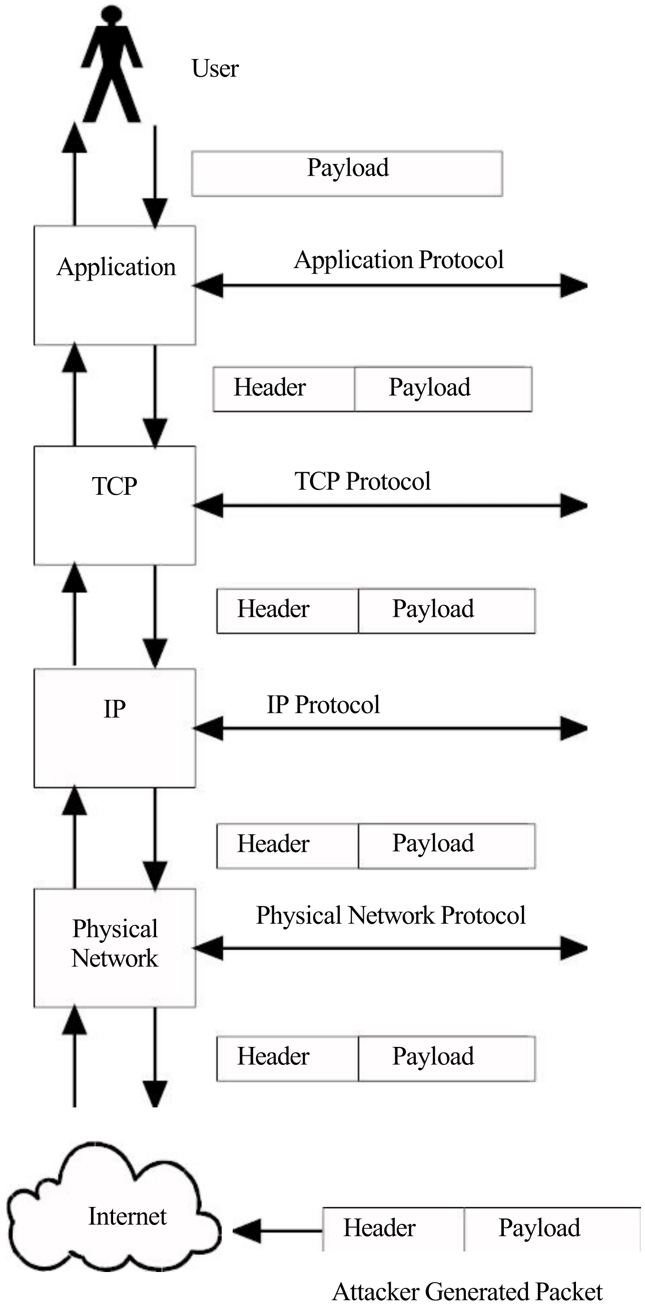
Introduction to Network Security

Chapter 4

Taxonomy of Network-Based Vulnerabilities

Network Security

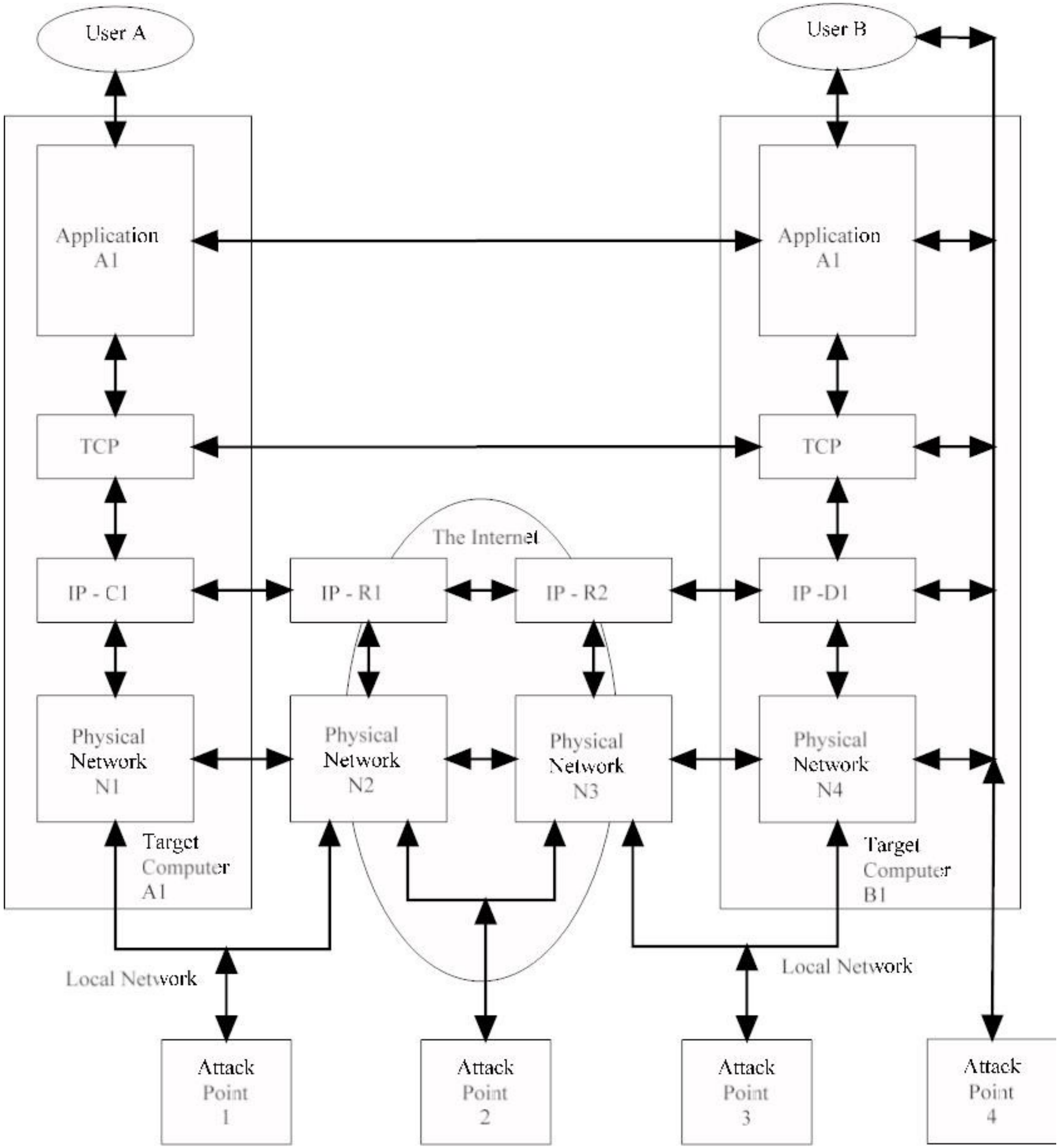
- Who (authentication)
  - Good guys
  - Bad Guys
- What to Attack
  - Protocols
  - Network connected Applications
  - Infrastructure



Layered Model of Attack Data

- Each layer receives data from the layer below and passes data to the layer above it without looking at it
- An attacker can insert information into the payload in order to send data to a particular layer

Threat Model



- Attacker 1 & 3 can attack any layer on computers connected to the same network

- Attacker 2 can attack the TCP & Application layers of computers A1 & B1 and the IP layer of any device

- Attacker 4 has taken over the computer

Bugs, Vulnerabilities, and Exploits

- A bug is a place where real execution behavior may deviate from expected behavior
- A vulnerability is a flaw or weakness in system security procedures, design, implementation, or internal controls that could be exercised (accidentally triggered or intentionally exploited) and result in a security breach or a violation of the system's security policy. (NIST's definition)
- An exploit is an input that gives an attacker an advantage

Vulnerability Finding Today

Security bugs can bring $500-$100,000 on the open market
Good bug finders make $180-$250/hr consulting
Few companies can find good people, many don’t even realize this is possible.
Still largely a black art

Security Vulnerabilities

- Remote Code Execution
  - Run arbitrary code in the context of the application
  - Hijack the execution logic of the application
  - Access to anything the application can access
- Denial-of-Service
  - Causes an application to crash or become unresponsive

    Category：

    Persistent

    Nonpersistent
- Information Disclosure

  Provide information it wasn’t originally designed to provide

    Contents of memory, filesystem paths, or authentication credentials

Security Vulnerabilities

- Authentication Bypass
  - Authenticate to the application without providing all the authentication credentials
  - SQL Injection → Bypass Authentication

- Authorization Bypass
  - Applications may support different types of users read-only, low-privilege, or administrator
  - Gain extra rights or access to resources they are not privileged to access.

