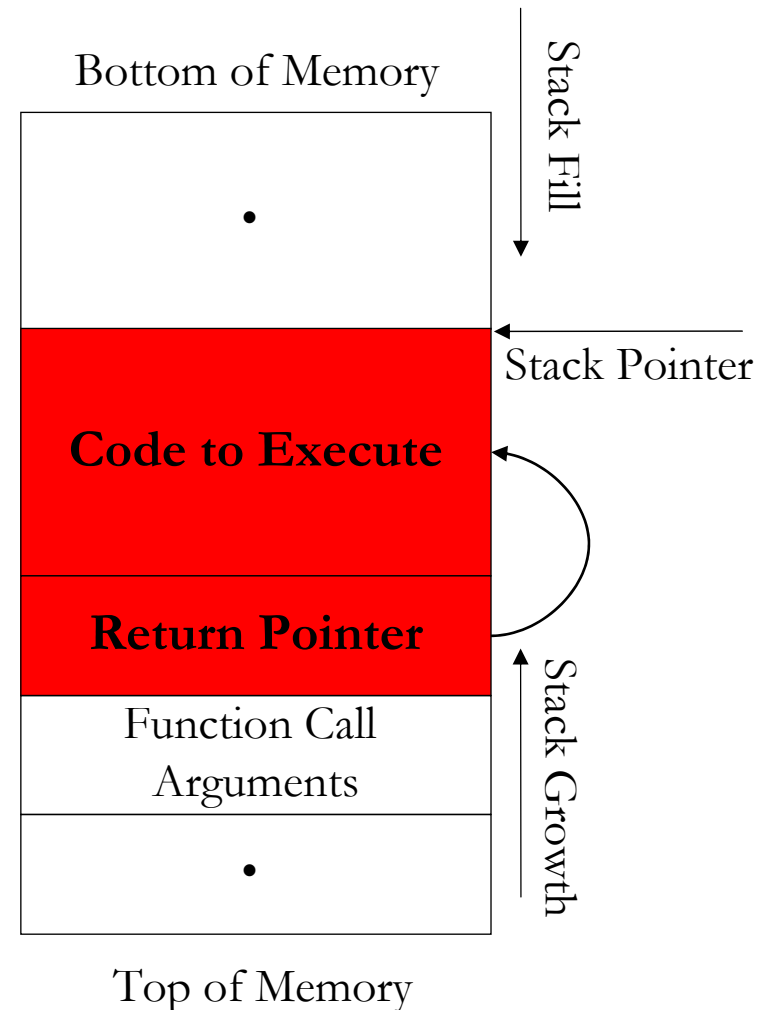
- Buffer overflow vulnerability occurs where an application reads external information such as a character string and an input string larger than the allocated buffer memory is sent (and the application doesn't check the size).
 - Input will normally come from an environment variable, user input, or a network connection.
 - e.g. if memory allocated for name is 50 characters, and a name of more than 50 characters is input by user
- The return pointer can be overwritten by the user data



