自由软件与 GNU/Linux 安全

Shawn the R0ck

#whois

* citypw(Shawn C)
* GNU/Linux 安全工程师
* 自由软件狂热分子
* EFF/FSF 会员
* Hardenedlinux 社区 (http://hardenedlinux.org/) 发起人

HardenedLinux

cat /proc/agenda

* 自由软件运动
* 安全理念以及安全编码
* GNU/Linux 安全运维
* 企业安全现状
* 加固案例

cat /proc/agenda

* 自由软件运动
* **********
* **********
* **********
* **********

HardenedLinux

Free(and Open) Software Movement

The free software movement is a social and political movement with the goal of ensuring software user's 4 basic freedoms:

* The freedom to run the program, for any purpose(freedom 0)

* The freedom to study how the program works, and change it so it does your computing as you wish(freedom 1). Access to the source code is a precondition for this.

* The freedom to redistribute copies so you can help your neighbor(freedom 2)

* The freedom to distribute copies of your modified versions to others(freedom 3). By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this.

Who is RMS ?

他是 :

* 一名黑客
* 早期 GCC 和 Emacs 的作者
* 自由软件运动发起人
* GNU 项目的发起人
* 自由软件哲学传教士

他还：

* 曾经受邀来过 XCON

谁被 RMS 病毒“感染”？

John Carmack is:

- id software 创始人
- 3D 第一人称射击游戏创始人 (Wolfenstein 3D)
- 自由软件支持者，id software 不断的公开上一代游戏引擎源代码 .

谁被 RMS 病毒“感染” ?

Dirk Morris: "I originally thought of open source back in 1983 when I was 3 years of age. I made the terrible mistake of communicating my ideas to a beard man at MIT who later went on found the Free Software Foundation and take credit for all my ideas. I am still upset about this and we have not spoken since."

- CTO and founder of Untangle

- 2007 年，Untangle 以 GPLv2 自由软件许可证公开了 UTM 网关的源代码。

自由软件与安全有关？

人们一直依赖”故意“忽视自由软件社区和 *EFF* 对于闭源软件的不信任的呼声，直到 .....

谁是对的？

Edward Snowden 关于 NSA 的爆料至少证明了一点：
RMS 和 EFF 从来没有向公众撒谎 !!!

cat /proc/agenda

* **********

* 安全理念以及安全编码

* **********

* **********

* **********

什么是安全？

安全：目的 vs. 敌人

1, 策略 --> C(onfidentiality), I(ntegrity), A(vailability)

2, 威胁建模 --> 对敌人的假设

3, 机制 -> 软件 / 硬件

安全工程与其他工程最大的区别是：

* 敌人不会按牌理出牌，完全的不可预测

理念：信息安全防护基本原则

我们坚信：信息安全 …

* 不是安装一堆防火墙

* 不是一个产品或者服务 …
(by Bruce Scheiner)

* 不是一个产品，而是一个持续
不断的过程 …(by Bruce Scheiner)

* 安全审计不是 " 扫描一堆端口 "

•理念：信息安全防护基本原则（续）

我们坚信：信息安全 …

* 是虑 " 你能在不影响业务的情况下组织不被入侵吗？ "

* 是最薄弱的防线将成为你的短板

* 是你所在企业的资源（机器和人）的风险管理，需要专业技能，时间管理，实施成本，数据备份 / 恢复的一系列流程

* 是关乎过程，方法论，成本，策略和人

是考虑 " 有人能社工进入公司并且访问计算机，磁盘和磁带 …"