Vulnerability Summary for CVE-2009-0341
Original release date:01/29/2009
Last revised:02/20/2009
Source: US-CERT/NIST
Static Link: http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2009-0341

Overview

The shell32 module in Microsoft Internet Explorer 7.0 on Windows XP SP3 might allow remote attackers to execute arbitrary code via a long VALUE attribute in an INPUT element, possibly related to a stack consumption vulnerability.

Impact

CVSS Severity (version 2.0):
CVSS v2 Base Score:9.3 **(HIGH)** (AV:N/AC:M/Au:N/C:C/I:C/A:C) (legend)
Impact Subscore: **10.0**
Exploitability Subscore: **8.6**
CVSS Version 2 Metrics:
Access Vector: **Network exploitable; Victim must voluntarily interact with attack mechanism**
Access Complexity: **Medium**
Authentication: **Not required to exploit**
Impact Type:**Provides administrator access, Allows complete confidentiality, integrity, and availability violation; Allows unauthorized disclosure of information; Allows disruption of service**

References to Advisories, Solutions, and Tools

By selecting these links, you will be leaving NIST webspace. We have provided these links to other web sites because they may have information that would be of interest to you. No inferences should be drawn on account of other sites being referenced, or not, from this page. There may be other web sites that are more appropriate for your purpose. NIST does not necessarily endorse the views expressed, or concur with the facts presented on these sites. Further, NIST does not endorse any commercial products that may be mentioned on these sites. Please address comments about this page to nvd@nist.gov.

External Source: **BID**
Name: **33494**
Hyperlink:http://www.securityfocus.com/bid/33494
External Source: **BUGTRAQ**
Name: **20090128 Internet explorer 7.0 stack overflow**
Hyperlink:http://www.securityfocus.com/archive/1/archive/1/500472/100/0/threaded

Vulnerable software and versions

Configuration 1 **AND OR cpe:/o:microsoft:windows_xp::sp3 OR * cpe:/a:microsoft:internet_explorer:7* Denotes Vulnerable Software**
* Changes related to vulnerability configurations

Technical Details

Vulnerability Type (View All)
Buffer Errors (CWE-119**)**
CVE Standard Vulnerability Entry:http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2009

Types of vulnerabilities

Types of vulnerabilities

API Abuse

Authentication Vulnerability

Authorization Vulnerability

Availability Vulnerability

Code Permission Vulnerability

Code Quality Vulnerability

Configuration Vulnerability

Cryptographic Vulnerability

Encoding Vulnerability

Environmental Vulnerability

Error Handling Vulnerability

General Logic Error Vulnerability

Input Validation Vulnerability

Logging and Auditing Vulnerability

Password Management Vulnerability

Path Vulnerability

Sensitive Data Protection Vulnerability

Session Management Vulnerability

Unsafe Mobile Code

Use of Dangerous API

内存损坏漏洞

Memory Corruption Vulnerabilities

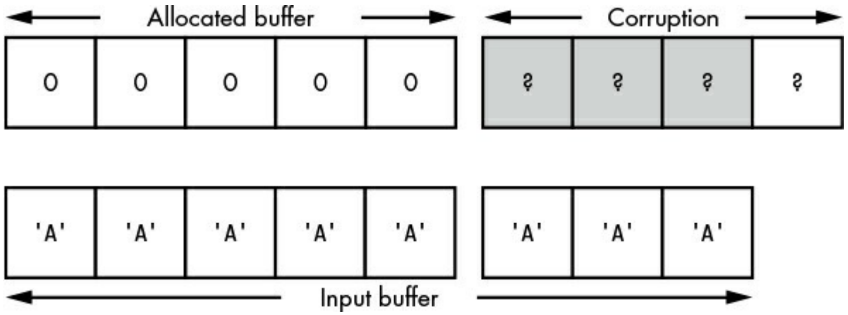
- Memory-Safe vs. Memory-Unsafe Programming Languages
  - Dependent on the programming language the application was developed in.
  - Perform bounds checking for in-memory buffer access

    执行内存缓冲区访问的边界检查

- Memory Buffer Overflows 内存缓冲区溢出

  Input data that is too large for the allocated buffer

  输入的数据对于分配的缓冲区来说太大



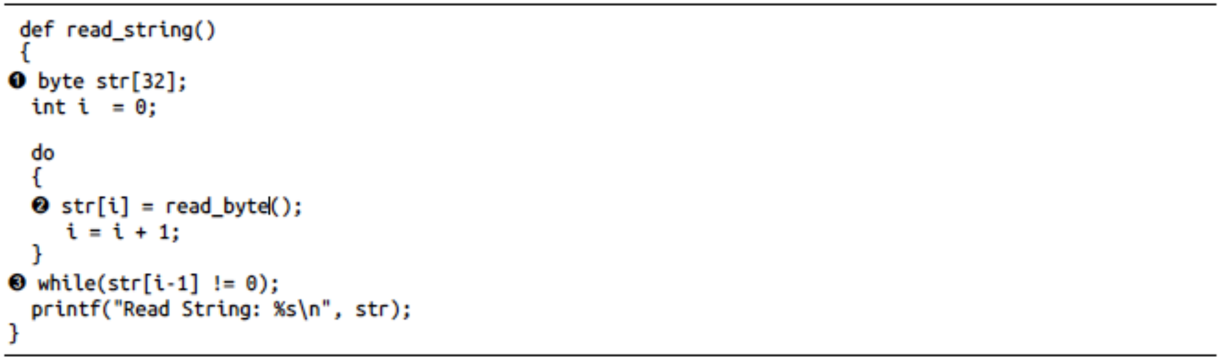
Fixed-Length Buffer Overflows

固定长度缓冲区溢出

- Memory length is determined prior to knowledge of the actual data length.

内存长度是在知道实际数据长度之前确定的

- Case Study



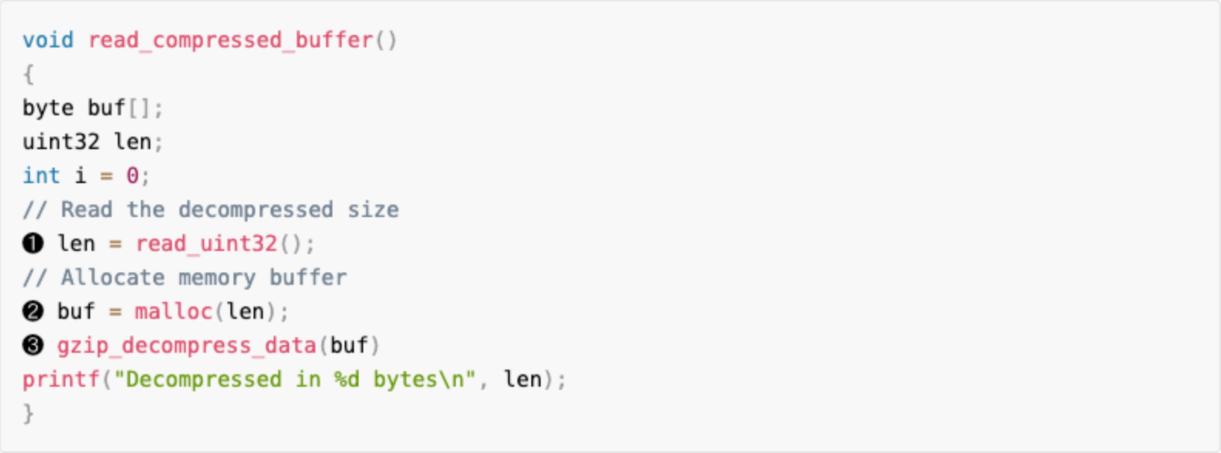
```
 def read_string()
 {
❶ byte str[32];
  int i  = 0;

  do
  {
  ❷ str[i] = read_byte();
     i = i + 1;
  }
❸ while(str[i-1] != 0);
  printf("Read String: %s\n", str);
}
```

The loop doesn’t verify the current length at ❸

Data Expansion Attack
数据扩展攻击

- The length of the decompressed data exceeds the size of the buffer 解压缩的数据长度超过了缓冲区的大小
- Case Study



```
void read_compressed_buffer()
{
byte buf[];
uint32 len;
int i = 0;
// Read the decompressed size
❶ len = read_uint32();
// Allocate memory buffer
❷ buf = malloc(len);
❸ gzip_decompress_data(buf)
printf("Decompressed in %d bytes\n", len);
}
```

Dynamic Memory Allocation Failures

动态内存分配失败

- A system’s memory is finite

  系统的内存是有限的

- In some languages

  Termination of the environment

  The generation of an exception.

