Software and Network Security

Furkan Alaca
furkan.alaca@carleton.ca

Carleton University

SYSC 4700, Winter 2018

Software Vulnerabilities

- Software vulnerabilities are often caused by programming errors, e.g., failure to **validate program input**
  - May be **locally** or **remotely** exploitable
- Vulnerabilities may also arise due to
  - Flaws in protocol design (e.g., KRACK)
  - Hardware flaws (e.g., Meltdown and Spectre)
- Software security is closely related to software quality and reliability. Primary difference is as follows:
  - Quality and reliability are concerned with accidental software failure due to some **legitimate** real-world input or program interaction that was unanticipated by the programmer or not adequately tested for.
  - Software security is concerned with ensuring that programs continue to function correctly even while under attack, either by executing safely or by failing gracefully. Requires **defensive programming**.
- When vulnerabilities are identified, they are typically **patched**

Buffer Overflow Introduction

- Programming error in which oversized input data is written to a fixed-size buffer and consequently overwrites adjacent memory

char buf1[5]; char buf2[5];

‘f’	‘o’	‘o’	‘\0’		‘b’	‘a’	‘r’	‘\0’	

strcpy(buf1, “SYSC 4700”);

‘S’	‘Y’	‘S’	‘C’	‘ ’	‘4’	‘7’	‘0’	‘0’	‘\0’

- Occurs in low-level languages (e.g., C/C++) that provide direct access to memory addresses (e.g., through pointers)

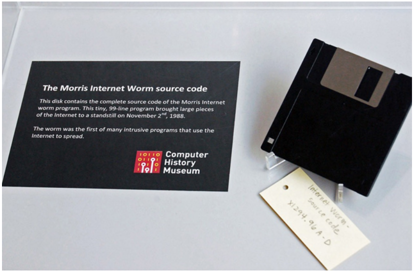
- Possible consequences:
  - Program crash
  - Data corruption (e.g., local variables)
  - Transfer control of the program to the attacker

Buffer Overflow Demo

Live demo exercise from:
`https://exploit-exercises.com/protostar/`

Infamous Buffer Overflow Incidents

- Morris worm of 1988: Infected and crippled 6,000 computers worldwide (about 10% of computers connected to the Internet)
- Blaster worm of 2003: Infected around 16 million computers
  - Implicated in Northeast blackout of 2003
  - Affected many other businesses, e.g., took down Air Canada check-in system



(a) Morris worm source code, Computer History Museum, Mountain View, CA

(b) Northeast power outage

Software Vulnerabilities: Summary of Defenses

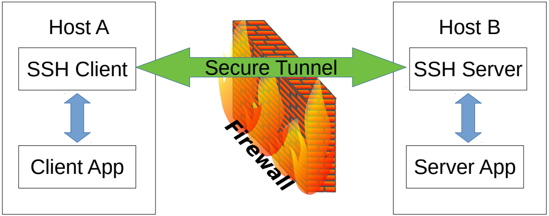
- Always stay up-to-date with latest security patches
- Defensive programming (e.g., input sanitization, use of "safe" library functions)
- Follow well-established security design principles, e.g., (non-exhaustive list)
  - Use secure defaults
  - Open design (do not rely on "security by obscurity")
  - Principle of least privilege
  - Defense in depth
  - Minimize attack surface (more on this in upcoming slides!)
  - More in the 1975 paper by Saltzer & Schroeder: *The Protection of Information in Computer Systems*

Reducing Network Attack Surface

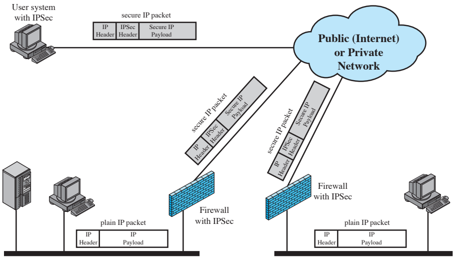
- Hosts on your network may be hosting network services which:
  - Don't follow stringent security practices
  - May be out-of-date, and therefore contain software vulnerabilities
- **Firewalls** can regulate access to your network in accordance with a security policy
  - Can filter all traffic based on source and destination IP address and TCP/UDP port
  - Most secure practice is to disallow all traffic by default, and make exceptions for the services which you would like to provide
- **Network Intrusion Detection Systems** (NIDS) can monitor network activity and detect:
  - Anomaly detection can detect traffic which deviates from normal patterns
  - Signature detection can detect traffic associated with known attacks
  - Honeypots can be used to observe attacker behaviour and improve firewall policies

Protecting Network Services

- Secure tunneling (e.g., `ssh` tunnelling) can be used to establish an encrypted connection between two hosts to access network services that are blocked by the firewall
- Reduced **attack surface** by keeping applications blocked on the firewall (i.e., disallow access from outside the network), and only allowing access to a single port through which authorized users can authenticate (e.g., via a password or public key) and establish a secure tunnel through which they can access any other applications (e.g., web server, network storage) running on the host