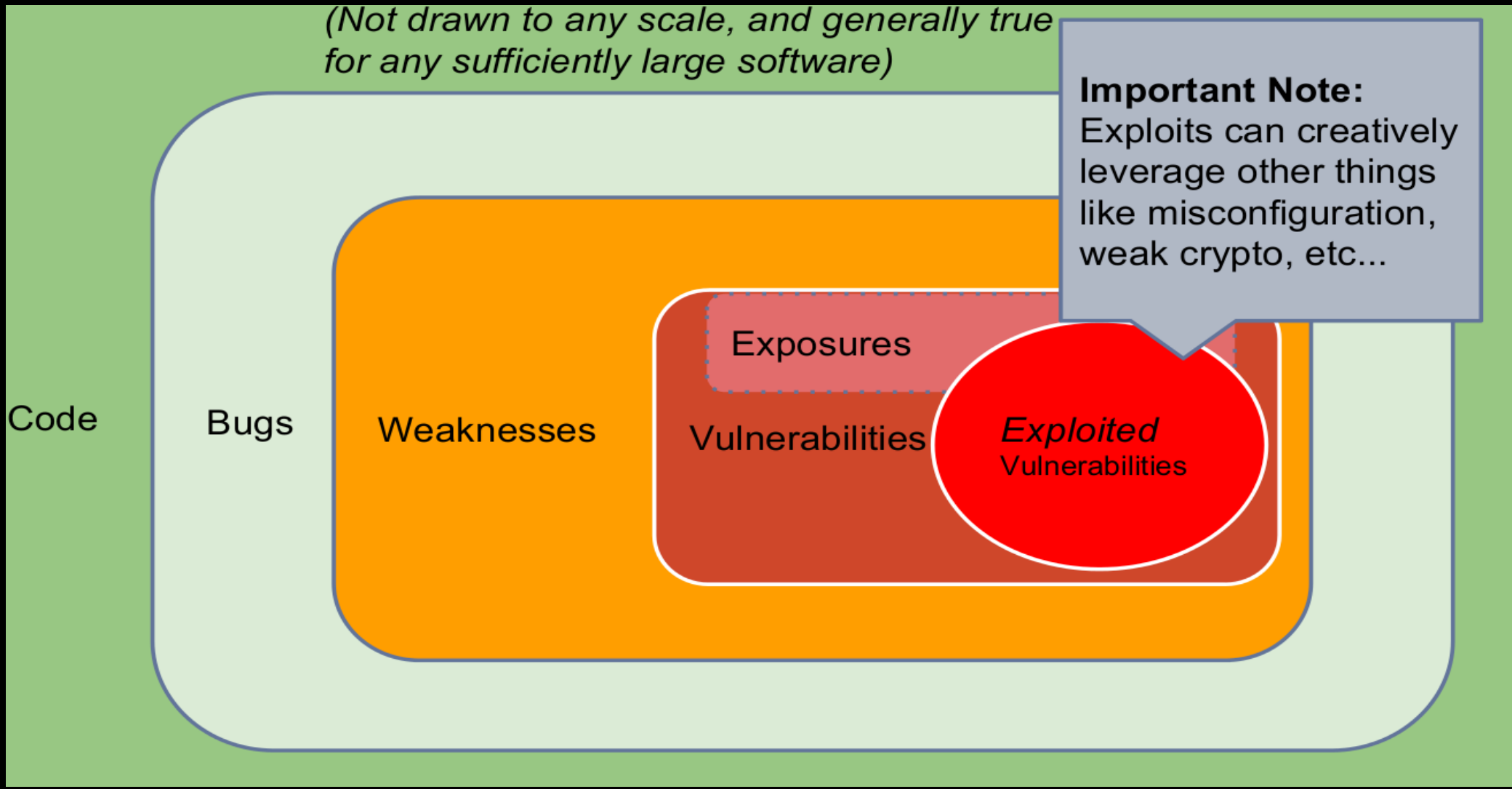
* 是 24 * 7 * 365… 持续不断 .. 并且永不停止

安全编码

* 编码安全概述
* C/C++ 代码的安全性
* 编译器加固选项
* 代码审计

编码安全概述

* 指由代码质量问题引发的安全问题



漏洞分类

* 技术性漏洞
    * 利用硬件，OS，库等特性进行利用

* 逻辑性漏洞

    利用处理业务部分的代码逻辑错误进行利用

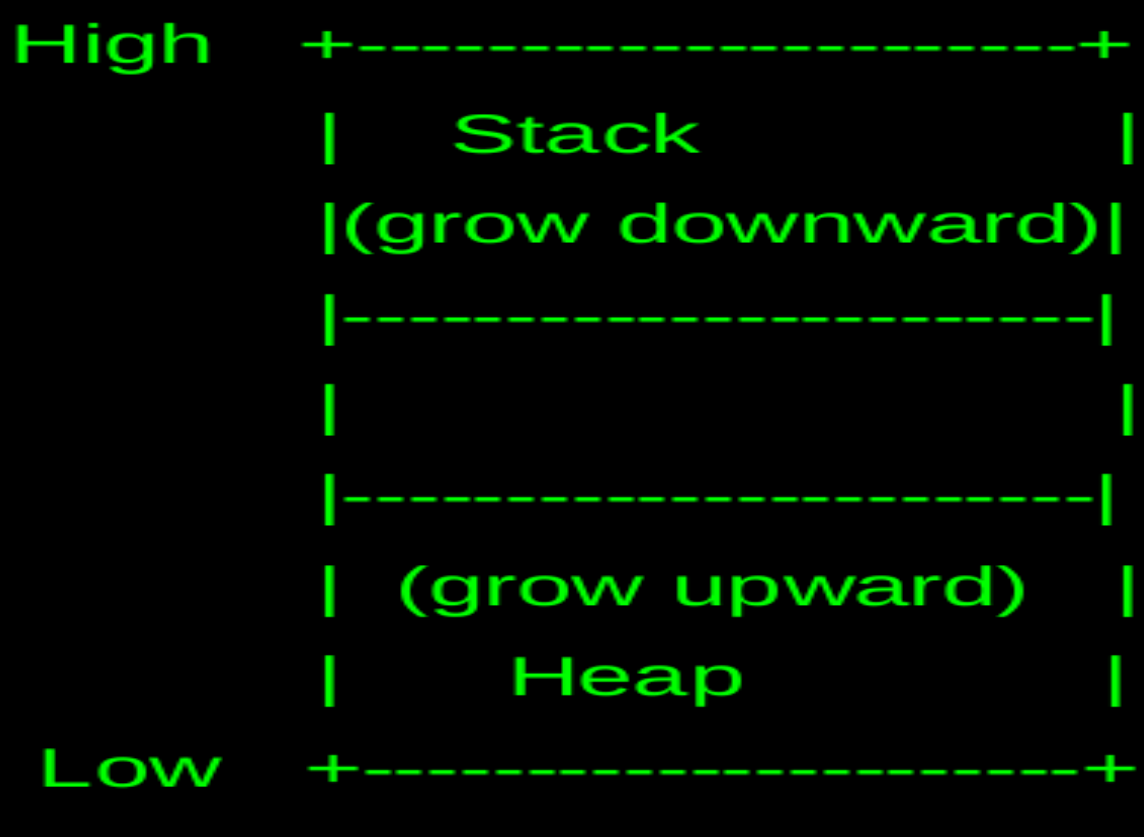
C/C++ 代码的安全性

常见漏洞类型：

* 缓冲区溢出
* 整数溢出
* 竞争情况
* 字符串格式化

栈结构

* Stack/Heap layout
* Stack grows down (x86, MIPS)
* ESP points to the current top of the stack
* EBP points to the current function frame



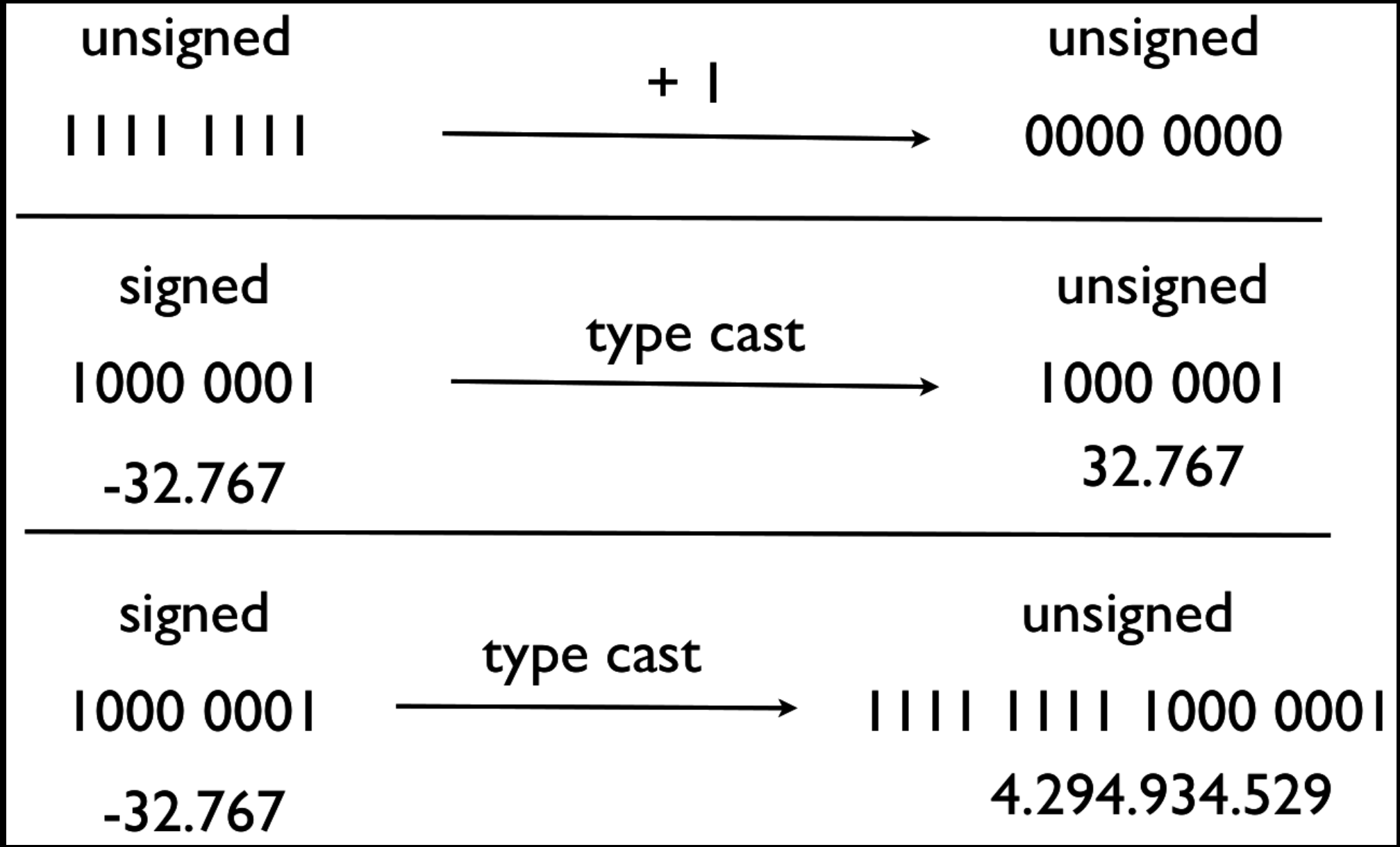
栈缓冲区溢出示例 1

```
#include <stdio.h>
#include <string.h>

void fuck_me(const char *input)
{
  int x; /* integer variable need 4-bytes in memory */
  char buf[10];
  strcpy(buf, input);
  printf("%s \n", buf);
}


int main(int argc, char *argv[])
{
  printf("The address of func fuck_me(): %p\n", fuck_me);
  fuck_me(argv[1]);//    -----> It's a vunerable func without bound checks!
  return 0;
}
```

实例

```
.....................
.....................
/* that's our secret phrase */
char origPassword[12] = "Geheim\0";
char userPassword[12];
/* read user input */
gets(userPassword);
if(strncmp(origPassword, userPassword, 12) != 0)
{

 printf("Password doesn't match!\n");exit(-1);
}
```

整数溢出


unsigned
1111 1111
+ 1
unsigned
0000 0000
signed
1000 0001
-32.767
type cast
unsigned
1000 0001
32.767
signed
1000 0001
-32.767
type cast
unsigned
1111 1111 1000 0001
4.294.934.529

整数溢出示例

```
#include <stdio.h>
#include <string.h>
int main(int argc, char *argv[]){
        unsigned short s;
        int i;
        char buf[80];

        if(argc < 3){
                return -1;
        }

        i = atoi(argv[1]);
        s = i;

        if(s >= 80){            /* [w1] */
                printf("Oh no you don't!\n");
                return -1;
        }

        printf("s = %d\n", s);

        memcpy(buf, argv[2], i);
        buf[i] = '\0';
        printf("%s\n", buf);

        return 0;
}
```

整数溢出示例

绕过边界检查：

```
linux-ionf:~ # ./a.out 5 test
s = 5
test
linux-ionf:~ # ./a.out 200 test
Oh no you don't!
linux-ionf:~ # ./a.out 65536 test
s = 0
Segmentation fault
linux-ionf:~ #
```

竞争情况

* TOCTOU(time-of-check-to-time-of-use)

原因：

- 非原子操作
- 没有正确使用琐

重灾区：

- 文件系统
- 多线程程序
- 分布式数据库

避免文件系统中的竞争

* 尽量少使用链接文件名，或者使用O_NOFOLLOW 禁用链接
* fork(2)+ 降低权限
* chdir(2) 前必须检查上级目录

```
chdir("/tmp/a");
chdir("b");
chdir("c");
chdir("..");
rmdir("c");
unlink("*");
```

Race Window

This code relies on the existence of a directory with path /tmp/a/b/c

An exploit is
mv /tmp/a/b/c /tmp/c

字符串格式化

* 2000 年左右公布
* printf 系列函数
* 读取： %x, %p
* 写入： %n

错误使用方法：snprintf(buf, sizeof(buf), user_data)

正确使用方法：snprintf(buf, sizeof(buf), “%s”, user_data)

编译器加固选项

作为 GNU 的三大核心基础组建之一的编译器集合 GCC ，提供了一系列加固选项（ MITIGATION ），目的是即使开发人员在开发过程中的各种原因造成了漏洞，但也让攻击者的漏洞利用难度加大。

Stack Canary

Stack Canary ：在压栈过程中加入一个值，在出栈时进行对比，如果被修改说明有漏洞利用的行为。

```
int fuc_me(int x, int y) /* x? WTH */
{
    Int v; /* v? */
    char buf[256];
    int h;
.....................
}
```

原始内存布局：

[High addr...[y].[x].[ret].[frame pointer].[v].[buf].[h]...Low addr]

