实验安排

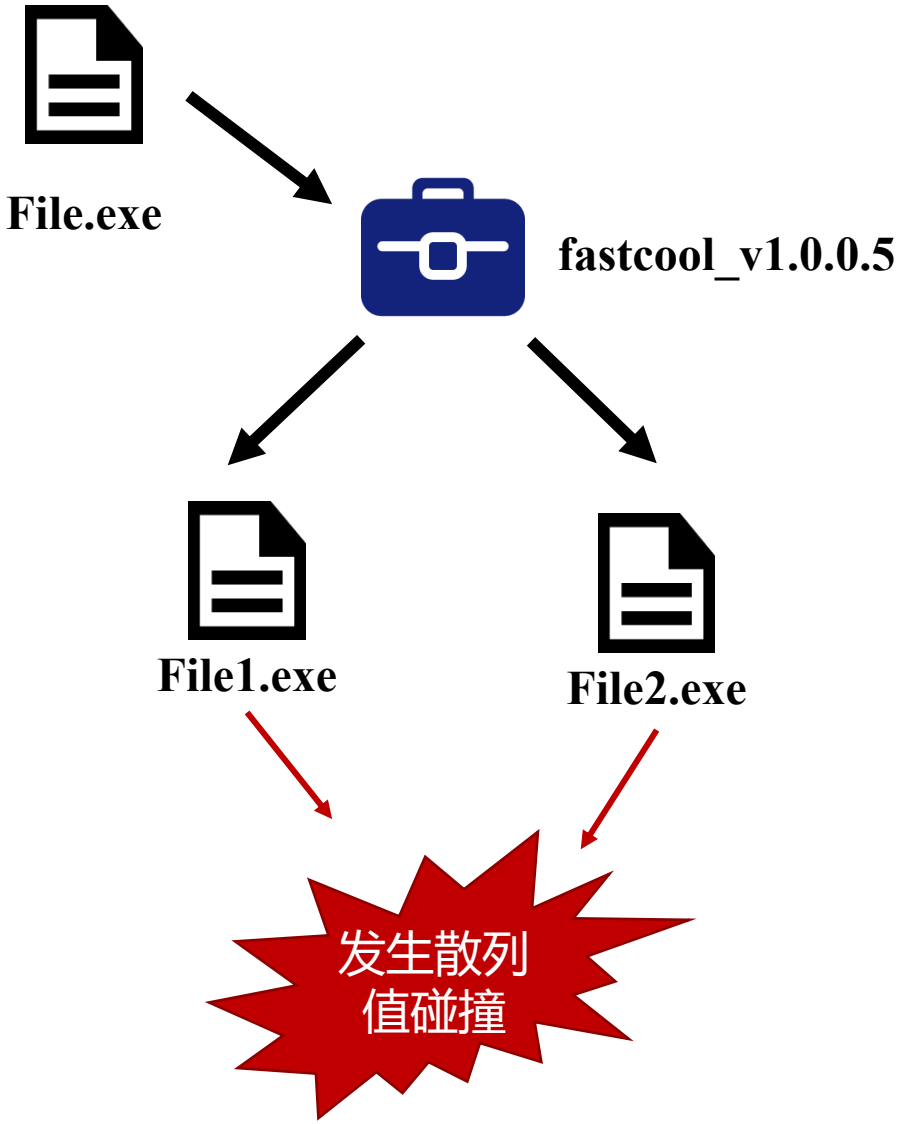
实验一：制造MD5算法的散列值碰撞 难度: ★☆☆

- **实验目的：**

利用散列校验文件的完整性，并利用散列碰撞工具产生散列碰撞现象

- **实验要求：**

在Windows环境下调用散列碰撞生成工具，生成散列码相同的两个文件



实验二：基于口令的安全身份验证协议 难度: ★★★

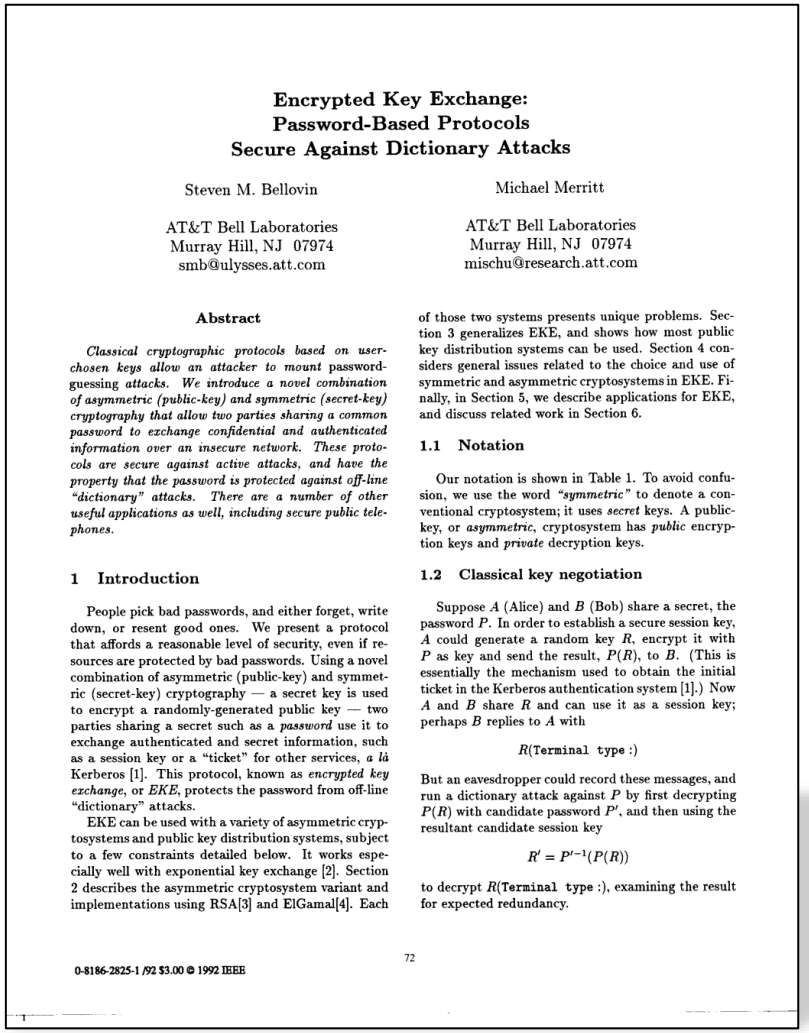
- **实验背景：**

口令是最原始的登录方式，虽然手机验证码登录盛行，但口令验证协议仍被京东、淘宝等主流电商平台采用

Bellovin-Merritt 协议诞生于贝尔实验室，被发表于92年的IEEE S&P，是早期口令安全协议的代表

- **实验目的：**

通过完整实现该身份认证协议，体验实现密码学库的乐趣，加深对身份验证安全协议的理解



Encrypted Key Exchange:
Password-Based Protocols
Secure Against Dictionary Attacks

Steven M. Bellovin
AT&T Bell Laboratories
Murray Hill, NJ 07974
smb@ulysses.att.com

Michael Merritt
AT&T Bell Laboratories
Murray Hill, NJ 07974
mischu@research.att.com

Abstract

Classical cryptographic protocols based on user-chosen keys allow an attacker to mount password-guessing attacks. We introduce a novel combination of asymmetric (public-key) and symmetric (secret-key) cryptography that allow two parties sharing a common password to exchange confidential and authenticated information over an insecure network. These protocols are secure against active attacks, and have the property that the password is protected against off-line "dictionary" attacks. There are a number of other useful applications as well, including secure public telephones.

1 Introduction

People pick bad passwords, and either forget, write down, or resent good ones. We present a protocol that affords a reasonable level of security, even if resources are protected by bad passwords. Using a novel combination of asymmetric (public-key) and symmetric (secret-key) cryptography — a secret key is used to encrypt a randomly-generated public key — two parties sharing a secret such as a *password* use it to exchange authenticated and secret information, such as a session key or a "ticket" for other services, *a là* Kerberos [1]. This protocol, known as *encrypted key exchange*, or *EKE*, protects the password from off-line "dictionary" attacks.

EKE can be used with a variety of asymmetric cryptosystems and public key distribution systems, subject to a few constraints detailed below. It works especially well with exponential key exchange [2]. Section 2 describes the asymmetric cryptosystem variant and implementations using RSA[3] and ElGamal[4]. Each of those two systems presents unique problems. Section 3 generalizes EKE, and shows how most public key distribution systems can be used. Section 4 considers general issues related to the choice and use of symmetric and asymmetric cryptosystems in EKE. Finally, in Section 5, we describe applications for EKE, and discuss related work in Section 6.

1.1 Notation

Our notation is shown in Table 1. To avoid confusion, we use the word *"symmetric"* to denote a conventional cryptosystem; it uses *secret* keys. A public-key, or *asymmetric*, cryptosystem has *public* encryption keys and *private* decryption keys.