Buffer Overflow

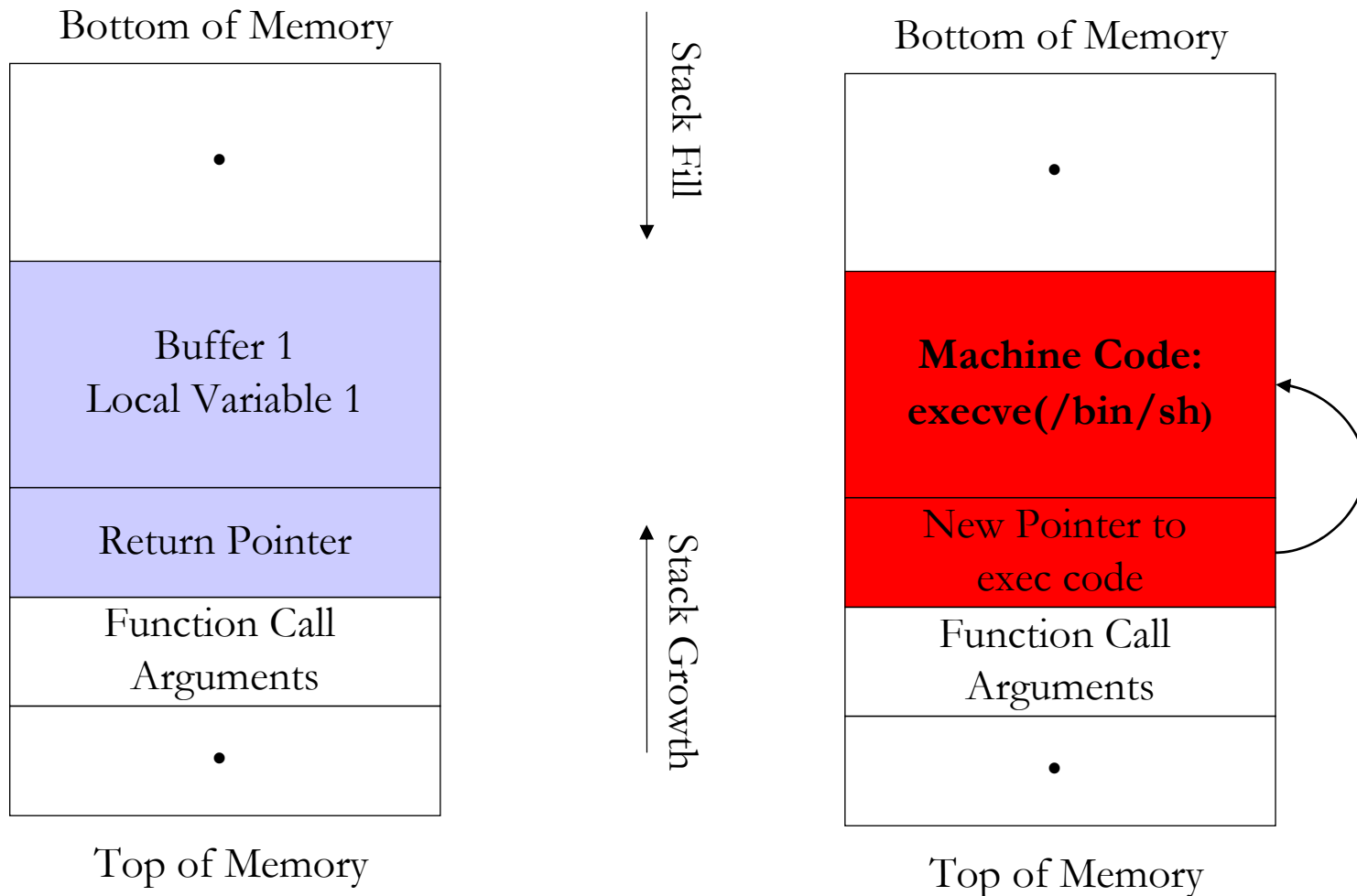
Executing the Attack

- Inject the attack code, which is typically a small sequence of instructions that spawn a shell, into a running process
- Change the execution path of the running process to execute the attack code.
 - Change the value of the return address to the address of malicious code
- Both of the goals must be achieved at the same time to perform a successful attack.



Buffer Overflow

Compare Stack



Buffer Overflow

Protection/Detection

- Avoid programming mistakes.
- **Patch, patch, patch !!!**
 - Does not effectively protect from zero day exploits.
- Keep your AV software up to date.
- Follow layered approach to information security – segment your network, only allow the necessary services, block and log everything else.
- Monitor your network for anomalous activity.
- Deploy Buffer Overflow kernel patches – not 100% effective but may be able to protect your environment the next time your patching process misses a few boxes.

Buffer Overflow

Scenario

- Impact: Can be used for espionage, denial of service or compromising the integrity of the data
- Common Programs
 - NetMeeting Buffer Overflow
 - Outlook Buffer Overflow
 - AOL Instant Messenger Buffer Overflow
 - SQL Server 2000 Extended Stored Procedure Buffer Overflow

Software Vulnerabilities

Tunneling

- Definition:
 - Attempts to get “under” a security system by accessing very low-level system functions (e.g., device drivers, OS kernels)
- Typical Behaviors:
 - Behaviors such as unexpected disk accesses, unexplained device failure, halted security software, etc.
- Vulnerabilities:
 - Tunneling attacks often occur by creating system emergencies to cause system re-loading or initialization.
- Prevention:
 - Design security and audit capabilities into even the lowest level software, such as device drivers, shared libraries, etc.
- Detection:
 - Changes in date/time stamps for low-level system files
or changes in sector/block counts for device drivers
- Countermeasures:
 - Patch or replace compromised drivers to prevent access
 - Monitor suspected access points to attempt trace back.

Software Vulnerabilities

Software Security: Back Door

- Definition: A back door is a secret entry point into a computer program that bypasses security mechanisms.
 - Back doors sometimes installed so program can be accessed for troubleshooting or other purposes.
 - Attackers often use back doors that they detect or install themselves, as part of an exploit
- Inserted during code development
 - Accidentally (forget to remove debugging code)
 - Intentionally (maintenance)
 - Maliciously (an insider creates a hole)

Software Vulnerabilities

Software Security: Back Door

- Prevention:
 - Enforce defined development policies
 - Limit network and physical access
- Detection
 - Audit trails of system usage especially user identification logs
- Countermeasures
 - Close trap door or monitor ongoing access to trace back to perpetrator

Software Vulnerabilities

Summary

- Weak software is often the cause of information system failure and vulnerability.
- It is important to consider CIA² when defining software security goals
 - Confidentiality, Integrity, Authentication, Availability, & Access Control
- Buffer Overflows result when a hacker takes advantage of memory stack limitations
- Tunneling involves getting “under” a system to execute low level functions (e.g. initialization of the system)
- A Back Door allows a hacker to return to a system through a secret entry point.