简单canary内存布局：

[High addr...[y].[x].[ret].[frame pointer].[canary].[v].[buf].[h]...Low addr]

复杂canary内存布局：

[High addr...[y].[x].[ret].[frame pointer].[carnary].[buf].[h].[v]...Low addr]

Stack Canary

GCC 选项：

-fstack-protector，编译时给部分的函数加入 canary

-fstack-protector-all ，编译时给所有的函数加入 canary

-fstack-protector-strong ，更智能，编译时判断是否有可能会有 stack 数据的操作，如果有就自动加入 canary

Position-Independent-Executable

在 GNU/Linux 平台上，加载器在加载 elf 文件后会映射 elf header 到内存，非 PIE 的文件内核会使用一个等价于 MMAP_FIXED 的内部 flag ，PIE 的情况会强制所有进程的代码段的基地址随机化。

PIE 的效果

GCC选项：

-pie，代码段随机化

条件： ASLR支持

```
void* getEIP () {
    return __builtin_return_address(0);
};

int main(int argc, char** argv){
    printf("retaddr: %p\n",getEIP());
    return 0;
}
```

---

retaddr: 0xb78d950a

retaddr: 0xb77a450a

retaddr: 0xb776350a

Turn off ASLR

```
root@sl13:/tmp# echo 0 > /proc/sys/kernel/randomize_va_space
root@sl13:/tmp# ./a.out
retaddr: 0xb7fff50a
root@sl13:/tmp# ./a.out
retaddr: 0xb7fff50a
root@sl13:/tmp# ./a.out
retaddr: 0xb7fff50a
root@sl13:/tmp# echo 2 > /proc/sys/kernel/randomize_va_space
root@sl13:/tmp# ./a.out
retaddr: 0xb77d450a
root@sl13:/tmp# ./a.out
retaddr: 0xb780c50a
root@sl13:/tmp# ./a.out
retaddr: 0xb786850a
root@sl13:/tmp#
```

Turn on ASLR

其他加固选项

GCC 选项：

-FORTIFY_SOURCE，静态以及运行时检测缓冲区溢出

-z nostack，不可执行的栈

-z relro, -z relro -z now, read-only relocation ， GOT(全局偏移表) 变成只读

编译器加固

编译器加固选项能解决漏洞利用的问题吗？

能在多大程度上增加攻击者的成本？
2011---$5,000?

最新情况？ 2015 ？

“猎杀”漏洞

主要方法：

* 代码审计

  需要源代码

* 逆向工程

  可以在没有源代码的情况下完成

  需要二进制程序

  耗时多和需要高水平人员

* Fuzzing（模糊测试）

  组合利用大量工具

代码审计

* 工作乏味而且耗费大量时间
* 难以估算工作时间
* 需要有特定语言的知识背景
* 重点观察有输入的地方

代码审计原则

* 不要假设，假设会把你引向错误的方向
* 抽象层出错概率高（多人用，但思路不同）
* 每次阅读大约 200-400 行代码
* 最高效的审计每小时大概阅读 500 行代码
* 每 90 分钟必须休息一段时间

排查隐患函数

1990s 后期，代码审计中经常出现用于寻找滥用 strcpy() 的潜在缓冲区漏洞：

```
# grep strcpy *.c
```

2000s ，业界又成这样了，寻找字符串格式化漏洞：

```
# grep –E –e ‘printf\s*\([^”]+[,\)]’ *.c
```

异常处理完善？

简单的 read(2) 的比较完善例子：

返回值小于 LEN ，重新计算没有读取的 buf ，然后继续 read(2)

如果由于信号阻断，继续尝试

loop terminates

```
void read_all(int fd, void *buf, size_t len)
{
        ssize_t ret;
        while (len != 0 && (ret = read(fd, buf, len)) != 0) {
                if (ret == -1) {
                        if (errno == EINTR) /* EINTR = signal break, EAGAIN = no data currently available */
                                continue;
                        perror("read");
                        break;
                }
                len -= ret;
                buf += ret;
        }
}
```

带符号的整数

if(x > y)

如果一个是 unsigned ，另外一个也会是

if(x > 16)

16 是 signed ， >MAX_INT 溢出

if(x > 16U)

16 是 unsigned

字符串格式化

printf(input);

不安全

printf(“%s”, input);

安全

sprintf(tmp, “%s”, input);
printf(tmp);

不安全

Off by One

```
char msg[5]
for (i = 0; i <= 5; i++)
//use msg;
```

应该 i<5

类似情况：

- 增量过多
- 错误的边界检查
- sizeof != strlen

i++ vs ++i

```
x = i++ - 5;
```

??

```
x = ++i – 5;
```

??

cat /proc/agenda

* **********

* **********

* GNU/Linux 安全运维

* **********

* **********

安全部署 --- 防范已知威胁

安全部署（基线）不应该是安全检查列表的“另一面”描述，而应该是对所有有可能影响安全因素的详细配置清单：

- * 服务器 /PC 安全修复更新（防 NDAY 漏洞利用）
- * 密码策略
- * 业务分离
- * 防火墙规则（5 元组或者 IDS/IPS?)
- * 审计
- * …

根据业务场景的安全部署 !!!

安全补丁

Debian 检查需要安全修复包：

```
sudo apt-get upgrade -s | grep -i security
```

OpenSuSE 发行版检查需要安全修复的包：

```
sudo zypper lp | awk '{ if ($7=="security"){ if ($11=="update") {print $13} else{ print $11 }}}' | sed 's/:$//' | grep -v "^$" | sort | uniq
```

RHEL/CentOS(6.4) 检查需要安全修复的包和明细：

```
sudo yum list-security | grep RHSA
sudo yum info-security RHSA-NUMBER
```

安全补丁

客户关心的问题：

* 安全威胁程度
* 是否影响业务
* Workaround?