1.2 Classical key negotiation

Suppose A (Alice) and B (Bob) share a secret, the password P. In order to establish a secure session key, A could generate a random key R, encrypt it with P as key and send the result, $P(R)$, to B. (This is essentially the mechanism used to obtain the initial ticket in the Kerberos authentication system [1].) Now A and B share R and can use it as a session key; perhaps B replies to A with

$$R(\texttt{Terminal type :})$$

But an eavesdropper could record these messages, and run a dictionary attack against P by first decrypting $P(R)$ with candidate password P', and then using the resultant candidate session key

$$R' = P'^{-1}(P(R))$$

to decrypt $R(\texttt{Terminal type :})$, examining the result for expected redundancy.

0-8186-2825-1 /92 $3.00 © 1992 IEEE

实验二：基于口令的安全身份验证协议 难度: ★★★

- **实验要求：**

根据交互图完全实现该协议：

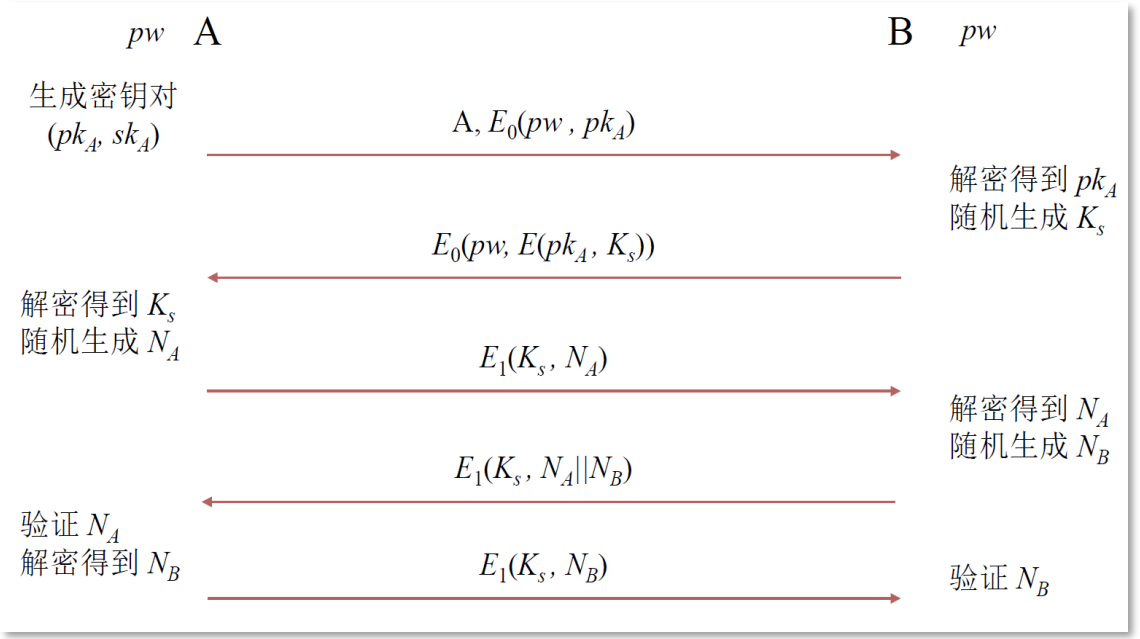
1. 通过TCP套接字进行传输， 用2个进程模拟交互流程，建立安全信道，并加密传输信息
2. 不限定编程语言，具体对称/非对称加密方案自选，可以使用AES/RSA等基础密码学库中的实现

- **实验资料：**

1. Bellovin-Merritt 协议原始文章：

https://ieeexplore.ieee.org/document/213269

Bellovin-Merritt 协议的交互流程图



实验三：数字证书综合使用

难度：★☆☆

- **实验目的：**

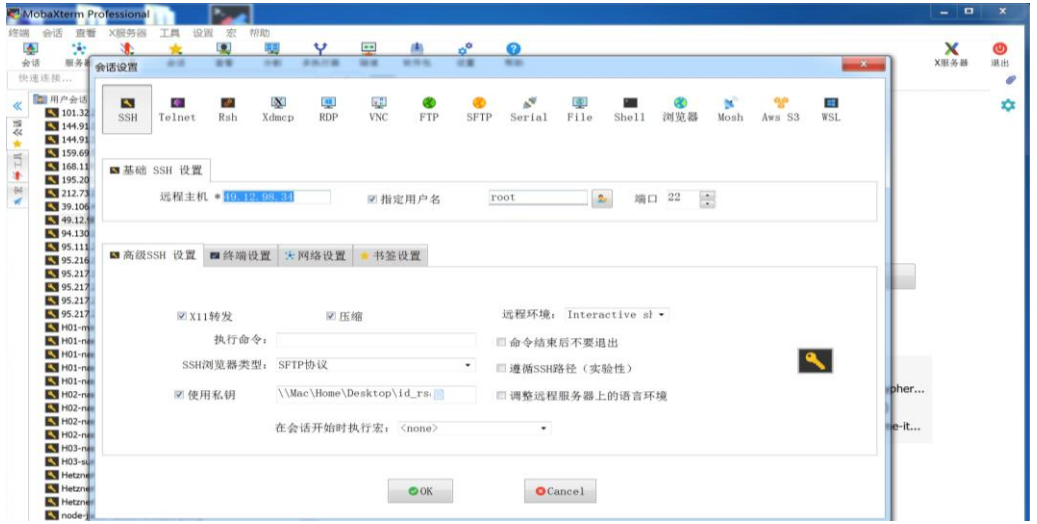
掌握PKI的原理和证书使用，加深对网络安全问题的认识，了解如何增加通信安全性

- **实验要求：**

1. 通过OpenSSH工具生成公私钥对，并能将公钥传到自己的远程服务器，使用密钥对通过ssh的方式安全访问远程服务器
2. 编写一个属于自己的简单网站，并通过nginx或者apache添加https协议，如果有域名可以通过阿里云申请对应的证书

```
→  key01 ls
id_rsa.server      id_rsa.server.pub
→  key01 cat id_rsa.server.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCt0bSpltaViz+haggLZi2PNAm8DhDw0d29Z/6w0yNo
Fwi4cwfm+eG3auut52OYmSMF3RbB5GzBeEOaevohBP7qWX5ohCAB5JBI8fdGvIJiY1vmOhCwjFmMpds5
iiS+vUBat4LzgwMSgYV/RfJVRP1wz4eQQxf4QWOaBfajPwVDB2nnd5Zb4NrYkE31062oYDWRt/otc6TP
T7IyEdOtGqAUrncxHG/dmeIKOXtuJLulaFJGSOOwEevHZkD8JJv2lzcb0oI8fd0ArEujvqeRATm+KGrd
xIzP1EWxtML/Aq1Tk0C02A1VCZELHFBx+puSvrkVXqbuxekF7KRGO5hwjkPP haizhitiantang1@163
.com
```

生成的公私钥对



为服务器配置公私钥访问

为网站配置https证书

实验四：同态加密匿名投票

难度：★☆☆

- **实验背景：**

在电子投票系统中，由于统计票数是使用加法累加票数进行统计的，因此具有加法同态性质的 Paillier 算法可被应用于实现匿名的电子投票系统，以保护投票人的投票信息

- **实验目的：**

基于Paillier算法的匿名电子投票系统可实现匿名的投票本地模拟程序

- Paillier算法具有加法同态性
- 密文相乘结果解密与明文相加结果相同

密钥生成

随机选取两个大素数p和q

计算$N = pq$和$\lambda = lcm(p-1, q-1)$

随机选取一个小于N^2的正整数g，并保证

$\gcd\left(L(g^{\lambda} \bmod N^2), N\right) = 1$，其中$L(x) = \frac{x-1}{N}$

公钥为(N, g)，私钥为λ

加密

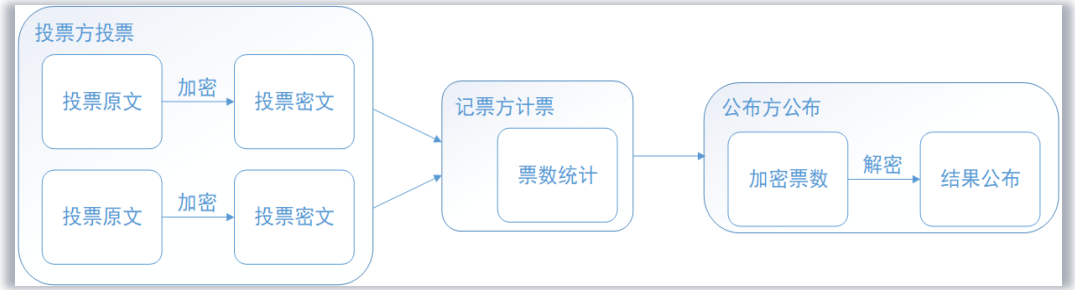
已知明文$m < N$，随机选取$r < N$

密文$c = g^m r^N mod N^2$

解密

已知密文$c < N^2$，明文$m = \frac{L(c^{\lambda} mod N^2)}{L(g^{\lambda} mod N^2)} mod\ N$

```
请输入第一个明文：123
请输入第二个明文：456
对第一个明文加密后得到密文：21074939849326247864279619
对第二个明文加密后得到密文：94145804565136665960095689
两密文相乘得到：19841171682766797052436847925323719
密文相乘后解密得到的明文为：579
```