  环境终止
  异常的产生

- Several possible vulnerabilities may arise 可能会出现几个可能的漏洞

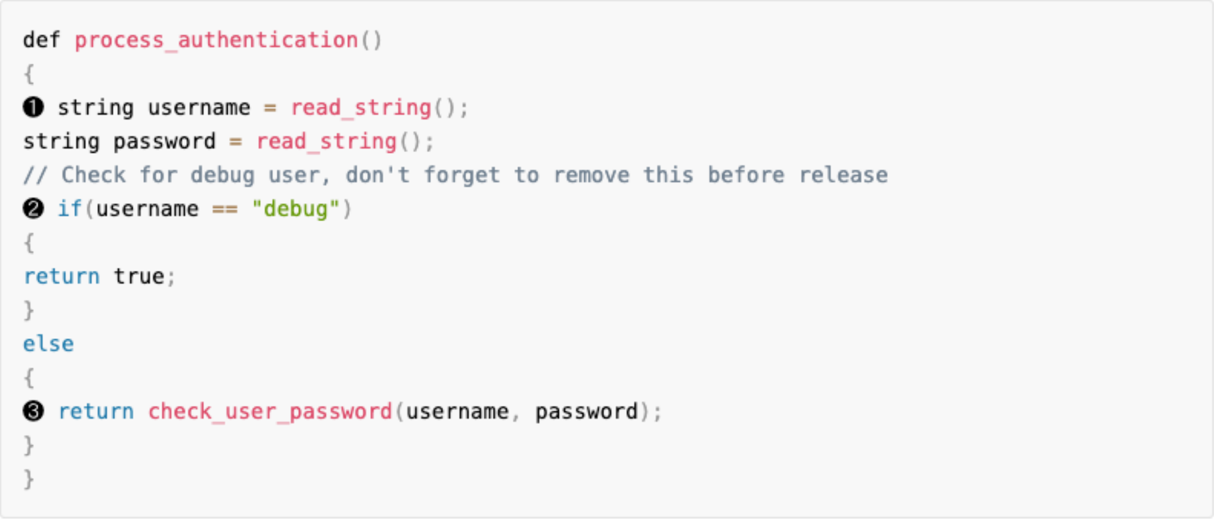
  Application crash → Denial-of-service condition

  应用程序崩溃→拒绝服务状态

Default or Hardcoded Credentials

默认或硬编码凭证

- Purpose
  - Debugging purposes
  - Intentional backdoor
- Case Study



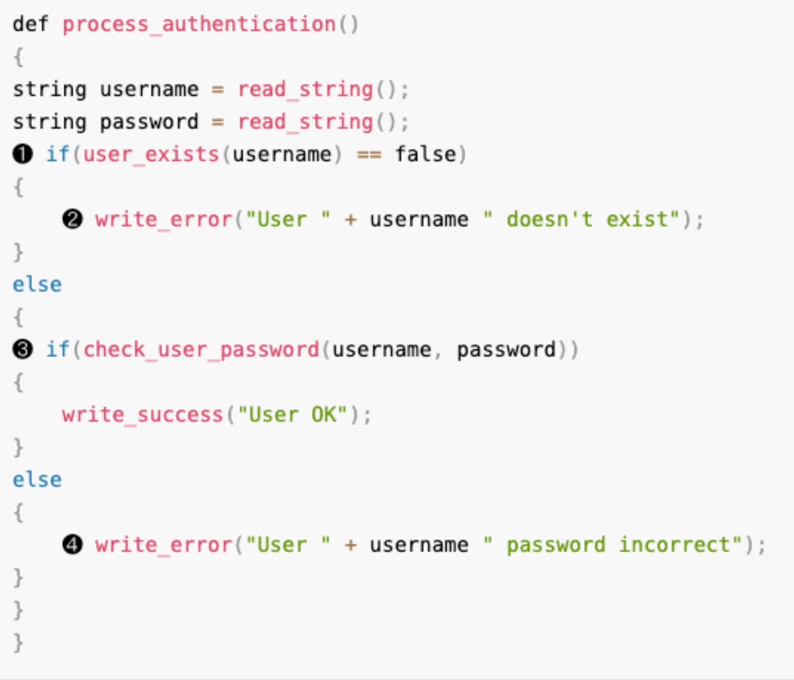
```
def process_authentication()
{
❶ string username = read_string();
string password = read_string();
// Check for debug user, don't forget to remove this before release
❷ if(username == "debug")
{
return true;
}
else
{
❸ return check_user_password(username, password);
}
}
```

User Enumeration 用户枚举

- Use usernames to control access to resources
  - Authentication →Username + Password.

- By identifying valid user accounts
  - Brute force passwords.

- Case Study

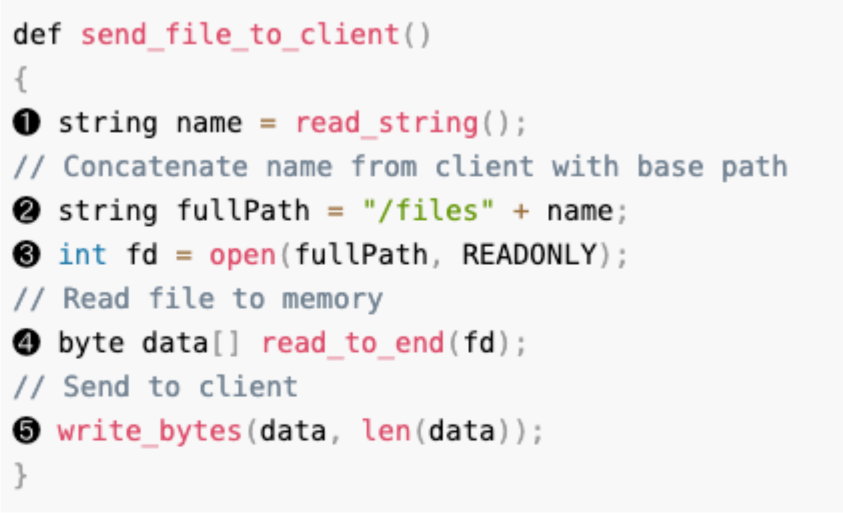


```
def process_authentication()
{
string username = read_string();
string password = read_string();
❶ if(user_exists(username) == false)
{
   ❷ write_error("User " + username " doesn't exist");
}
else
{
❸ if(check_user_password(username, password))
{
   write_success("User OK");
}
else
{
   ❹ write_error("User " + username " password incorrect");
}
}
}
```

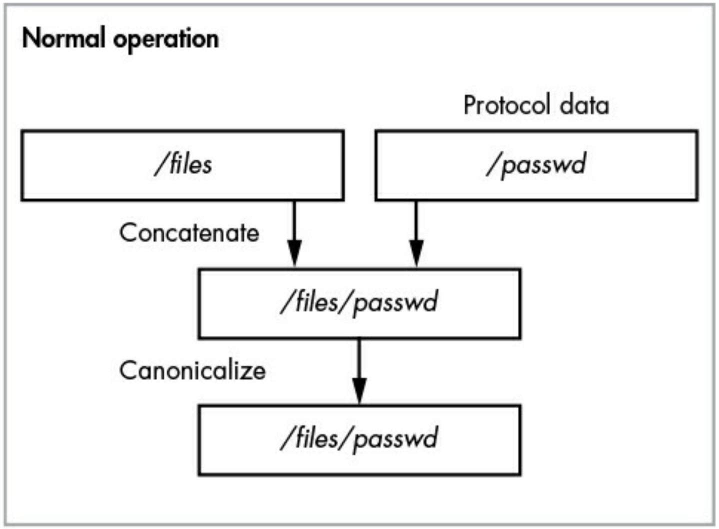
Canonicalization

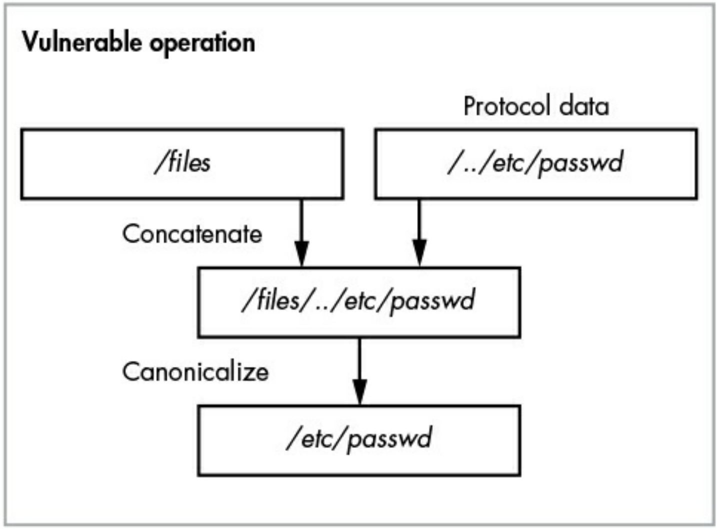
标准化

- Remote file protocol
  - Take a path supplied by a remote user
  - Concatenate it with a base directory
- Case Study