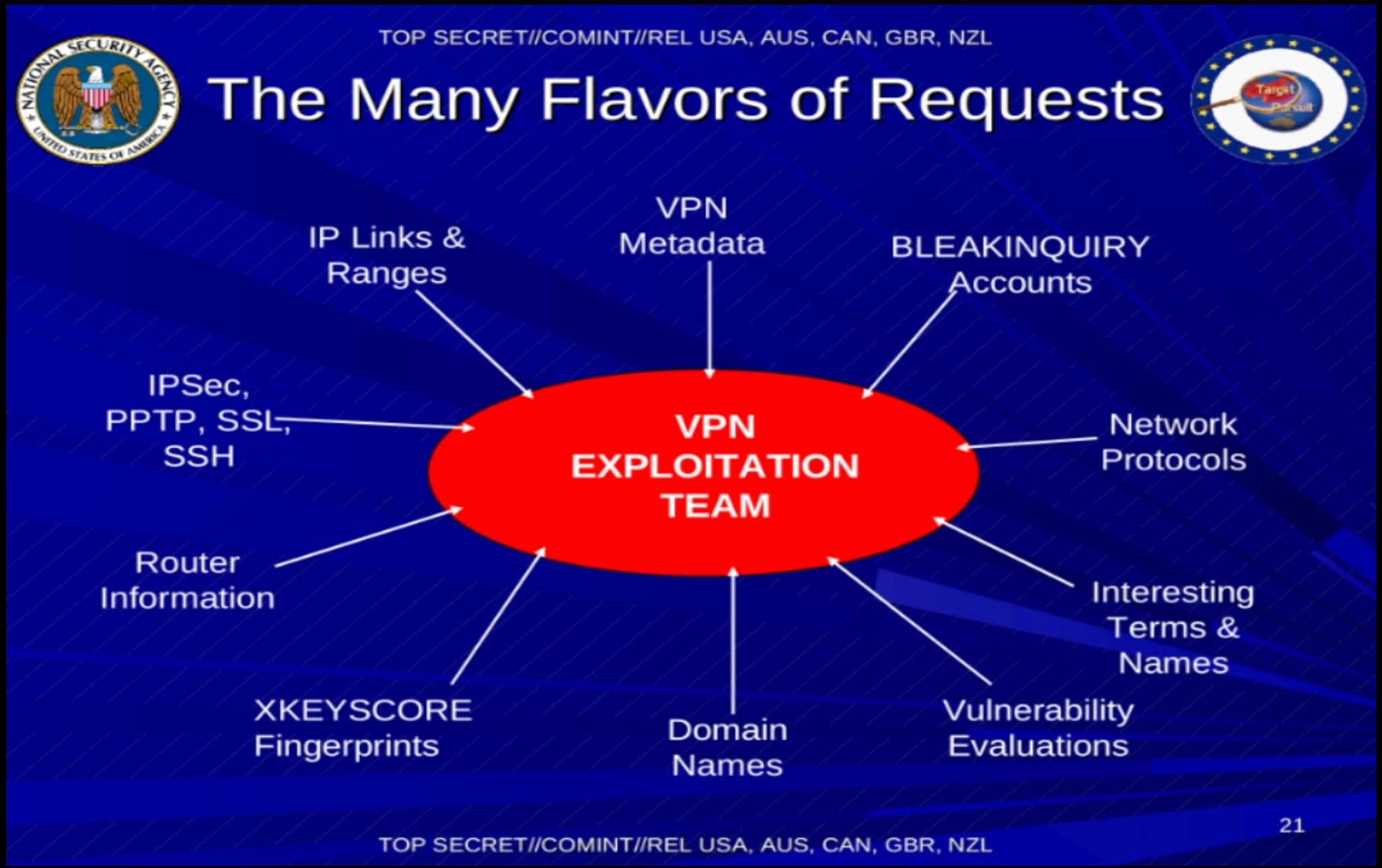
安全部署检查 --- 安全审计

* SSH 安全配置
* SSL/TLS 安全链路
* UNIX 通配符陷阱
* 异常进程 / 流量
* GNU/Linux 链接器威胁
* ETC

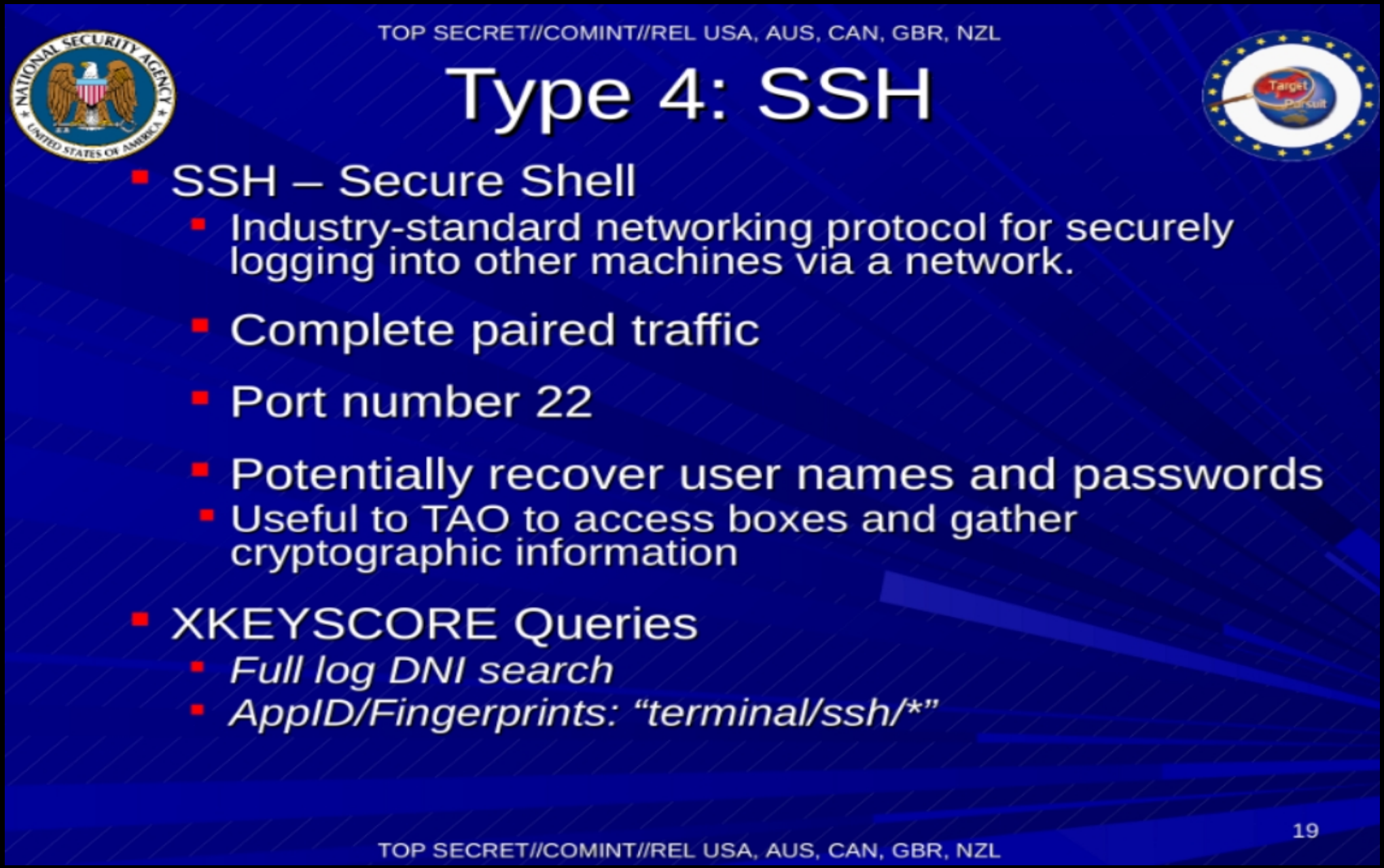
SSH 传统基线

known_hosts 保存相关服务器的签名，所以必须把主机名 hash	HashKnownHosts yes
SSH 协议 v1 不安全	Protocol 2
如果没用 X11 转发的情况	X11Forwarding no
关闭 rhosts	IgnoreRhosts yes
关闭允许空密码登录	PermitEmptyPasswords no
最多登录尝试次数	MaxAuthTries 5
禁止 root 登录	PermitRootLogin no

后棱镜时代的 SSH 尴尬



SSH 基线整改？


TOP SECRET//COMINT//REL USA, AUS, CAN, GBR, NZL
Type 4: SSH
SSH – Secure Shell
Industry-standard networking protocol for securely logging into other machines via a network.
Complete paired traffic
Port number 22
Potentially recover user names and passwords
Useful to TAO to access boxes and gather cryptographic information
XKEYSCORE Queries
Full log DNI search
AppID/Fingerprints: “terminal/ssh/*”
TOP SECRET//COMINT//REL USA, AUS, CAN, GBR, NZL
19

OpenSSH 新基线选型

KX 选择标准：

- ECDH 曲线的 NIST 安全标准被污染
- 排除 DH 密钥强度 (<=1024)
- 排除 SHA1

认证选择标准：

- DSA 强度过低排除
- ECDSA 被 NIST 安全标准污染
- 只能信任 >=2048 的 RSA

OpenSSH 新基线选型

对称算法选型标准：

- 算法本身的安全性，排除 RC4 和 DES
- 密钥强度至少 128-bit
- 块加密情况下至少 128-bit 的 block
- Cipher mode 推荐 AE（认证模式）+CTR

MAC 选型标准：

- MD5 和 SHA1 被排除
- 必须先加密再 MAC ，不带 -etm 被排除
- 密钥强度至少 128-bit

SSL/TLS 检查

检测 SSLv2	gnutls-cli -d 5 -p 443 --priority "NORMAL:-VERS-TLS1.2:-VERS-TLS1.1:-VERS-TLS1.0:-VERS-SSL3.0" www.google.com
检测 SSLv3(POODLE)	gnutls-cli -d 5 -p 443 --priority "NORMAL:-VERS-TLS1.2:-VERS-TLS1.1:-VERS-TLS1.0" www.google.com | grep -i version
检测 TLSv1.1	openssl s_client -tls1_1 -connect www.google.com:443 | grep -i protocol
检测 TLSv1.2	openssl s_client -tls1_2 -connect www.google.com:443 | grep -i protocol
安全重协商（显示“ Safe renegotiation succeeded” ）	gnutls-cli -d 5 -p 443 www.google.com
公钥长度检测	gnutls-cli -p 443 www.google.com | grep -i key
检测是否存在较弱的 ciphersuites	openssl s_client -cipher NULL,EXPORT,LOW,DES -connect www.google.com:443

SSL/TLS 检查

PFS(Perfect Forward Secrecy)	openssl s_client -cipher EDH,EECDH -connect www.google.com:443
抗 CRIME 攻击 (显示 “ -Compression: NULL “)	gnutls-cli www.google.com -p 443 | grep -i compress
检测 OpenSSL 的 heartbeat	openssl s_client -tlsextdebug -connect www.google.com:443 | grep -i heart
POODLE(显示 "alert inappropriate fallback")	openssl s_client -ssl3 -fallback_scsv -connect www.google.com:443 | grep -i "alert ina"

SSL/TLS 部署：社区最佳实践

Note: All the ciphersuites & SSL/TLS protocol are tested and verified that they works with Firefox(version 33—37), Internet Explorer(8 and later version).

SSL/TLS protocols:

SSLv2	Insecure protocol. Recommend: disable it.
SSLv3	Expose to POODLE attack. Recommend: disable it.
TLSv1	Weak ciphers(DES, RC4) are potential risks. CRIME attack need to be mitigated.
TLSv1.1	Need to secure config, properly.
TLSv1.2	Need to secure config, properly.

“Secure” configuration for SSL/TLS from the best practice, which still work in the post-prism era during in 2014/2015. The example of configurations are massive deployed by free & open source community. Edit /etc/httpd/conf.d/ssl.conf:

```
SSLProtocol ALL -SSLv2 -SSLv3

SSLHonorCipherOrder On
SSLCipherSuite
ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+AES128:DH+AES:ECDH+3DES:DH+3DES:RSA+AESGCM:RSA+AES:RSA+3DES:!aNULL:!MD5
```

In OpenSSL 1.0.1e code base, 21 ciphersuites are available.