实验五： Spectre 攻击验证

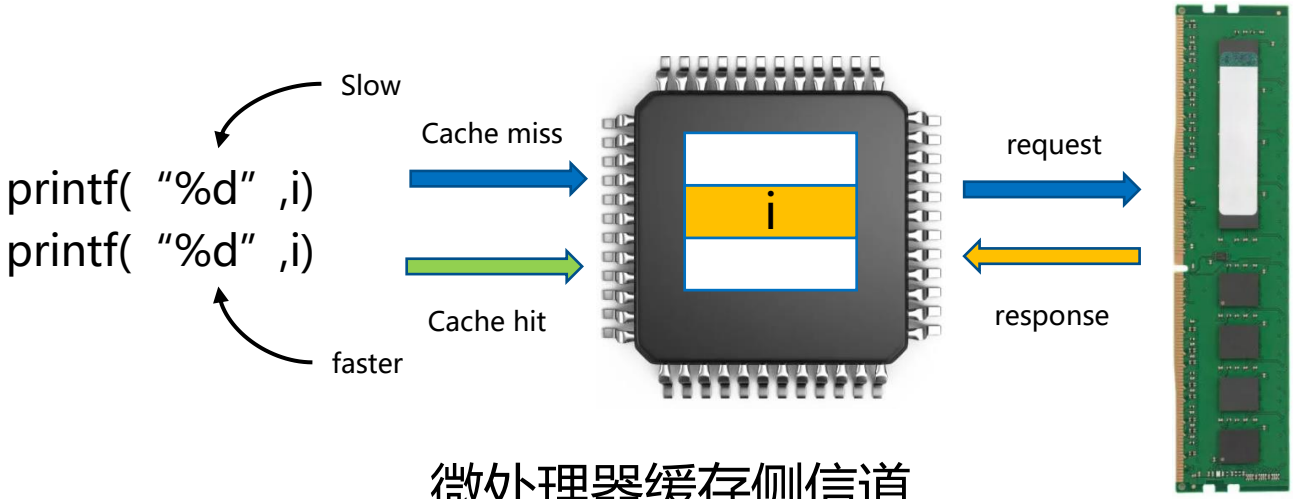
难度: ★★★

- **实验背景：**

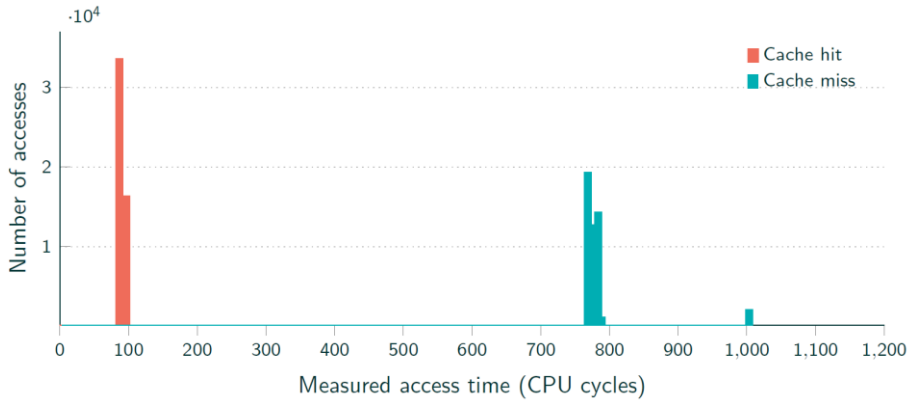
微处理器架构侧信道严重损害内存的隔离性，泄露用户隐私信息

- **实验目的：**

通过实现Spectre攻击样例，学习了解CPU性能优化技术（分支预测）带来的安全问题，进一步理解CPU的工作进程，加深对处理器硬件安全的认识



微处理器缓存侧信道



微处理器缓存命中情况

实验五： Spectre 攻击验证

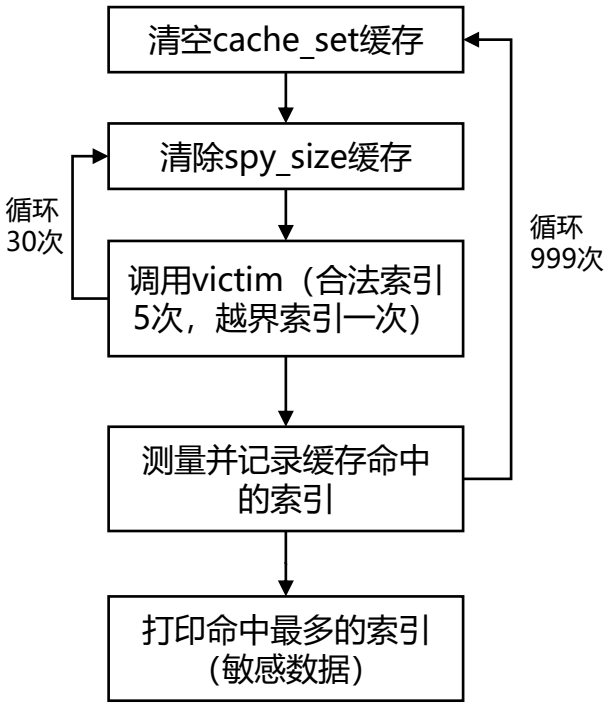
难度: ★★★

- **实验要求：**

1. 根据发现Spectre的文章附录当中的源代码复现文中 Section IV 中的实验，窃取内存当中的机密信息
2. 要求为C语言版本，可以参考其他源码但必须标明出处，且实验效果需要与教材一致

- **实验资料：**

1. IEEE S&P上的Spectre原始文章：
https://ieeexplore.ieee.org/abstract/document/8835233



攻击流程

```
Reading 40 bytes:
0x54='T' score=999
0x68='h' score=999
0x65='e' score=999
0x20=' ' score=999
0x4D='M' score=999
0x61='a' score=999
0x67='g' score=999
0x69='i' score=999
0x63='c' score=999
0x20=' ' score=999
0x57='W' score=999
0x6F='o' score=999
0x72='r' score=999
0x64='d' score=998
0x73='s' score=999
0x20=' ' score=999
0x61='a' score=999
0x72='r' score=999
0x65='e' score=999
0x20=' ' score=998
0x53='S' score=999
0x71='q' score=999
0x75='u' score=999
```