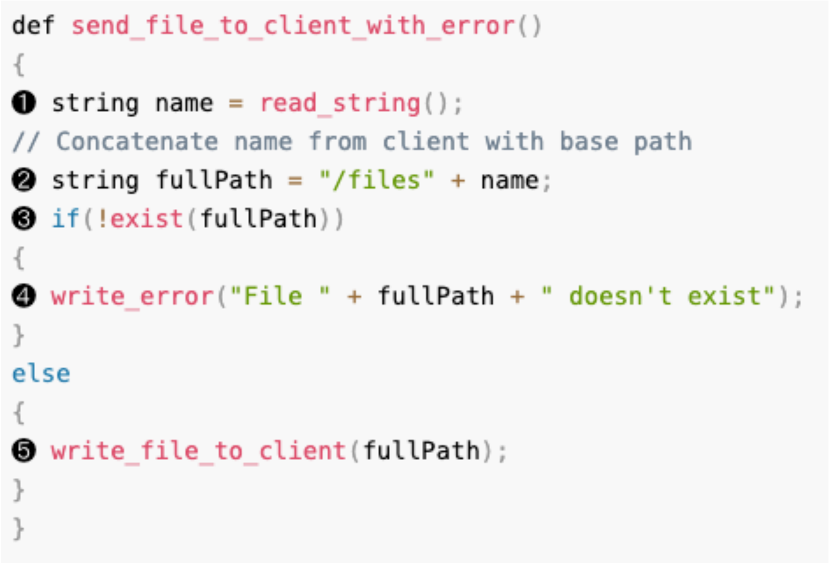
```
def send_file_to_client()
{
❶ string name = read_string();
// Concatenate name from client with base path
❷ string fullPath = "/files" + name;
❸ int fd = open(fullPath, READONLY);
// Read file to memory
❹ byte data[] read_to_end(fd);
// Send to client
❺ write_bytes(data, len(data));
}
```





Verbose Errors

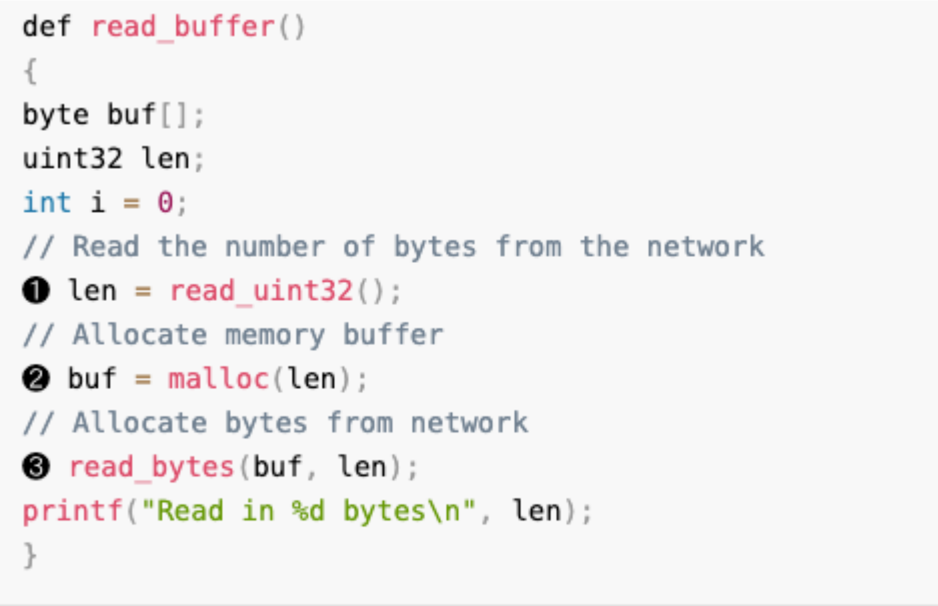
- Error information
  - Inserts local information about the resource being
- Case Study



```
def send_file_to_client_with_error()
{
❶ string name = read_string();
// Concatenate name from client with base path
❷ string fullPath = "/files" + name;
❸ if(!exist(fullPath))
{
❹ write_error("File " + fullPath + " doesn't exist");
}
else
{
❺ write_file_to_client(fullPath);
}
}
```

Memory Exhaustion Attacks

- System resources are finite
  - Memory, disk and CPU
- Allocates memory dynamically based on an absolute value transmitted in the protocol
- Case Study



```
def read_buffer()
{
byte buf[];
uint32 len;
int i = 0;
// Read the number of bytes from the network
❶ len = read_uint32();
// Allocate memory buffer
❷ buf = malloc(len);
// Allocate bytes from network
❸ read_bytes(buf, len);
printf("Read in %d bytes\n", len);
}
```

Storage Exhaustion Attacks

- Embedded systems or devices without Storage
  - The application or others on that system could begin failing
  - Prevent the system from rebooting

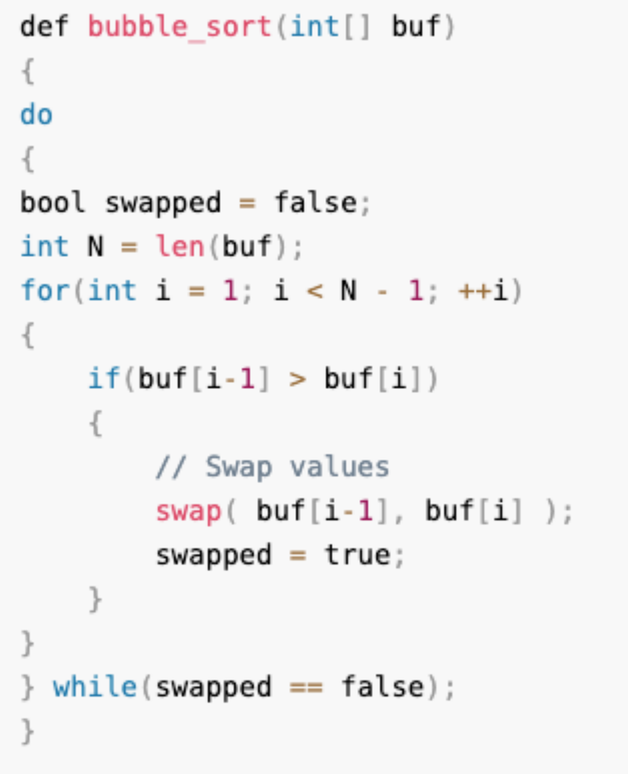
- Most common cause
  - The logging of operating information to disk.

CPU Exhaustion Attacks

- Algorithmic Complexity
  - Algorithms have an associated computational cost
  - The more work an algorithm requires, the more time,it needs from the system's processor

- Buddle Sort
  - Best case → O(N)
  - Worst case → $O(N^2)$

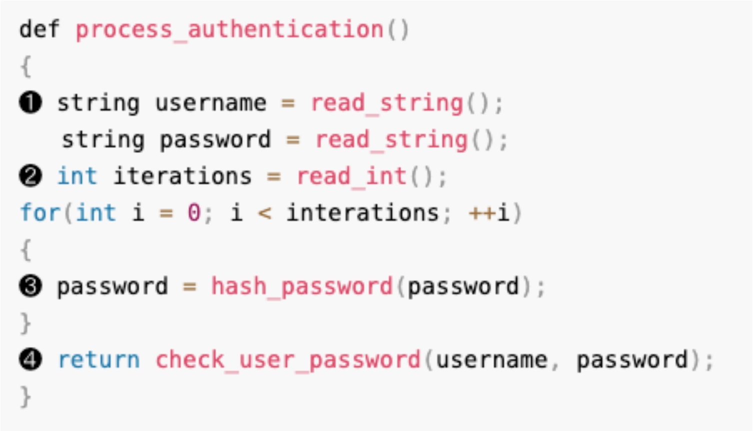
- Specify a large number of reverse sor
  - Denial-of-service



```
def bubble_sort(int[] buf)
{
do
{
bool swapped = false;
int N = len(buf);
for(int i = 1; i < N - 1; ++i)
{
    if(buf[i-1] > buf[i])
    {
        // Swap values
        swap( buf[i-1], buf[i] );
        swapped = true;
    }
}
} while(swapped == false);
}
```

CPU Exhaustion Attacks

- Configurable Cryptography
  - Cryptographic primitives processing, such as hashing algorithms, can also create a significant amount of computational workload
  - Store the hash value of the password
    Run the hashing operation multiple times increases computational cost
  - Case Study