SSL/TLS 部署：社区最佳实践

Preference	Ciphersuite	Protocols	PFS（非常重要！）	Browser support
1	ECDHE-RSA-AES256-GCM-SHA384	TLSv1.2	ECDH,P-256,256bits	Firefox(33.1 or later). IE 11. Chrome(32 or later)
2	ECDHE-RSA-AES256-SHA384	TLSv1.2	ECDH,P-256,256bits	...
3	ECDHE-RSA-AES256-SHA	TLSv1,TLSv1.1,TLSv1.2	ECDH,P-256,256bits	...
4	DHE-RSA-AES256-GCM-SHA384	TLSv1.2	DH,2048bits	...
5	DHE-RSA-AES256-SHA256	TLSv1.2	DH,2048bits	...
6	DHE-RSA-AES256-SHA	TLSv1,TLSv1.1,TLSv1.2	DH,2048bits	...
7	AES256-GCM-SHA384	TLSv1.2	N/A	Firefox(33.1 or later). IE 11. Chrome(32 or later)

SSL/TLS 部署：社区最佳实践

Preference	Ciphersuite	Protocols	PFS（非常重要！）	Browser support
...	...	...	...	...
16	AES128-GCM-SHA256	TLSv1.2	N/A	Firefox(27 or later). IE 11. Chrome(30 or later)
17	AES128-SHA256	TLSv1.2	N/A	Firefox(27 or later). IE 11. Chrome(30 or later)
18	AES128-SHA	TLSv1,TLSv1.1,TLSv1.2	N/A	Firefox(27 or later). IE(8 or later) . Chrome(30 or later)
19	ECDHE-RSA-DES-CBC3-SHA	TLSv1,TLSv1.1,TLSv1.2	ECDH,P-256,256bits	Firefox(27 or later). IE 11. Chrome(30 or later)
20	EDH-RSA-DES-CBC3-SHA	TLSv1,TLSv1.1,TLSv1.2	DH,2048bits	Firefox(27 or later). IE 8. Chrome(30 or later)
21	DES-CBC3-SHA	TLSv1,TLSv1.1,TLSv1.2	N/A	Firefox(27 or later). IE(8 or later) . Chrome(30 or later)

链接器陷阱

Vulnerability title: SetUID/SetGID Programs Allow Privilege Escalation Via Insecure RPATH In IBM DB2

CVE:	CVE-2014-0907
Vendor:	IBM
Product:	DB2
Affected versions:	V9.1, V9.5, V9.7, V10.1 and V10.5
Fixed versions:	V9.7 FP9a, V10.1 FP3a, V10.1 FP4 and V10.5 FP3a
Reported by:	Tim Brown

Details:

It has been identified that binaries that are executed with elevated privileges (SetGID and SetUID programs) in IBM's DB2 for AIX and Linux have been compiled in manner that means they searched for libraries in insecure locations.

```
SUIDFILE='/db2/db2gpe/sqllib/acs/acscim' SUIDFILELS='-rwsr-x--- 1 root dbgpeadm 43848124 04
Oct 2012 /db2/db2gpe/sqllib/acs/acscim' RPATH='../../common/unx/supincl/pegcim251/lib'
RPATHRELATIVE=yes RPATHLS=N/A RAPTHEXISTS=N/A ISBAD=yes
SUIDFILE='/db2/db2gpe/sqllib/acs/acsnnas' SUIDFILELS='-rwsr-x--- 1 root dbgpeadm 43399984 04
Oct 2012 /db2/db2gpe/sqllib/acs/acsnnas' RPATH='../../common/unx/supincl/ontapsdk16/lib/aix'
RPATHRELATIVE=yes RPATHLS=N/A RAPTHEXISTS=N/A ISBAD=yes
SUIDFILE='/db2/db2gpe/sqllib/acs/acsnsan' SUIDFILELS='-rwsr-x--- 1 root dbgpeadm 46767866 04
Oct 2012 /db2/db2gpe/sqllib/acs/acsnsan' RPATH='../../common/unx/supincl/ontapsdk16/lib/aix'
RPATHRELATIVE=yes RPATHLS=N/A RAPTHEXISTS=N/A ISBAD=yes
SUIDFILE='/db2/db2gpe/sqllib/adm/db2iclean' SUIDFILELS='-r-sr-x--- 1 root dbgpeadm 23157 25
May 2013 /db2/db2gpe/sqllib/adm/db2iclean' RPATH='.' RPATHRELATIVE=yes RPATHLS=N/A
RAPTHEXISTS=N/A ISBAD=yes
SUIDFILE='/db2/db2gpp/sqllib/acs/acscim' SUIDFILELS='-rwsr-x--- 1 root dbgppadm 43848124 04
Oct 2012 /db2/db2gpp/sqllib/acs/acscim' RPATH='../../common/unx/supincl/pegcim251/lib'
RPATHRELATIVE=yes RPATHLS=N/A RAPTHEXISTS=N/A ISBAD=yes
SUIDFILE='/db2/db2gpp/sqllib/acs/acsnnas' SUIDFILELS='-rwsr-x--- 1 root dbgppadm 43399984 04
Oct 2012 /db2/db2gpp/sqllib/acs/acsnnas' RPATH='../../common/unx/supincl/ontapsdk16/lib/aix'
```

审计方法：

```
for i in $SETUID_PROGS; do
        objdump -x $i | grep -i path > /dev/null
        if [ $? -eq 0 ]; then
                echo -e "*- \e[91mPlease check this binary: $i"
        fi
done
```

UNIX 通配符陷阱

```
shawn@shawn-fortress /tmp/exp $ ls
admin.php  --checkpoint=1  --checkpoint-action=exec=sh shell.sh  db.php  shell.sh  undo.php
shawn@shawn-fortress /tmp/exp $ cat shell.sh
/usr/bin/id
shawn@shawn-fortress /tmp/exp $ tar cf test.tar *
uid=1000(shawn) gid=1000(shawn) groups=1000(shawn),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),108(lpadmin),110(samba
share),1002(promiscuous)
shawn@shawn-fortress /tmp/exp $
```

审计方法：

find / -path /proc -prune -name "-*"

UNIX 文件审计

Audit method	Description
find / -path /proc -prune -name "-*"	WildCards is a powerful feature in UNIX-like platform, but it can be exploited by attackers: http://www.defensecode.com/public/DefenseCode_Unix_WildCards_Gone_Wild.txt
find / -path /proc -prune -o -perm -2 ! -type l -ls	World-writable file audit
find /var/log -perm -o=r ! -type l	World-readable file audit, correct permission: chmod 640 /var/log/
find / -path /proc -prune -o -nouser -o -nogroup	Check if files without owners
egrep -v '.*:*|:\!' /etc/shadow | awk -F: '{print $1}'	List avaiable users

UNIX 文件审计 2

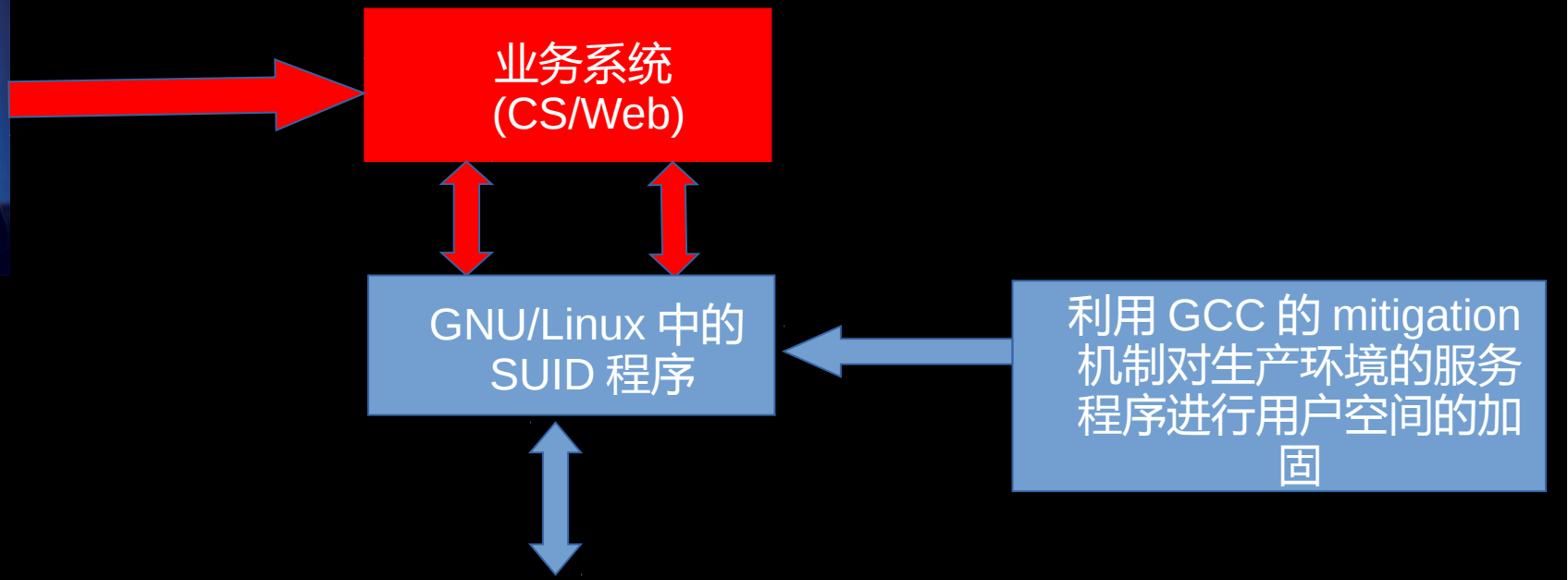
Audit method	Description
find / -path /proc -prune -o -user account -ls	Check which files belong to whom. Then delete the user correctly: userdel -r account
grep -v ':x:' /etc/passwd	List which users are unavaiable
cat /etc/shadow | cut -d: -f 1,2 | grep '!'	List expired passwords
ls -l /boot	The correct permission should be 644 at least. 600 would be better.
find / -xdev -user root \(-perm -4000 -o -perm -2000 \)	Files with suid or sgid flags
objdump -x $i | grep -i path	Check if some stupid mistakes has been made.