Virtual Private Networks



- A **Virtual Private Network (VPN)** consists of a set of LANs and remote hosts which are connected securely over a public network
  - Virtual: Uses existing infrastructure
  - Private: Data is encapsulated through a secure (encrypted) protocol
  - Network: The LANs and remote hosts can operate as one network
- Reduces attack surface by eliminating the need to make network services accessible through the public Internet
  - VPN users from remote locations can access services the same way as they would if they were physically present on the same LAN

Concluding Summary

- We learned the basics of why software vulnerabilities occur, how they are exploited, how to protect against them
- Network-level techniques for reducing attack surface
- Security design principles: These apply to all kinds of systems, e.g., computer networks, operating systems, web browsers
- Defense is more difficult than offense: The defender needs to defend against all possible attacks, but the attacker only needs to develop a single successful attack to gain a foothold on your system or network

Figures credit

- Figure from slide 10 taken from Computer Security Principles and Practice 3e by Stallings & Brown (2014)

Extra Slides

Software Vulnerability Case Study: OpenSSL Heartbleed

- Root cause of the 2014 OpenSSL Heartbleed vulnerability: Failure to validate input
- The TLS Heartbeat protocol sends a periodic message to indicate that the host is still alive during long idle periods
  - Client sends `heartbeat_request` message which includes payload length, payload, and padding fields
  - Server receives the request, allocates a buffer large enough to hold the message header, payload, and padding
  - Server saves the incoming message into the buffer, and transmits a response message which includes the payload length and payload fields
  - What happens if client sends a `heartbeat_request` with a payload length of 64KB but only includes a 16-byte payload?

Denial of Service Attacks

- Denial of Service (DoS) is a form of attack on the availability of some service (typically a network service)
- Can target network- , transport-, or application-layer resources
- DoS attacks may originate from:
  - A single system under the attacker's control
  - A large group of compromised hosts (e.g., botnets)
  - Traffic that the attacker "reflects" off of remote hosts
- Reflected DoS attacks are enabled by:
  - The ability to spoof the source IP address
    - ISPs should perform **egress filtering** to filter IP addresses that do not originate from their network
  - Network services on legitimate hosts which respond to any requests
    - e.g., a DNS request (˜60 bytes) can result in a massive 4000-byte response
  - Networks which accept directed broadcast packets
    - Organizations should perform **ingress filtering** to filter such packets

Denial of Service Attacks

Layer 3: Network bandwidth

- If there is more traffic destined to the organization's network than its link can carry, routers on the path will buffer and subsequently drop packets
- Hardest type of DoS attack to handle: Need to co-operate with upstream ISPs to install filtering rules on routers

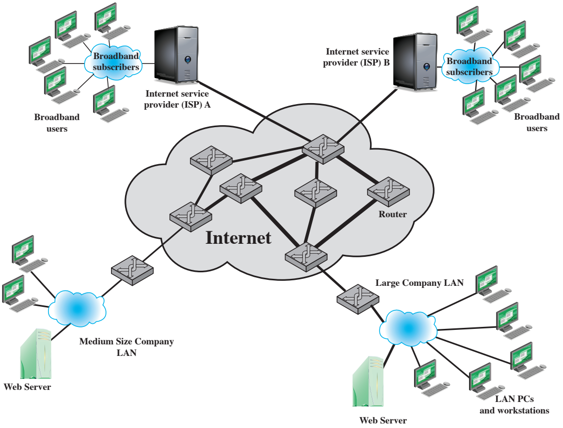


Figure 7.1 Example Network to Illustrate DoS Attacks

Denial of Service Attacks

Layer 4: System resources

- Aims to overload network handling software by targeting resources such as incoming packet buffers or tables of open connections
- Classic example: **SYN spoofing** overflows a server's TCP connection table to obstruct subsequent incoming connections
- Recall the TCP three-way handshake:
  - Client initiates the request for a TCP connection by sending a SYN packet to the server
  - Server records the request details (e.g., client address and port number, initial seq. number) in a table and responds with a SYN-ACK packet
  - Client responds with ACK, connection is established
- An attacker could flood the server with SYN packets with forged unused/inactive source addresses which will not reply to the server's SYN-ACK (typically with a RST, if the SYN-ACK was unsolicited)
  - **SYN Cookies:** Defensive mechanism where the server encodes connection details into initial sequence number in SYN-ACK packet instead of the connection table

Denial of Service Attacks

Layer 7: Application resources

- An attack on a specific application, e.g., a Web server, typically involves sending valid requests, each of which consumes significant resources
- Typical example: Flood target with HTTP requests that perform expensive SQL queries
  - Can rate-limit requests and blacklist IP addresses of abusive hosts to prevent overloading
  - Can use Captcha puzzles to help distinguish between legitimate human initiated-traffic and automated bots
- Other attacks may exploit software vulnerabilities, e.g., a buffer overflow or race condition, to cause a server to crash
  - Software should be kept up-to-date with security patches
  - Application-level firewalls can filter out known attack signatures