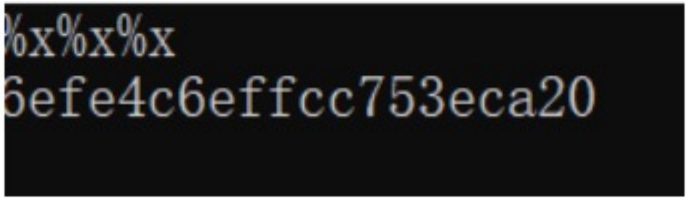


```
def process_authentication()
{
❶ string username = read_string();
   string password = read_string();
❷ int iterations = read_int();
for(int i = 0; i < interations; ++i)
{
❸ password = hash_password(password);
}
❹ return check_user_password(username, password);
}
```

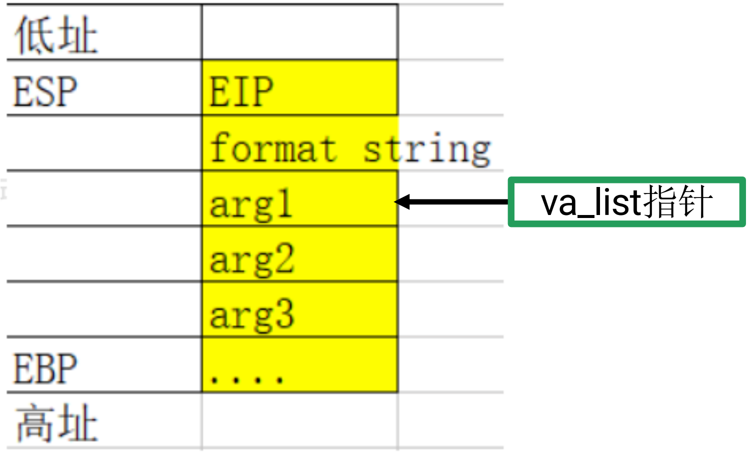
Format String Vulnerabilities

- User input contains some formatted control characters
- C language's printf and its variants
  - ➢ Printf (Error use)

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
   char a[100];
   scanf("%s",a);
   printf(a);
   return 0;
}
```



```
%x%x%x
6efe4c6effcc753eca20
```

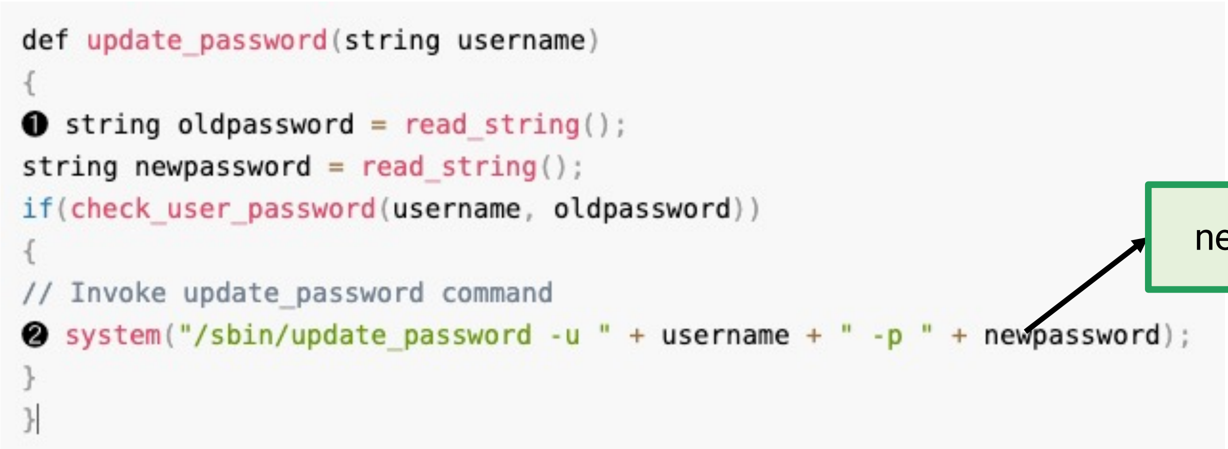


低址	
ESP	EIP
	format string
	arg1
	arg2
	arg3
EBP	....
高址	

Command Injection

命令注入

- Unix-based Oses
  - Include a rich set of utilities designed for various tasks
- The command line contains some data from the network client
- Case Study

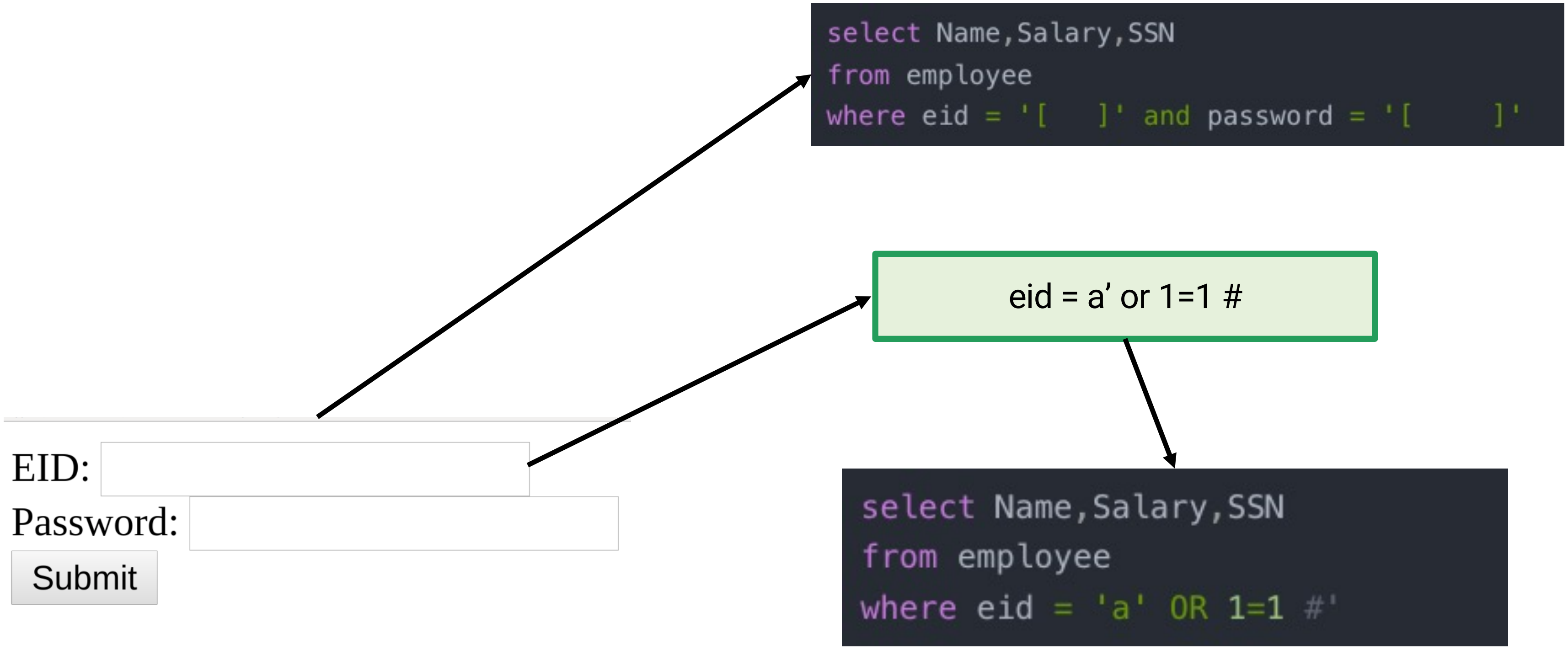


```
def update_password(string username)
{
❶ string oldpassword = read_string();
string newpassword = read_string();
if(check_user_password(username, oldpassword))
{
// Invoke update_password command
❷ system("/sbin/update_password -u " + username + " -p " + newpassword);
}
}
```

newpasswod=password; xcalc

SQL Injection

- Relational Database
  - Persistently store and retrieve data
- Structured Query Language (SQL)
  - Text-based language
  - Build queries using string operations.
- Case Study



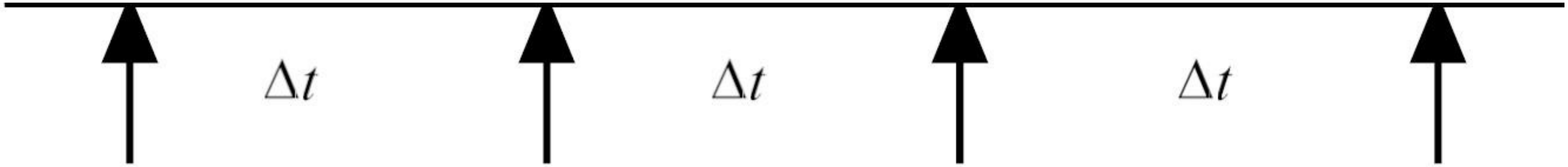
```
select Name,Salary,SSN
from employee
where eid = '[    ]' and password = '[      ]'
```

eid = a’ or 1=1 #