Linux 内核安全配置

net.ipv4.tcp_syncookies	SYN 洪水攻击防护，建议设置为 1（开启）
SYNPROXY(Debian 8/CentOS 7 支持)	N/A
net.ipv4.tcp_max_orphans	默认 65536 ，值调大能防止简单 DOS 攻击，每个 ORPHAN 消耗大约 64k 的内存，65535 相当于 4GB 内存
kernel.randomize_va_space	地址随机化， 0: 关闭 1: 针对 stack ， VDSO ，数据区和 mmap()'ed 的随机化 2: 增加 heap 的随机化
kernel.kptr_restrict	内核符号限制访问，类似 CVE-2014-0196 的 exploit 对于这一项就很难做到“通杀”
vm.mmap_min_addr	内存映射最小地址，防止 NULL DEREF 攻击，默认 4096 ，建议 65536

深度加固 --- 遏制未知威胁

为什么需要系统层加固？



MAC/RBAC

但仅仅靠用户空间加固并不能解决所有问题 ……

SELinux/Apparmor/SMACK/TOMOYO ？ --- 深入内核层的安全加固机制，通过结合应用系统行为的安全策略配置，遏制漏洞利用！

以 Apparmor 的白名单配置为例 :

```
/home/jim/a.out {
  #include <abstractions/base>
  /home/jim/a.out mr,
  /home/jim/hello r,
  /home/jim/world w,
   network stream,
}
```

深度加固 --- “ 终极” 加固

但是，对于真正面临复杂安全环境，确实需要高级安全防护能力的机构，最极端的加固防御是：PaX/Grsecurity

对于注重完美的“老派（ old school ）” 黑客社区而言，没有 PaX/Grsecurity 的定制方案是不完美的，对于商业客户而言，是否定制是一种选择。

grsecurity

内核加固方案对比

功能	SELinux	Apparmor	PaX/Grsecurity
强制访问控制 (MAC)	支持	支持	支持
审计	支持	支持	支持
阻止用户态运行时代码生成	依赖规则	不支持	支持
自动阻止 ptrace 程序调试	依赖规则	不支持	支持
配合其他 LSM 一起使用	不支持	不支持	支持
降低内核本身的信息泄漏风险机制	不支持	不支持	支持
防御内核任意代码执行	不支持	不支持	支持
防御内核直接访问用户态内存空间	不支持	不支持	支持

内核加固方案对比

功能	SELinux	Apparmor	PaX/Grsecurity
防御内核引用计数器溢出	不支持	不支持	支持
内核重要函数指针以及重要数据结构的只读	不支持	不支持	支持
在内核早期启动和运行中增加 entropy	不支持	不支持	支持
减少敏感数据在内核停留的时间	不支持	不支持	支持
随机化用户进程的栈	不支持	不支持	支持
加固 ASLR 防止信息泄漏和 entropy 减少	不支持	不支持	支持
遏制针对用户态和内核态的爆破	不支持	不支持	支持
防御 64-bit 内核栈溢出漏洞	不支持	不支持	支持

内核加固方案对比

SELinux ：

优势：

RedHat 商业支持， CentOS/Fedora 社区支持

在最大程度上防御用户态漏洞

劣势：

规则难以编写

维护和调试成本较高

无法防御内核漏洞

内核加固方案对比

AppArmor ：

优势：

SUSE/Canonical 商业支持， Ubuntu/OpenSuSE 社区支持

中等程度上防御用户态漏洞

编写规则简单

维护和调试成本低

劣势：

无法防御内核漏洞

内核加固方案对比

PaX/Grsecurity ：

grsecurity

优势：

强大的社区支持， Debian/Gentoo

防御未知 (0Day) 漏洞利用的能力

能防御内核本身的漏洞

劣势：

没有商业 GNU/Linux 发行版原生支持

业务回归测试成本较高

Hardenedlinux 社区建议

* 如果你是 Five-Eye 成员国的政府部门，你应该使用 SELinux 。

* 如果你是普通个人用户，你可以使用 AppArmor 。

* 如果你是重视安全的个人用户（ Anarchist/ 自由软件狂热分子）和企业用户，请考虑使用 AppArmor 或者 PaX/Grsecurity 。

* 如果你需要保护 critical 环境的基础设施，你应该使用 PaX/Grsecurity 。

cat /proc/agenda

* **********

* **********

* **********

* 企业安全现状

* **********

网络安全技术进化

1990s

- Packet Firewall
- Stateful Firewall(ipfw, iptables)

2000s

- IDS/IPS(Snort, Suricata)

2004

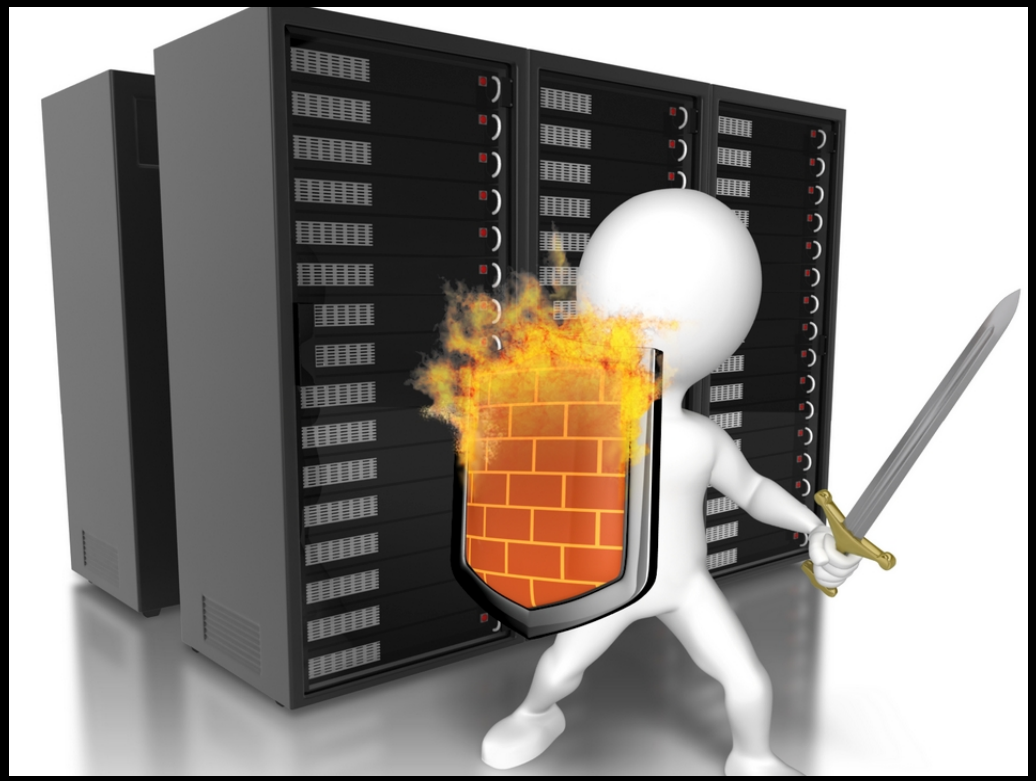
- 追求功能的 UTM(Untangle)

2010s

- 追求性能的 NGFW
- 未知威胁防御

假设对吗？

我们有一堆设备，很安全？



往往故事发展成这样？

我们有一堆设备，很安全？

怎么进入我们的系统的？
How the hell did they get into our system?

好像出自你的电脑 夫人
It appears to be your computer, ma'am.

并且已经穿透了我们的防火墙
Correction. This is behind our firewall.

绝望的守护者

复杂攻击与精妙利用层出不穷—绝望的防守者？

信息安全防御变得越来越困难，感觉不管采用怎么强悍的防御手段，都很快会被一些“精妙”的攻击所击破：

- APT攻击让传统防御手段变得形同虚设；
- 信息交互的刚需使网络隔离难以奏效；
- 各种宣称“解决一起安全问题”的防御技术很快被绕过...