实验结果

实验六：简单栈溢出实验

难度：★★☆

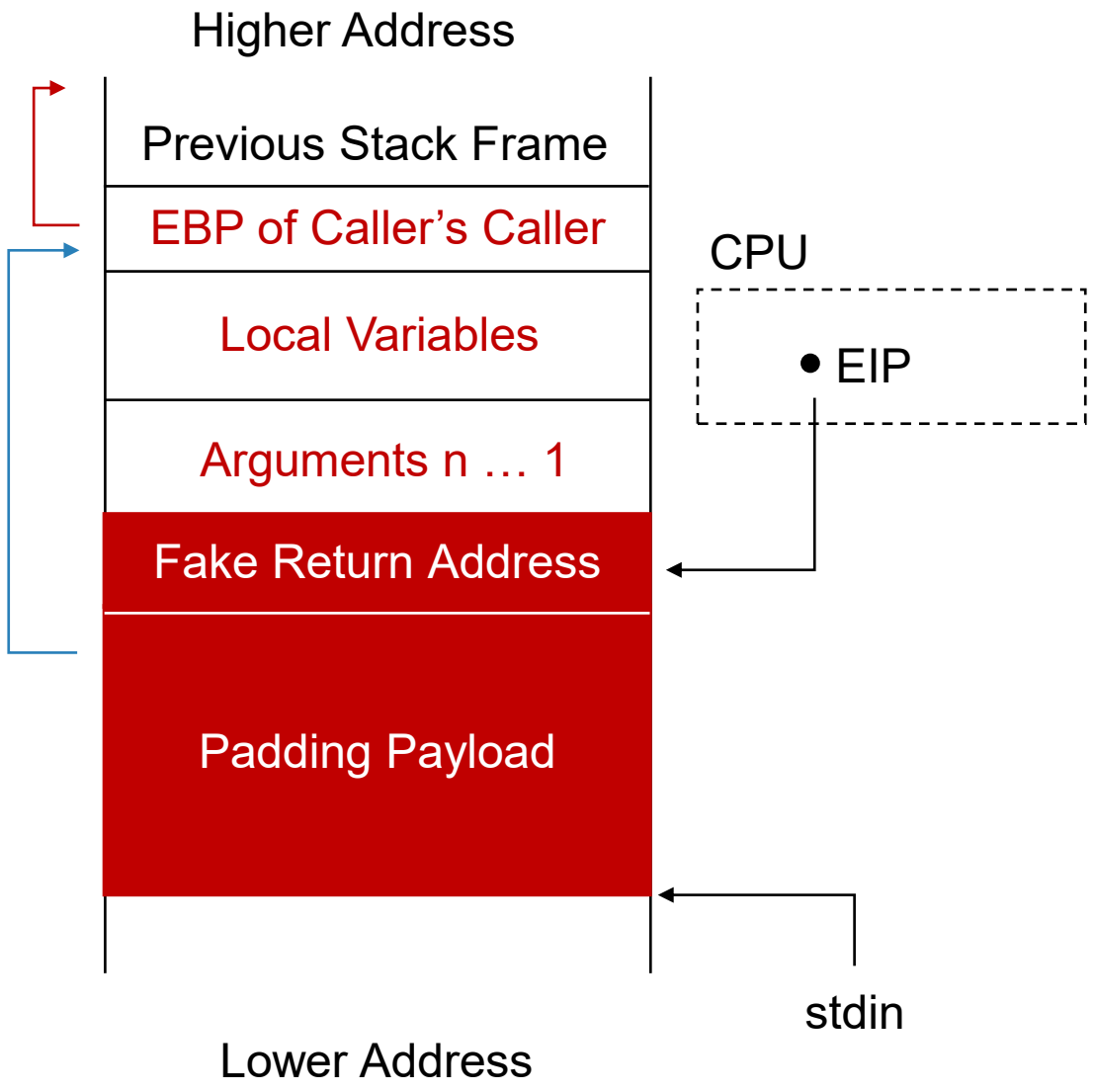
- **实验背景：**

栈区溢出攻击，是最常见的缓冲区溢出攻击方式，是多种攻击的基础

攻击者构造恶意的程序输入覆盖栈当中的返回地址，不正当触发函数执行，达到修改进程行为的目的

- **实验目的：**

本实验模拟一次朴素的栈区溢出攻击，核心在于掌握“如何构造覆盖栈帧的恶意输入”



实验六：简单栈溢出实验

难度：★★☆

- **实验要求：**

实验需要完成如下步骤：

1. 给出一个C语言受害程序，包含一个不设长度检查的 gets 函数调用

2. 关闭内存防御方案（ASLR、Stack Canary等）的条件下编译受害程序

3. 对受害程序进行反汇编，并分析汇编码，可以使用 IDA Pro这类专用的逆向分析工具

4. 编写恶意程序，构造非法输入，推荐使用 Python 的 pwntools 工具

5. 执行恶意程序，恶意执行目标函数

实验七：基于栈溢出的模拟勒索实验 难度：★★★

- **实验目的：**

尝试复现真实世界当中的Nginx的栈溢出漏洞：CVE-2013-2028

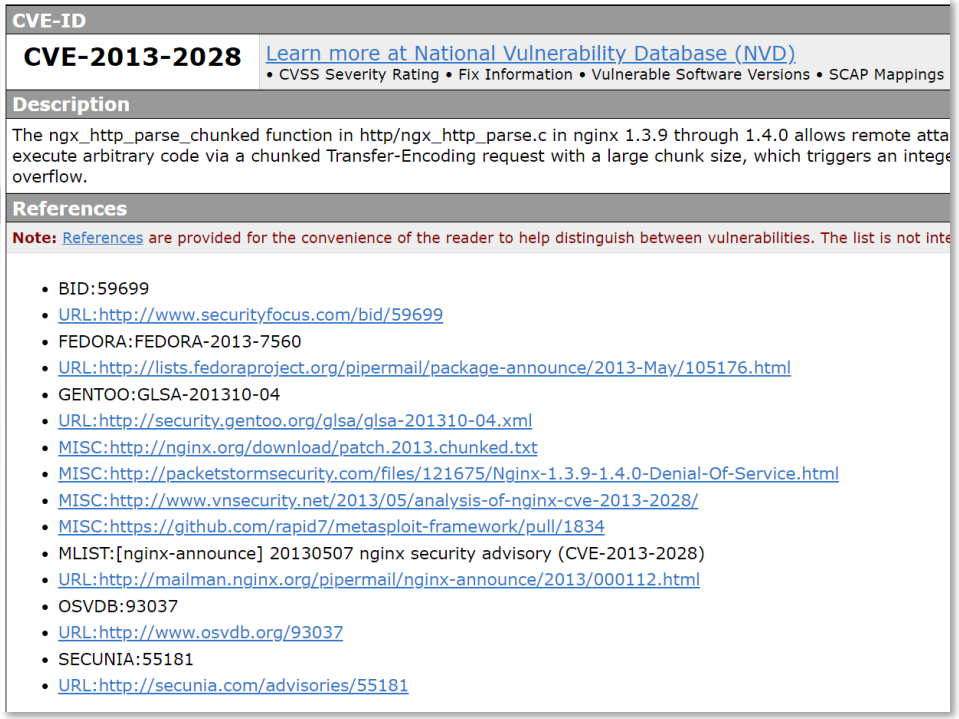
掌握在现实世界中可行的漏洞利用技巧：BROP，理解BROP如何绕过Canary、ASLR等防御机制

- **实验要求：**

1. 去CVE中检索该漏洞相关的信息
2. 部署存在漏洞的Nginx版本
3. 编写漏洞利用源代码，可使用现有工具库，亦可参考现有源代码，但需要在实验报告中标明出处，禁止抄袭源代码
4. 连接Shell，完全控制Nginx服务器，并尝试访问受害服务其文件系统

- **实验资料：**

1. **CVE：*https://cve.mitre.org/***



CVE-ID

CVE-2013-2028 Learn more at National Vulnerability Database (NVD)
• CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings

Description

The ngx_http_parse_chunked function in http/ngx_http_parse.c in nginx 1.3.9 through 1.4.0 allows remote atta execute arbitrary code via a chunked Transfer-Encoding request with a large chunk size, which triggers an intege overflow.

References

Note: References are provided for the convenience of the reader to help distinguish between vulnerabilities. The list is not inte

- BID:59699
- URL:http://www.securityfocus.com/bid/59699
- FEDORA:FEDORA-2013-7560
- URL:http://lists.fedoraproject.org/pipermail/package-announce/2013-May/105176.html
- GENTOO:GLSA-201310-04
- URL:http://security.gentoo.org/glsa/glsa-201310-04.xml
- MISC:http://nginx.org/download/patch.2013.chunked.txt
- MISC:http://packetstormsecurity.com/files/121675/Nginx-1.3.9-1.4.0-Denial-Of-Service.html
- MISC:http://www.vnsecurity.net/2013/05/analysis-of-nginx-cve-2013-2028/
- MISC:https://github.com/rapid7/metasploit-framework/pull/1834
- MLIST:[nginx-announce] 20130507 nginx security advisory (CVE-2013-2028)
- URL:http://mailman.nginx.org/pipermail/nginx-announce/2013/000112.html
- OSVDB:93037
- URL:http://www.osvdb.org/93037
- SECUNIA:55181
- URL:http://secunia.com/advisories/55181