```
select Name,Salary,SSN
from employee
where eid = 'a' OR 1=1 #'
```

Vulnerabilities, Exploits and Attacks

漏洞、利用及攻击

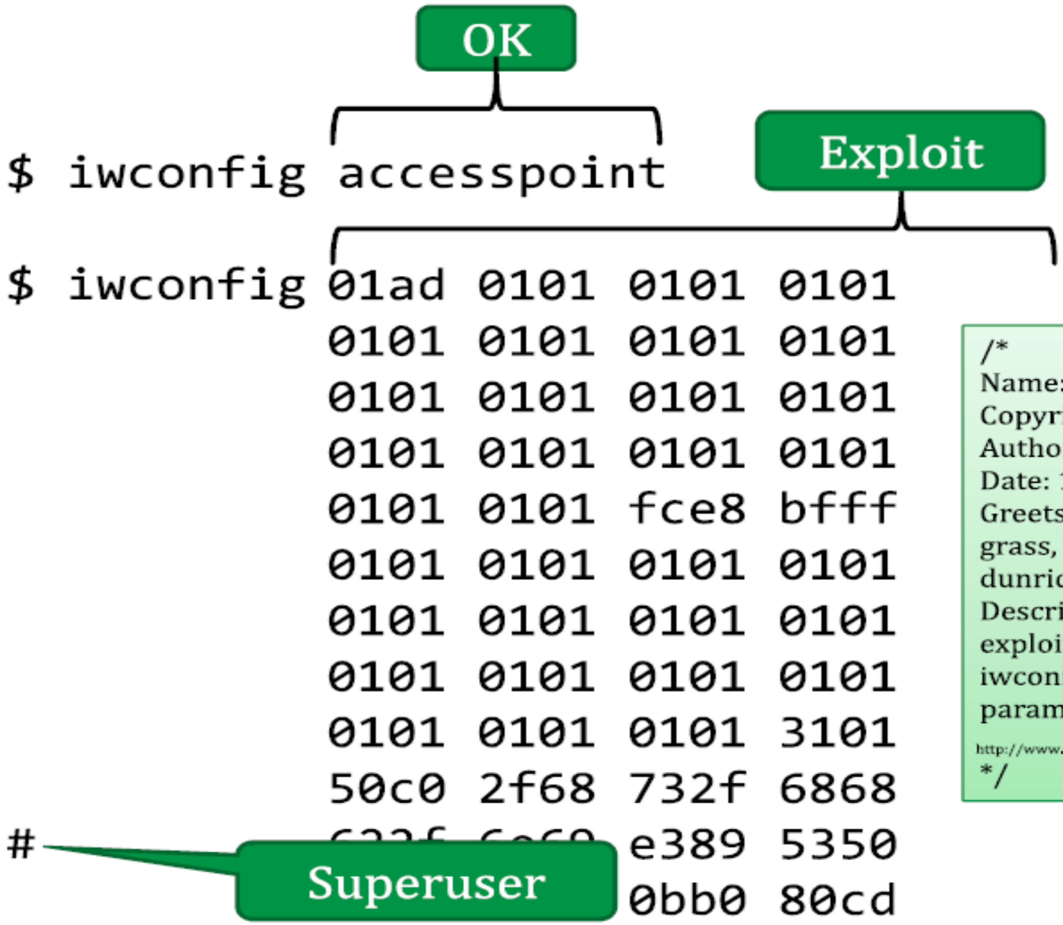


Δt Δt Δt

Vulnerability Exploit Attack Code Attacks

Design Vulnerability
Implementation Vulnerability
Configuration Vulnerability

Vulnerability and Exploits



OK

Exploit

Superuser

```
$ iwconfig accesspoint

$ iwconfig 01ad 0101 0101 0101
           0101 0101 0101 0101
           0101 0101 0101 0101
           0101 0101 0101 0101
           0101 0101 fce8 bfff
           0101 0101 0101 0101
           0101 0101 0101 0101
           0101 0101 0101 0101
           0101 0101 0101 3101
           50c0 2f68 732f 6868
#          622f 6e69 e389 5350
                     0bb0 80cd
```