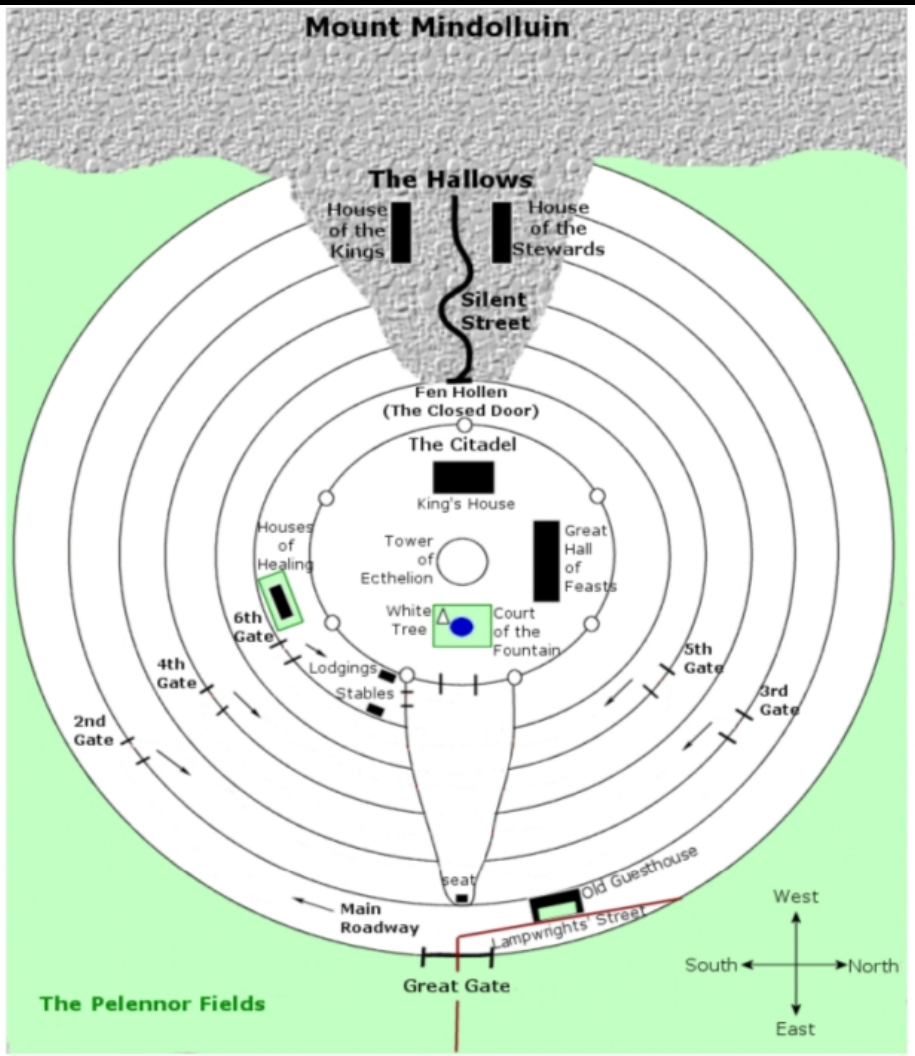
信息安全防守看起来那么让人绝望，重要信息系统就像游戏中的BOSS那样，最终逃不过被打垮的命运...

面对”丧尸”式的攻击

单点防御还有效吗？

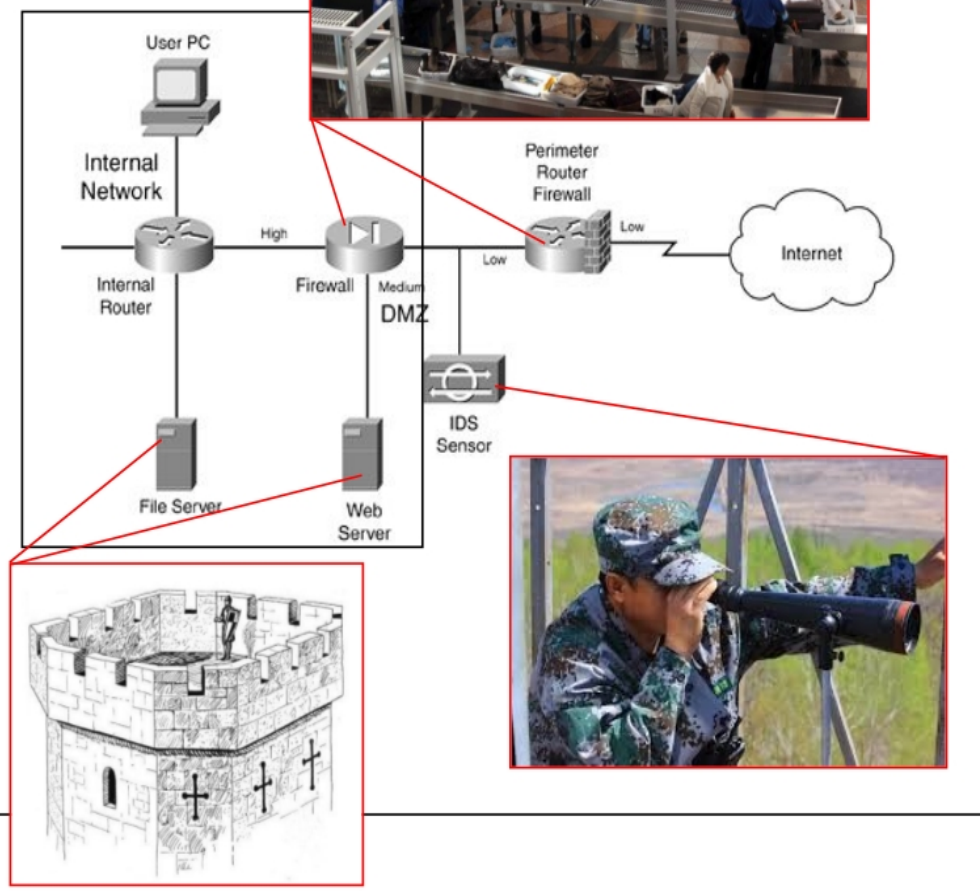
纵深防御是唯一的选择？

纵深防御典范 _ 魔戒之刚铎首都 (Minas Tirith)



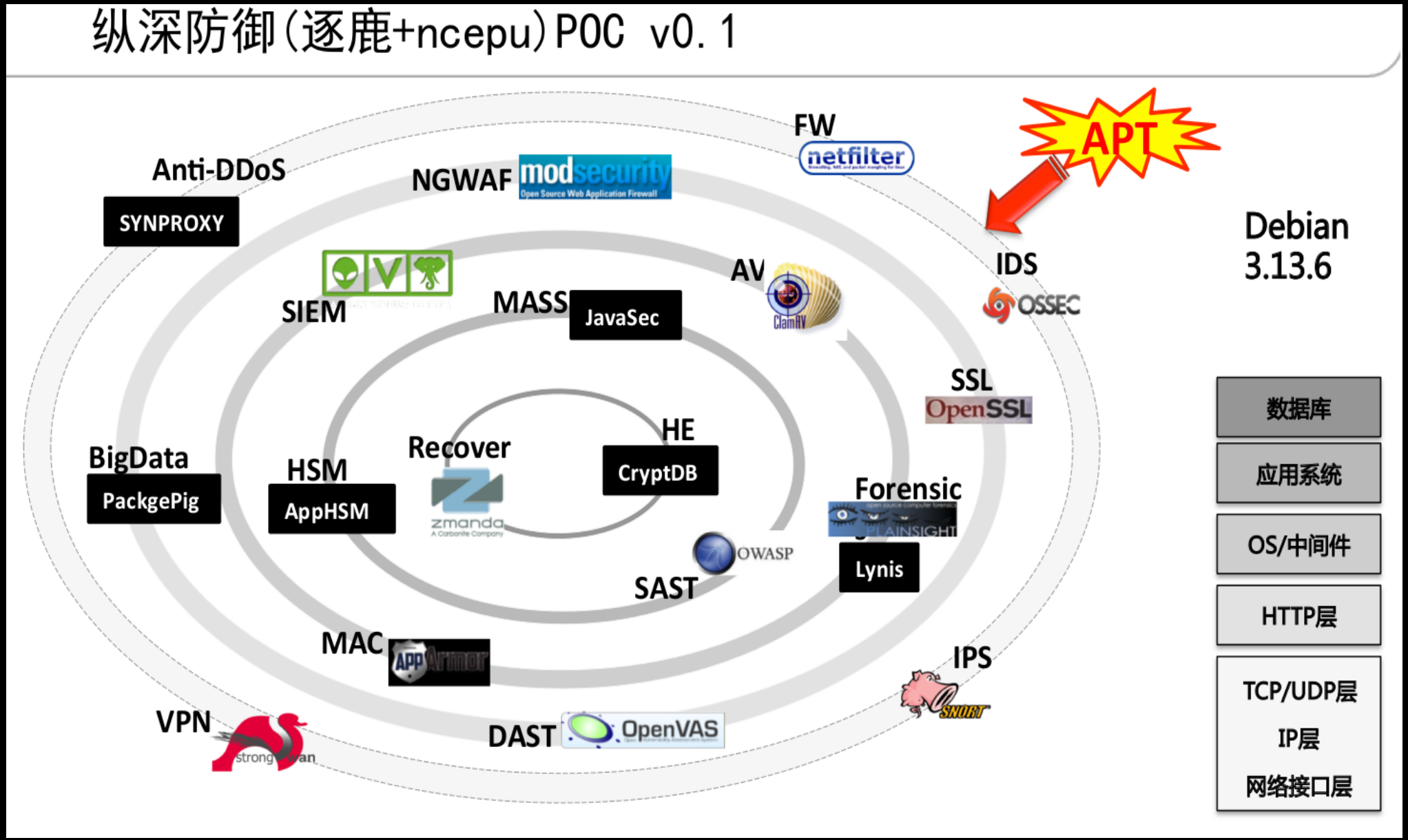
纵深防御是堆积木？

理想目标



无奈现状

自由软件组合能做什么？


纵深防御(逐鹿+ncepu)POC v0.1
APT
FW
netfilter
Anti-DDoS
SYNPROXY
NGWAF
modsecurity
Open Source Web Application Firewall
IDS
OSSEC
Debian
3.13.6
SIEM
MASS
JavaSec
AV
ClamAV
SSL
OpenSSL
BigData
PackgePig
HSM
AppHSM
Recover
zmanda
A Carbonite Company
HE
CryptDB
Forensic
PLAINSIGHT
OWASP
SAST
Lynis
MAC
AppArmor
IPS
SNORT
VPN
strongSwan
DAST
OpenVAS
数据库
应用系统
OS/中间件
HTTP层
TCP/UDP层
IP层
网络接口层