CVE-2013-2028 检索结果

实验八： SYN洪泛攻击

难度: ★☆☆

- **实验背景：**

DDoS攻击天天都在发生，简单洪泛式DDoS攻击的代表是SYN洪泛攻击

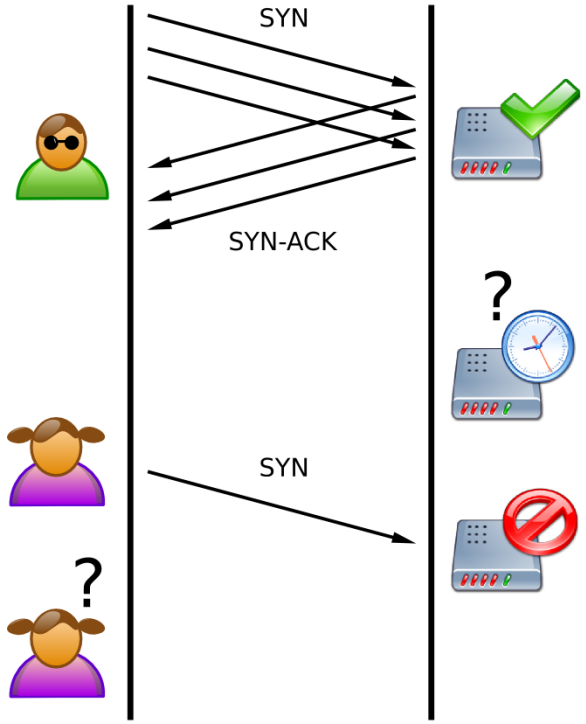
- **实验目的：**

1. 部署一台受害Web服务器，测量访问时延

2.攻击者启动 50 个线程，调用 Scapy 库，持续伪造 SYN 请求报文

3.在客户端再次启动浏览器，发现对应Web 页面无法打开（或存在很大延时）

本实验千万要注意安全



SYN洪泛攻击 示意图

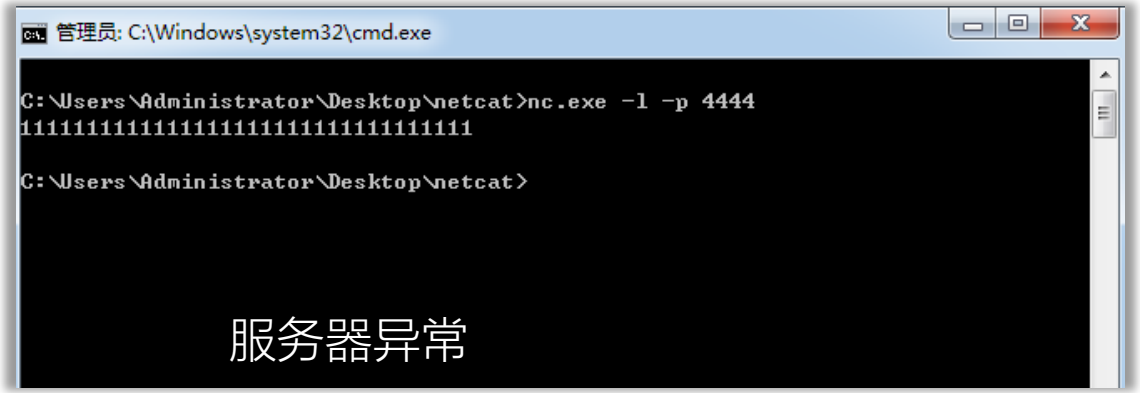
实验九：基于IPID的TCP侧信道漏洞 难度: ★★★

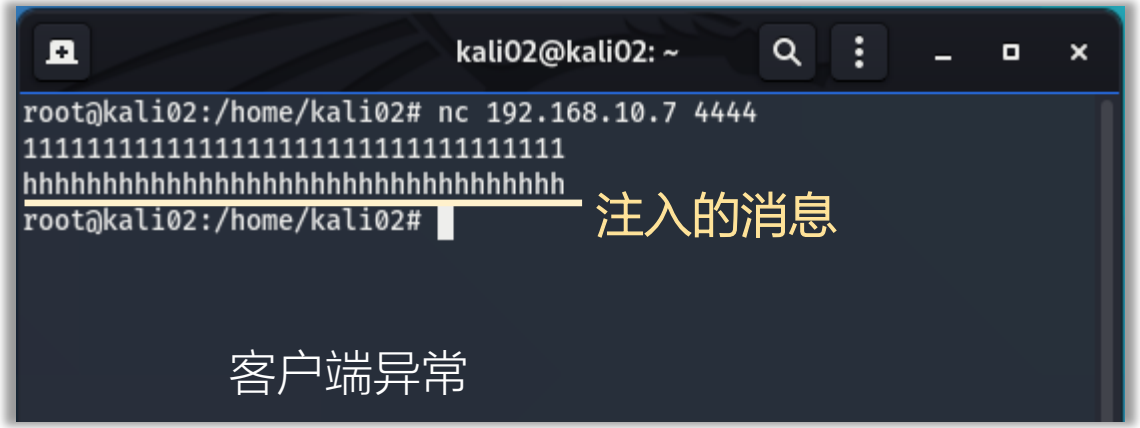
- **实验背景：**

网络侧信道攻击借助通讯内容无关信息干涉通讯的正常进行，可实现链接的强制中断或虚假信息注入

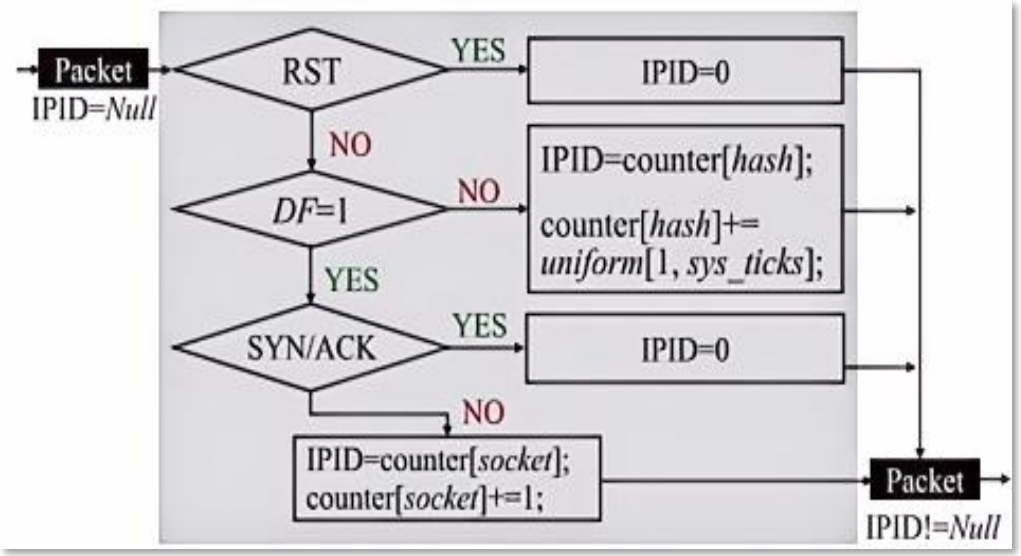
- **实验目的：**

复现基于IPID计数器的TCP连接注入攻击，了解网络协议栈常见安全威胁及漏洞，加深对网络安全问题的认识

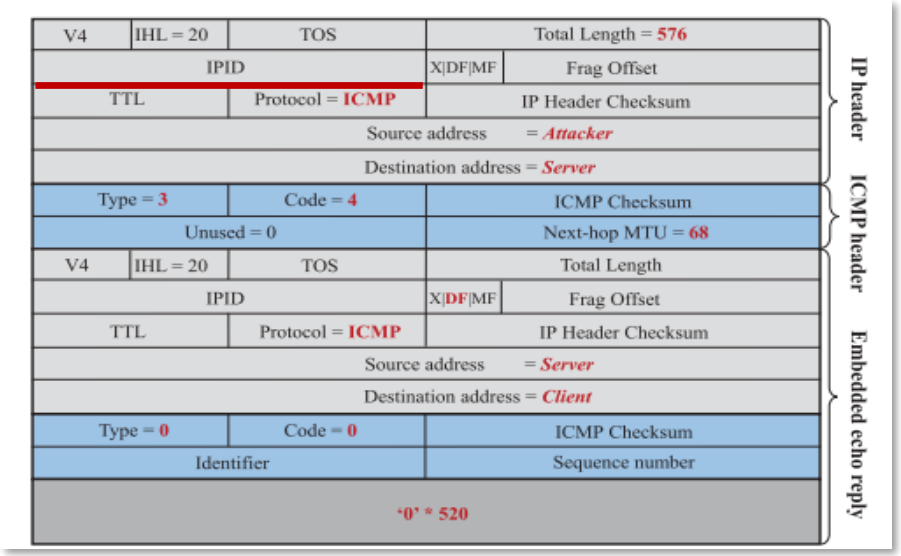




实验九：基于IPID的TCP侧信道漏洞 难度: ★★★

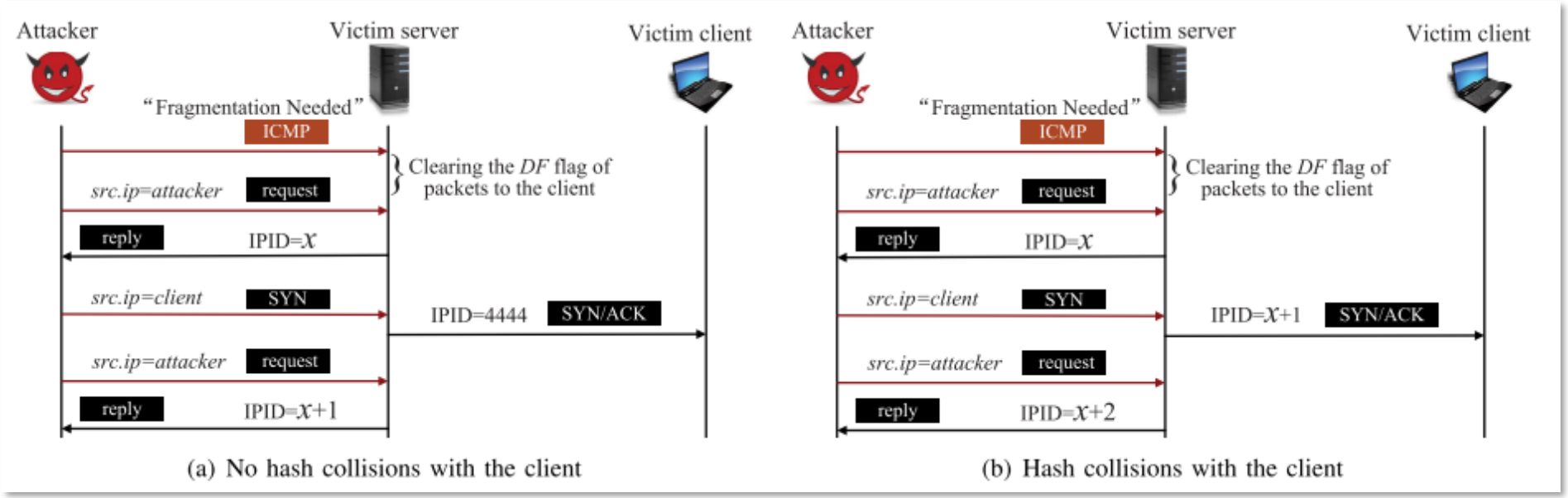


IPID实现流程图 Linux4.18



ICMP Error报文格式

基于IPID的TCP侧信道漏洞攻击原理



实验九：基于IPID的TCP侧信道漏洞　　难度: ★★★

- **实验要求：**

1. 根据实验环境设置通讯环境，配置正确的内核版本，建立客户端和服务器之间的受害TCP链接
2. 发起基于IPID的TCP侧信道攻击
3. 向TCP通讯信道当中注入恶意信息，可以在客户端注入或服务器端，**注入前链接不可中断**