```
/*
Name: iw-config.c
Copyright: !sh2k+!tc2k
Author: heka
Date: 11/11/2003
Greets: bx, pintos, eksol, hex, keyhook,
grass, toolman, rD, shellcode,
dunric, termid, kewlcat, JiNKS
Description: /sbin/iwconfig – local root
exploit
iwconfig manipulate the basic wireless
parameters
http://www.securityfocus.com/archive/1/344272/2003-11-10/2003-11-16/0
*/
```

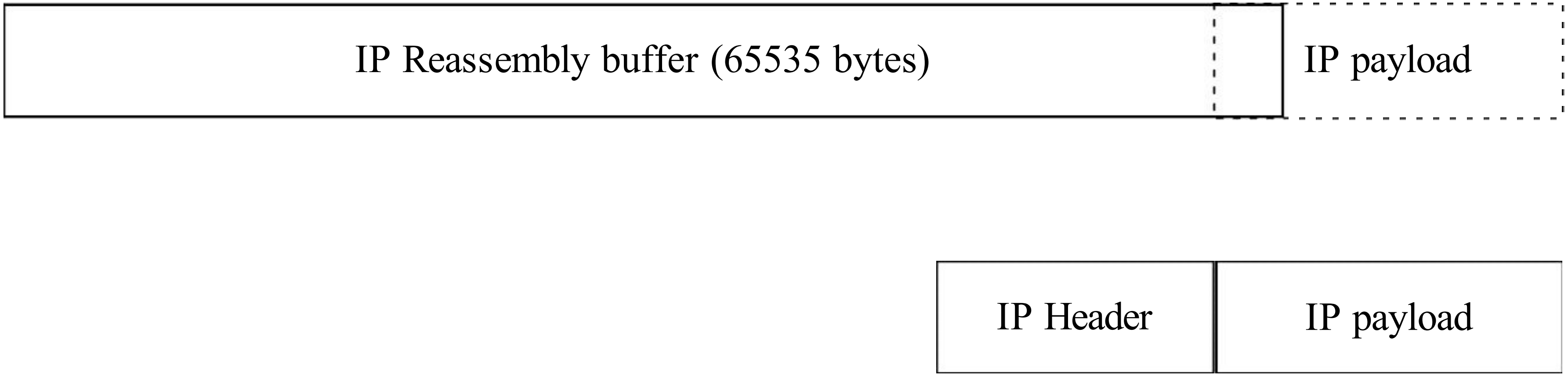
Network Security Taxonomy
网络安全分类

- Header based 基于头部的漏洞和攻击
- Protocol based 基于协议的漏洞和攻击
- Authentication based 基于验证的漏洞和攻击
- Traffic Based 基于流量的漏洞和攻击

Header Based 基于头部的漏洞和攻击

- Creation of invalid packets, different protocols handle bad packets differently 创建无效报文，不同协议对坏报文的处理方式不同
- Source and destination address manipulation 源和目标地址操作
  - Device can be confused by setting source and destination to the same address 如果将源地址和目的地址设置为相同的地址，可能会导致设备混淆
- Setting bits in the header that should not be set 设置头中不应该设置的位
- Putting values in the header that are above or below the level specified in the standard 在标头中放置高于或低于标准中指定的级别的值

Example: Ping of Death



IP Reassembly buffer (65535 bytes)

IP payload

IP Header

IP payload

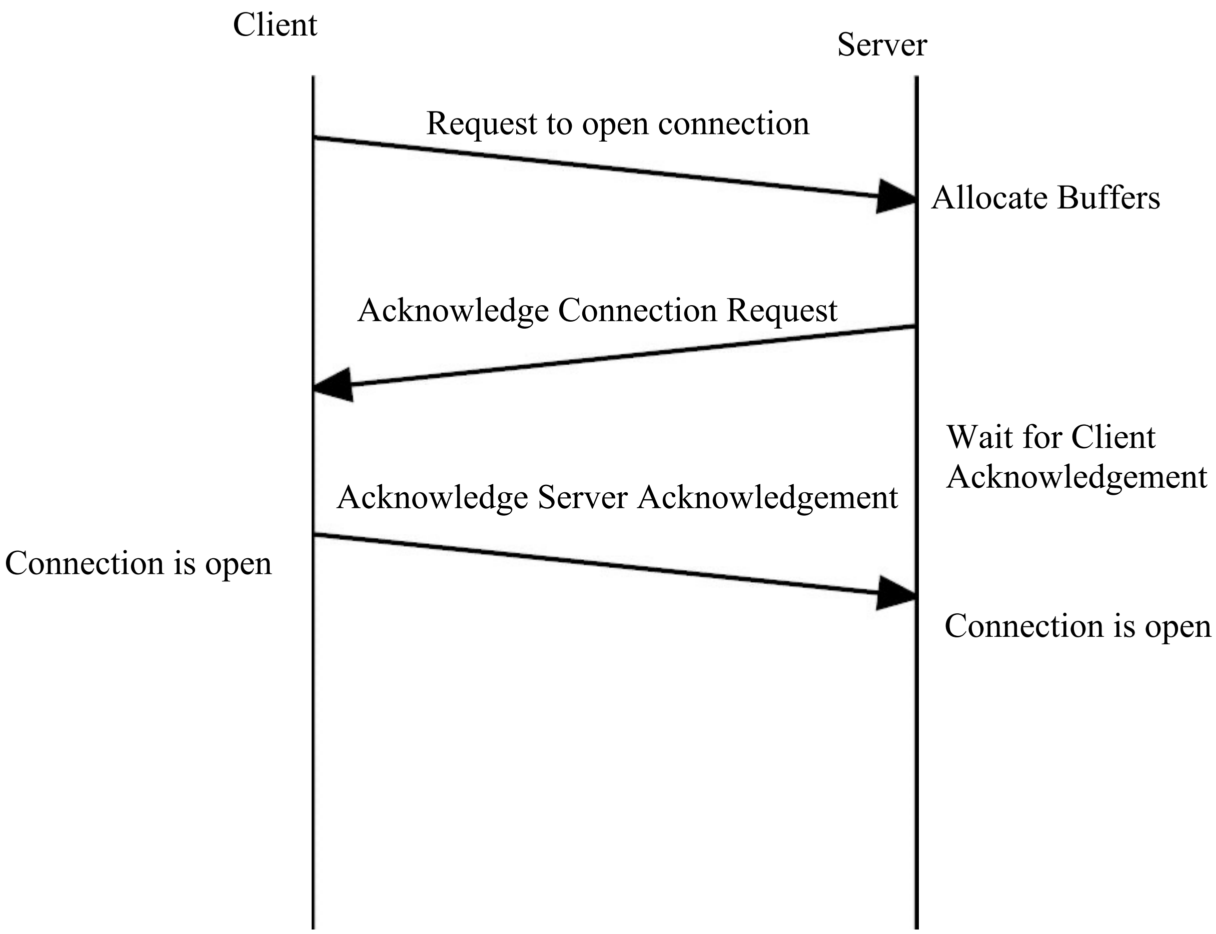
offset = 65528 (max value)
length = 100

Protocols attacks

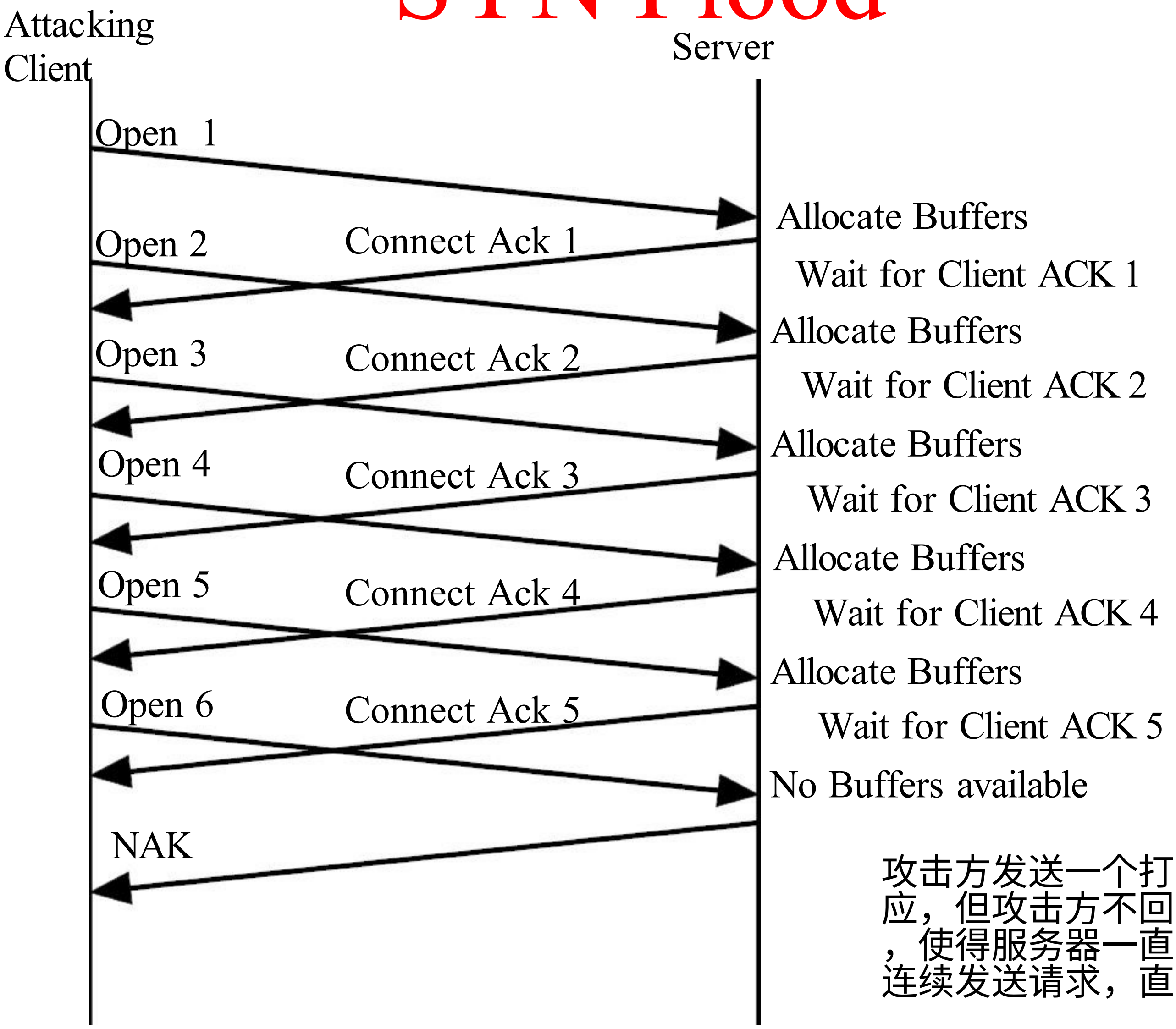
- You can shutdown the protocol itself 您可以关闭协议本身
- Send packets telling the device to stop talking 发送数据包告诉设备停止通话
- For connectionless protocols you can answer as the server and tell the client the server is down. 对于无连接协议，您可以作为服务器应答并告诉客户端服务器已关闭

Example: Syn Flood

- TCP 3-way Handshake


Client
Server
Request to open connection
Allocate Buffers
Acknowledge Connection Request
Wait for Client Acknowledgement
Acknowledge Server Acknowledgement
Connection is open
Connection is open

SYN Flood

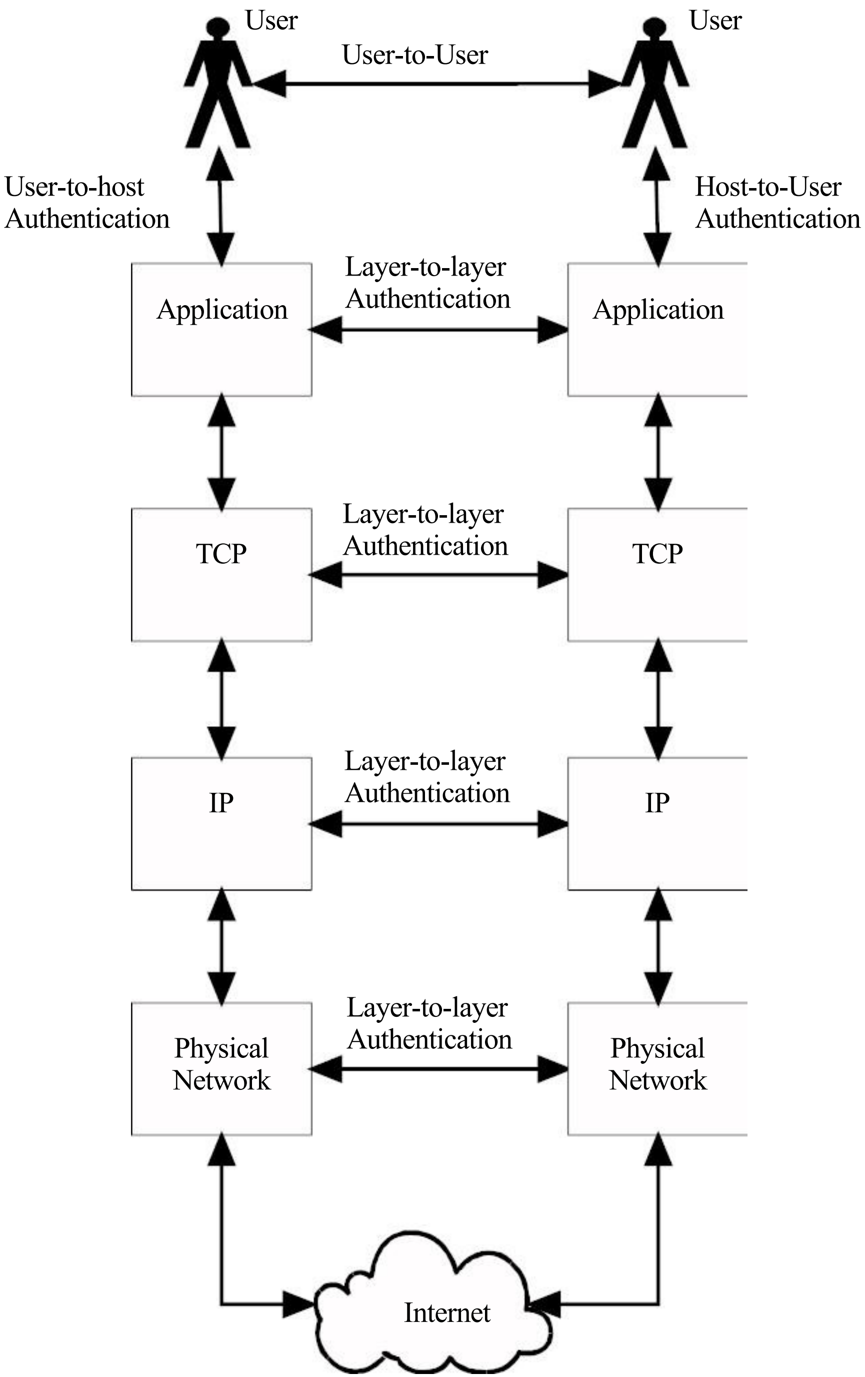


攻击方发送一个打开连接请求，服务器回应，但攻击方不回应服务器完成三次握手，使得服务器一直处于打开状态。攻击者连续发送请求，直到服务器缓冲区满