- **实验参考：**

1. IPID侧信道漏洞原始文章：
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9556515
2. 参考源代码：
https://github.com/fuchuanpu/TCP-exploit

实验环境设置

如果你觉得这项工作还不错，
请留给我们一个 Star ⭐

实验十：互联网路由异常检测

难度：★★★

- **实验背景：**

边界网关协议（BGP）是全球范围内的互联网域间路由协议，支撑着互联网的互联互通。然而，由于 BGP 缺乏内建的安全机制，路由劫持和路由泄露等安全威胁时有发生，可能导致流量被劫持、监听或者形成流量黑洞，对网络安全带来重大隐患

- **实验背景：**

本实验旨在帮助读者理解路由劫持和路由泄露的攻击原理，并了解如何进行自动化路由异常检测。

- **实验参考：**

1. 参考文章Learning with Semantics: Towards a Semantics-Aware Routing Anomaly Detection System, Security 2024
2. 源代码：*https://github.com/yhchen-tsinghua/routing-anomaly-detection*

类别	名字	时间（±12h）	持续时间	异常路由通告数量
路由劫持（子前缀、路径）	$SP_{backcon\_5}$	2016-05-20 21:30:00	1h33m47s	296
	$SP_{backcon\_4}$	2016-04-16 07:00:00	11m28s	1032
	$SP_{backcon\_2}$	2016-02-20 08:30:00	3m10s	825
	$SP_{bitcanal\_1}$	2015-01-07 12:00:00	12m55s	286
	$SP_{petersburg}$	2015-01-07 09:00:00	28s	1000
	SP_{defcon}	2008-08-10 19:30:00	26s	72
路由劫持（子前缀、源）	SO_{iran}	2018-07-30 06:15:00	3h25m42s	587
	$SO_{bitcanal\_3}$	2018-06-29 13:00:00	47m34s	672
	$SO_{backcon\_3}$	2016-02-21 10:00:00	4s	1156
	$SO_{backcon\_1}$	2015-12-03 22:00:00	16m5s	695
	$SO_{bitcanal\_2}$	2015-01-23 12:00:00	5m11s	284
	SO_{h3s}	2014-11-14 23:00:00	2s	581
	$SO_{pakistan}$	2008-02-24 18:00:00	4h57m6s	156
路由劫持（前缀、源）	PO_{brazil}	2014-09-10 00:30:00	1h28m39s	127
	PO_{sprint}	2014-09-09 13:45:00	7s	198
路由泄露	RL_{jtl}	2021-11-21 06:30:00	1h14m47s	2419
	$RL_{stelkom}$	2021-11-17 23:30:00	3m18s	1888
	$RL_{itregion}$	2021-11-16 11:30:00	31m32s	2409

互联网域间路由系统攻击数据集

实验十一：实现本地DNS缓存中毒攻击 难度: ★★☆

- **实验背景：**

DNS是多种服务的基础，当DNS发生安全故障时，全部依赖DNS的服务都将被影响，最终危害大量使用者的安全

- **实验目的：**

通过实验，掌握DNS缓存中毒实施过程及原理，从而加深对网络安全问题的认识，并了解相应的安全防御手段

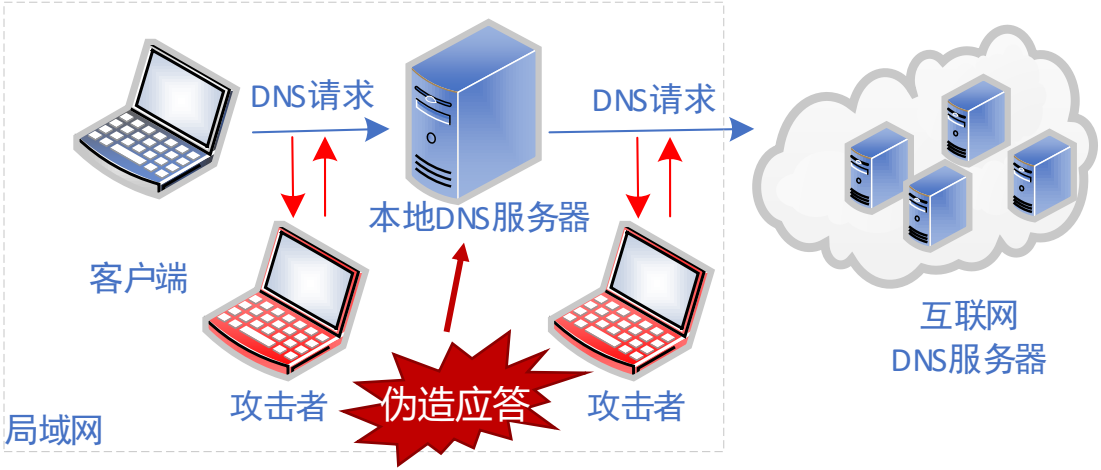
```
; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1367
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL:
 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.example.org.                IN      A

;; ANSWER SECTION:
www.example.org.        172800  IN      A       108.109.10.66

;; Query time: 19 msec
;; SERVER: 10.0.10.5#53(10.0.10.5)
;; WHEN: Fri Jan 22 12:27:52 EST 2021
```

攻击成功后，用户查询域名时，得到“攻击者”设定的伪造地址

实验十一：实现本地DNS缓存中毒攻击 难度: ★★☆

- **实验要求：**

1. 配置bind9本地DNS缓存服务器
2. 受害者访问某一域名
3. 攻击者编写并运行程序监听DNS请求，并在监听到请求后注入恶意DNS响应
4. 受害者得到错误DNS响应
5. 在缓存有效期内，受害者和其他客户端访问该域名均将得到错误响应



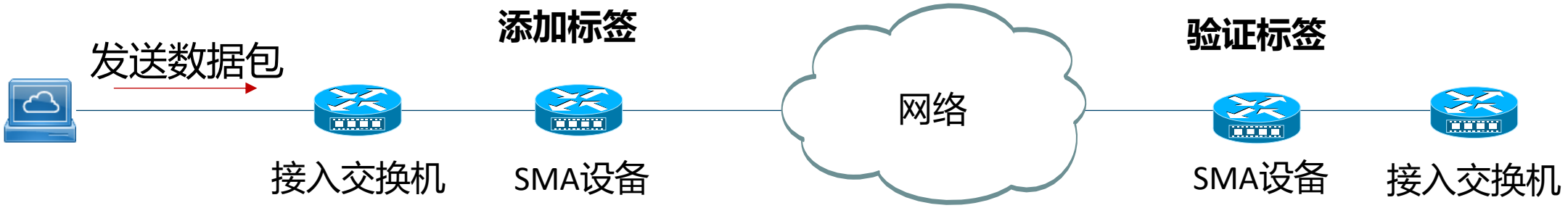
缓存中毒攻击流程

实验十二：域间源地址验证技术模拟 难度: ★★☆

- **实验背景：**

域间源地址验证技术SMA中，AS之间通过建立状态机，并在AS的边界路由器进行标签的添加、验证和移除，实现自治域间的源地址验证

发送方边界核心路由器在分组添加标签，以传递源地址前缀的真实性。目的域在边界路由器检查标签正确性，以验证所关联源地址的真实性，过滤伪造分组



实验十二：域间源地址验证技术模拟　　难度：★★☆

- **实验要求与参考资料：**

实验环境配置两台计算机，两台计算机分别充当发送方边界路由器以及目的域边界路由器，进行状态机建立、添加标签、验证标签三项功能的模拟

可将两台计算机作为一个虚拟机运行在同一物理计算机中

```
sava2@ubuntu:~/Desktop$ sudo python3 server.py
----------------------
Build share state!
['75b', 'cd15', '159a', '55a0', '1f12', '3bb5', '74', 'cbb1']
----------------------
Packet has correct tag!
Tag is:  ::aebf:45b3:3ab8:f7ac
----------------------
Packet has correct tag!
Tag is:  ::e3ec:cd99:c43d:a815
----------------------
Packet has wrong tag, drop!
----------------------
Packet has correct tag!
Tag is:  ::e5a3:ed60:dec5:6447
```

协商建立状态机

检验标签，同步状态机

实验十三：可视化分析流量交互图 难度: ★★☆

- **实验目的：**

本实验使用基于流量交互图的攻击流量识别程序，来检测公开数据集当中的攻击流量，进而分离攻击流量以及与之相关的正常流量，最后绘制流量交互图分析攻击流量

- **实验参考：**

项目源码：https://github.com/fuchuanpu/HyperVision

请思考：

1. 实验中介绍的流量交互图检测方法的具体原理是什么
2. 使用的是有监督训练还是无监督训练
3. 具体采用了多少数据包进行训练
4. 具体采用了多少流进行训练
5. 检测的准确度如何，如何解读这些指标

如果你觉得这项工作还不错，请留给我们一个 Star⭐

实验十四：网站指纹攻击实现

难度：★★☆

- **实验背景：**

匿名通信网络（如 Tor）虽然增强了用户的隐私保护，但也面临网站指纹攻击的威胁。这种攻击通过分析加密网络流量的特征来推断用户访问的网站，即使在匿名网络中也难以完全规避

- **实验目的：**

在开发并实现一种网站指纹攻击模型，通过监听和分析网络流量模式推测用户访问的网站。在Tor这样的匿名通信系统中，尽管流量经过多重加密和混淆，不同网站的流量特征（如数据包的方向、时间间隔等）仍然具有可区分性，攻击者可以利用这些特征进行识别。这种攻击对用户隐私构成严重威胁，因为它可能暴露用户的访问习惯及敏感信息，从而影响通信的安全性

- **实验参考：**

1. 参考文章：Robust and Reliable Early-Stage Website Fingerprinting Attacks via Spatial-Temporal Distribution Analysis, CCS 2024

2. 项目源码：https://github.com/Xinhao-Deng/Website-Fingerprinting-Library

实验十五：容错算法的模拟与验证 难度: ★★☆

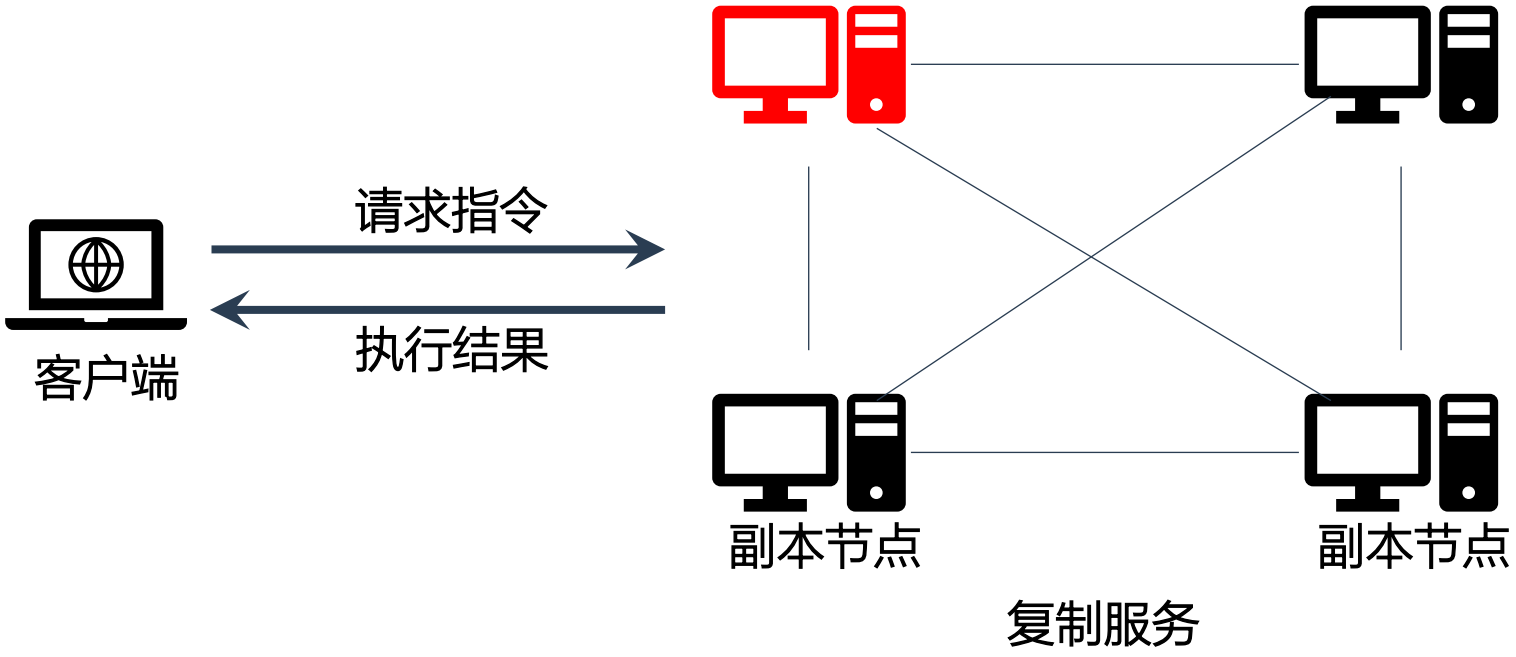
- **实验背景：**

容错算法可以用于构建一个分布式复制服务，能容忍少部分节点出现错误，持续稳定运行

- **实验目的：**

了解实际运行一个分布式系统需要配置的参数信息，加深对容错算法的认识

即使服务中的一个节点出现故障，服务依然能稳定运行



实验十五：容错算法的模拟与验证 难度: ★★☆

- **实验要求：**

用4个终端（进程）来构建一个四节点容错服务，另外一个终端充当客户端，发送服务请求

借助一个拜占庭容错共识开源库bft-smart完成

- **实验步骤：**

1. 开启由四个共识节点，彼此建立连接，组成映射表服务，并运行客户端使用该服务
2. 手动关闭其中一个节点，然后查看各共识节点的运行状态，并通过客户端判断系统是否能够正常运行
3. 将步骤2中关闭的节点重新开启，查看各共识节点的运行状态，判断是否可以恢复正常运行
4. 尝试手动关闭Leader节点对应的机器，再通过客户端发送请求，观察Leader节点更换过程

实验十六：实现本地Web攻击

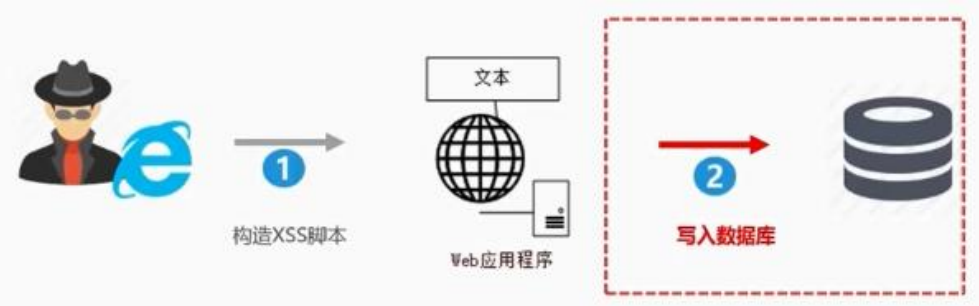
难度: ★★☆

- **实验目的：**

演示三种常见的 Web 攻击：XSS 、CSRF 、SQL 注入，并体会现代浏览器如何防御这些漏洞

- **实验要求：**

1. 在2018年后的浏览器上完成本实验，**需要关闭一系列防御机制**
2. 编写并部署存在漏洞的示例网站程序
3. 进行持久/非持久XSS，CSRF，以及SQL注入攻击