Authentication-Based

- Authentication is the proof of one’s identity to another. 身份验证是一个人对另一个人的身份证明
- Often thought of as username & password based 通常认为是基于用户名和密码
- In a network addresses are often used to authenticate packets. 在网络中，地址经常被用来验证数据包
  - Like the 4 addresses used to identify a packet in the Internet 比如在因特网上用来识别一个数据包的4个地址

Network Authentication



Authentication

- Four different types of authentication 四种不同类型的身份验证
  - User to host 用户到主机
    - Person proves the identity to computer resource 人向计算机资源证明身份
    - Most prevalent 非常盛行
  - Host to Host 主机到主机 （两个应用两个主机两个应用层之间的验证，通常借助主机地址或应用程序地址）
    - Work being done to strengthen this
    - In past usually done by IP address
  - User to User 用户到用户
    - Contracts, secure email 合同，安全电子邮件
    - Useful for online auctions 适用于在线拍卖
  - Host to User 主机到用户
    - Server authenticating to user

Traffic-Based

- Too much data
  - To a single:
    - Application
    - Network device网络设备
    - Protocol layer
  - From:
    - Multiple machines
    - Single attackers
- Traffic Capture (sniffing) 流量捕获(嗅探)

Traffic Attacks

- You can shutdown a service by:
  - – flooding it with packets
  - – opening a large number of connections

您可以通过以下方法关闭服务:
-充斥着数据包
-打开大量的连接

- You can shutdown network by:
  - – flooding it with a large number of packets.
  - – Broadcast packets will do the most damage

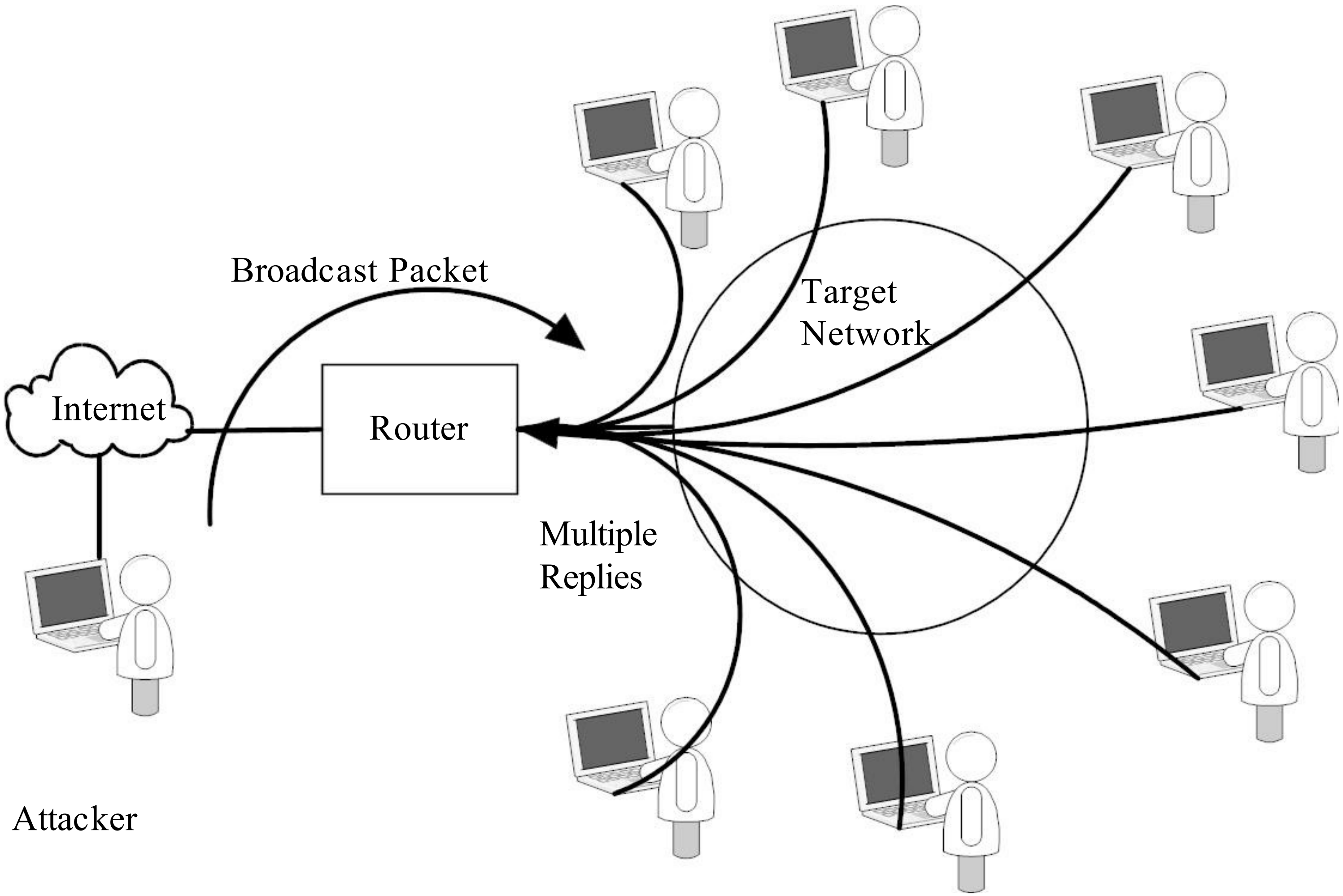
您可以通过以下方式关闭网络:
-用大量的数据包淹没它。
—广播数据包伤害最大

- You can shutdown a machine by:
  - – flooding a machine with packets on multiple services
  - – Broadcast storms

你可以通过以下方法关闭机器:
-用多个服务的数据包淹没一台机器
-广播风暴

Broadcast Flood Attack

广播洪水攻击



Traffic Capture 流量捕获

- Packet sniffing can be played out against any layer in the network if the attacker is in a position to “see” the traffic.

如果攻击者处于 “ 看到 ” 流量的位置，就可以对网络中的任何层进行包嗅探