持久型XSS

非持久型XSS

实验十六：实现本地Web攻击

难度: ★★☆

XSS实验步骤举例

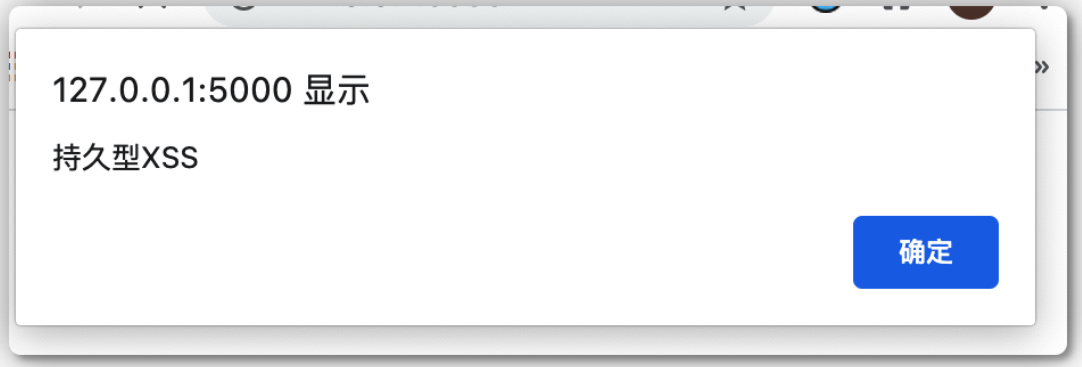
1. 安装Flask框架并运行Flask服务器

2. 访问受害服务器

```
$ flask run
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

3. 在搜索框内输入：

<script>alert(‘反射型XSS’)</script>



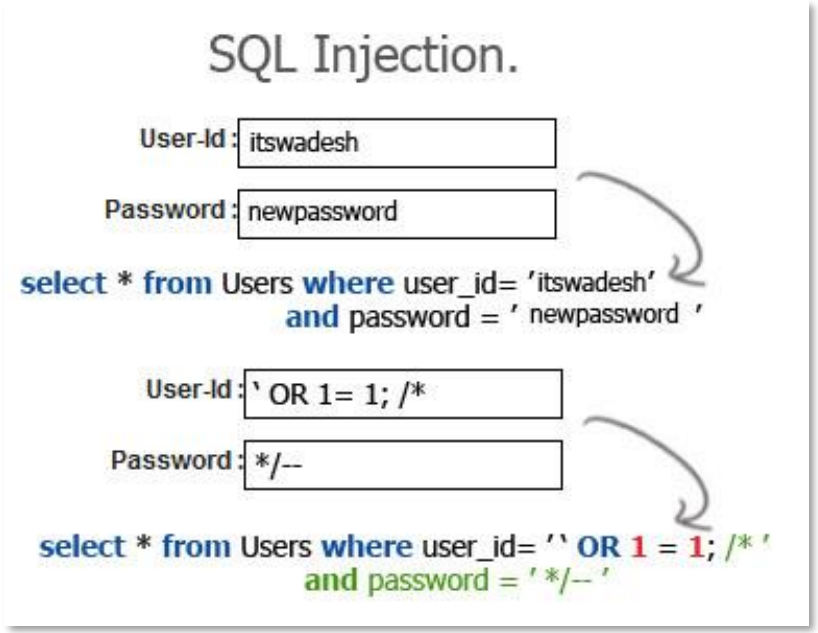
4. 在评论框内输入：

<script>alert('持久型XSS')</script>

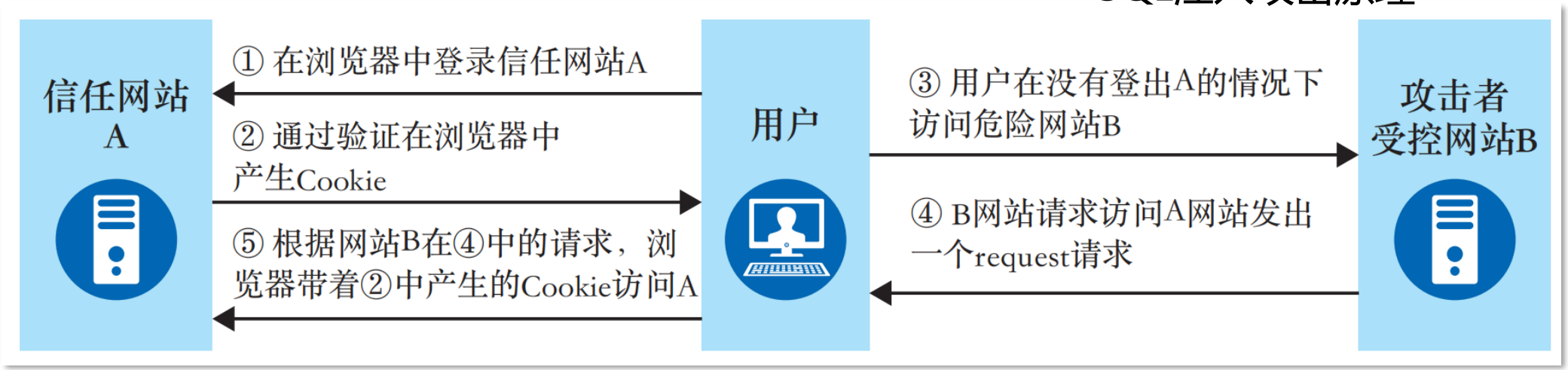
实验十六：实现本地Web攻击

难度: ★★☆

类似地可以完成CSRF和SQL注入攻击



SQL注入攻击原理



CSRF攻击原理

实验十七：后门攻击与防御的实现 难度：★★☆

- **实验背景：**

加深对人工智能安全问题的认识，了解后门攻击的实现逻辑以及防御细节，以及相应的防御

- **实验目的：**

采用成熟的神经网络后门攻击与防御方案，对常见神经网络结构植入后门，并部署后门防御机制

- **实验资料：**

1. BadNets 文章和源代码:

https://arxiv.org/abs/1708.06733

https://github.com/Kooscii/BadNets

2. Neural Cleanse 文章和源代码:

https://ieeexplore.ieee.org/abstract/document/8835365

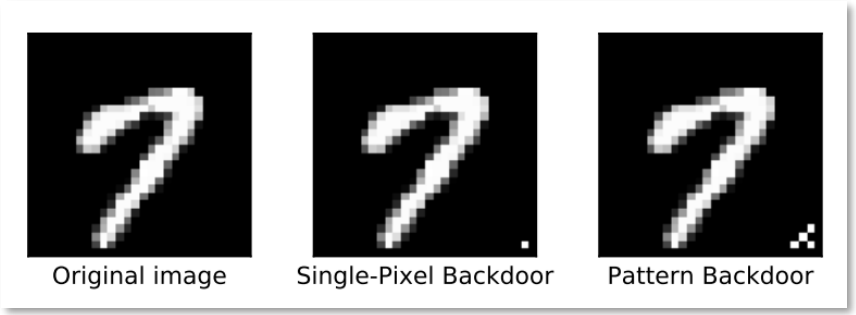
https://github.com/bolunwang/backdoor

神经网络暗门示意

实验十七：后门攻击与防御的实现 难度：★★☆

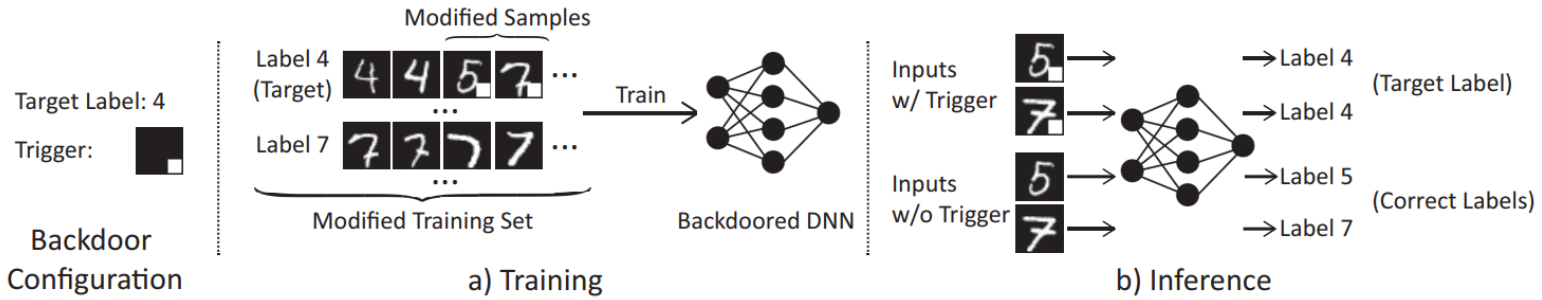
- **实验要求：**

1. 复现BadNets暗门方案：不改变原有深度学习模型结构，向训练数据增加特定模式噪音，修改训练数据标签，没有遇到特定模式噪音时正常工作，遇到特定模式噪音数据输出与预定规则相匹配的错误行为



处理后的图像

2. 复现Neural Cleanse暗门防御方案：指的是通过利用数据的独特属性或者精心设计的防御机制，来降低后门攻击的成功率



暗门防御方案

实验十八：流量大模型的流量检测能力实现 难度：★★★

- **实验背景：**

能够深入理解流量大模型的工作原理及其实际效果，加深对大模型和流量安全问题的认识，并掌握如何将大模型技术应用于网络安全领域

- **实验目的：**

开发一种基于流量大模型的网络威胁检测与分析系统，帮助大模型实现文本与流量模态的对齐

- **实验资料：**

1. ChatGLM2-6B
https://github.com/THUDM/ChatGLM2-6B
2. 大模型微调框架项目代码：
https://github.com/ZGC-LLM-Safety/TrafficLLM

- **实验步骤：**

1. 下载开源大模型的基座模型参数，微调框架项目代码，依赖库，文本指令数据集，流量数据集
2. 配置指令微调阶段的训练参数，并进行指令微调
3. 配置任务微调阶段的训练参数，并进行任务微调
4. 评估流量大模型在具体任务上的检测效果
5. 修改配置文件中指令微调模型和任务微调模型的存储路径，以备模型推理使用
6. 完成流量大模型的对话环境部署