Know Your Enemy

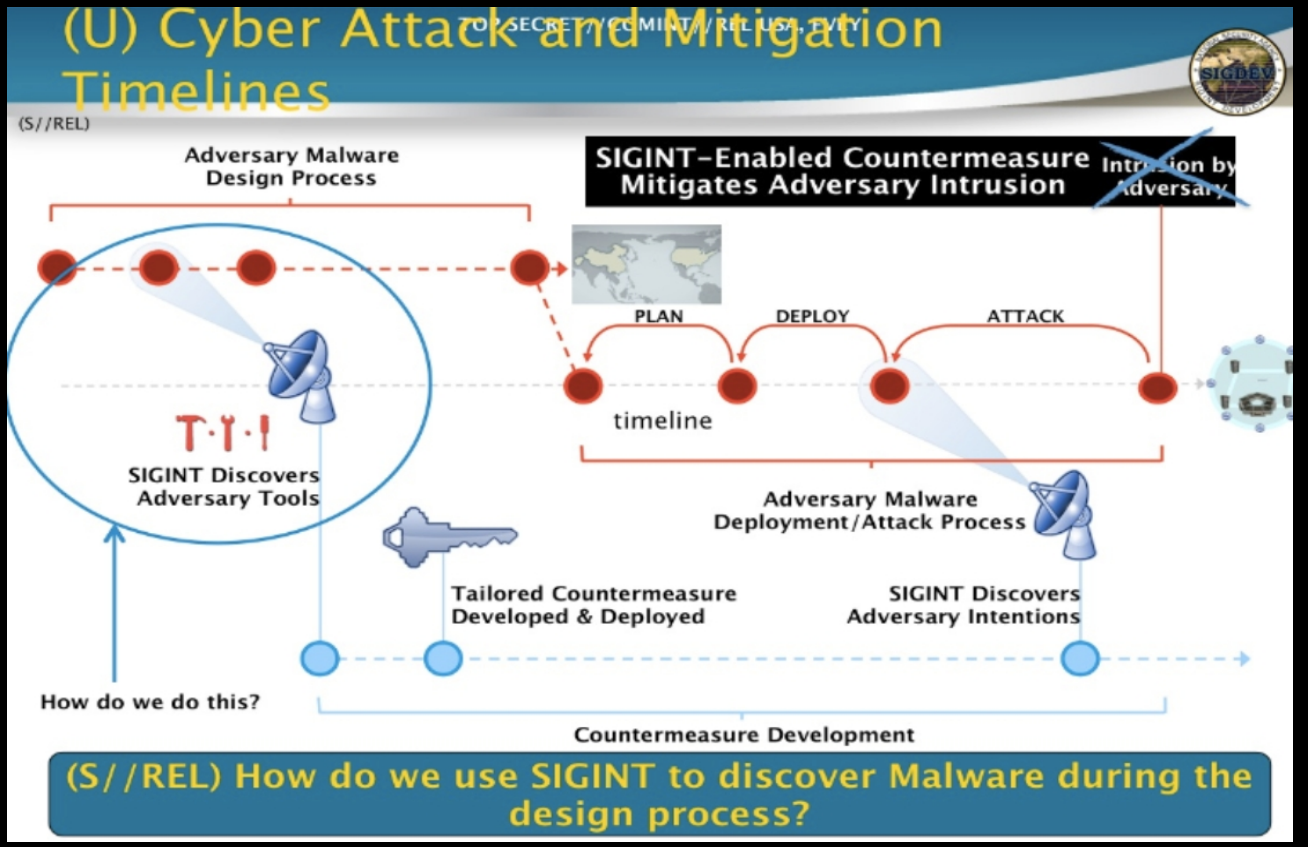
知己知彼，百战不殆！

未来攻防的关键：威胁情报搜集与分析

- OSINT
- OPSEC

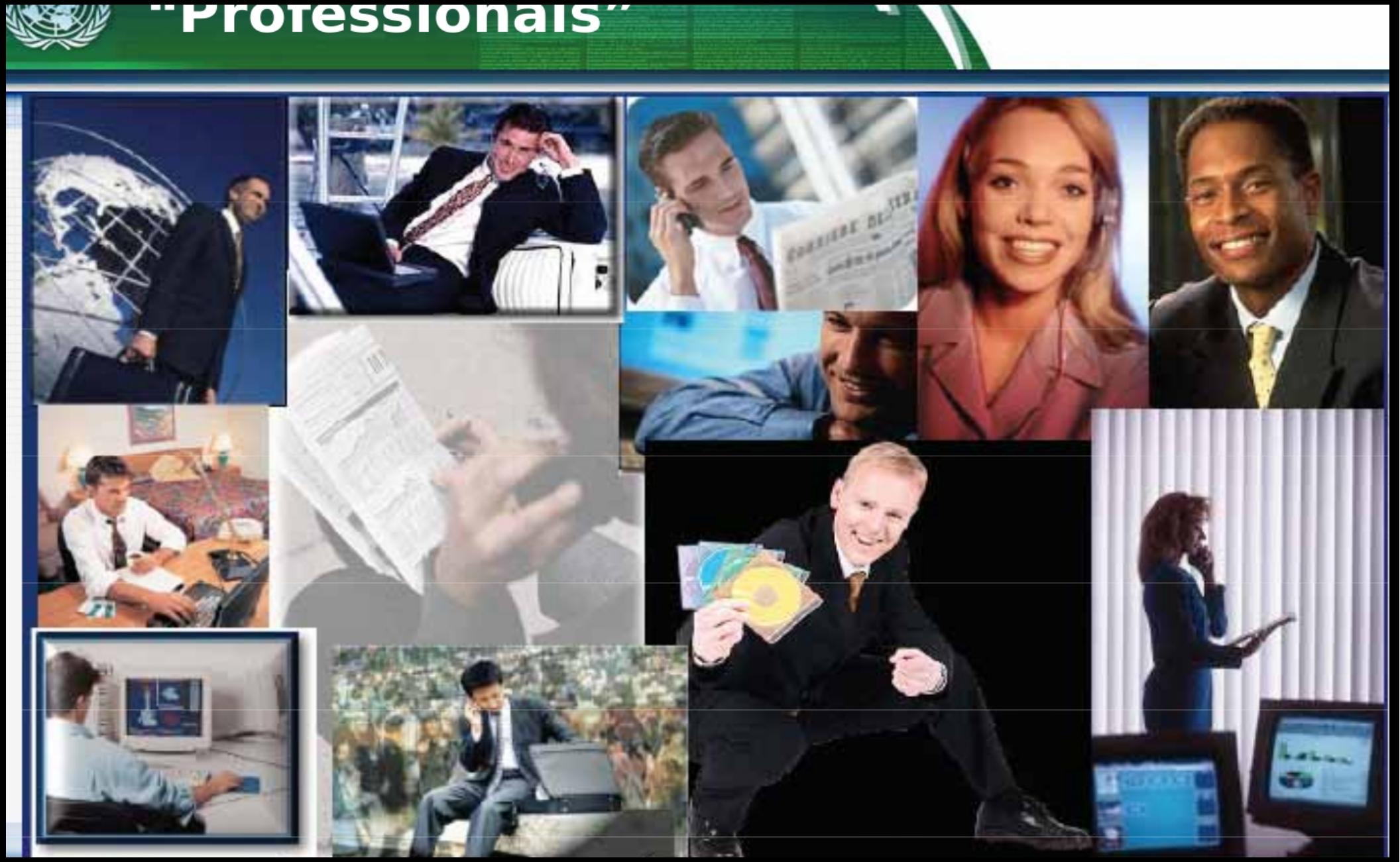
安全事件高发区：

- 代码质量缺陷
- 运维失误



攻击者到底是谁？

攻击者到底是谁？



公开情报搜集

GCHQ 对公开情报的搜集目标：

TOP SECRET STRAP1 COMINT

The maximum classification allowed on GCWiki is **TOP SECRET STRAP1 COMINT**. Click to report inappropriate content.

For GCWiki help contact: webteam [REDACTED] Support page

Open Source for Cyber Defence/Progress

From GCWiki

< Open Source for Cyber Defence

Jump to: navigation, search

Many structured datasets are now available in the HAPPY TRIGGER database. Unstructured datasets are being worked on and will go to LOVELY HORSE. Other integration with TWO FACE and ZooL is in place, and more will come to XKEYSCORE

Contents

公开情报搜集

漏洞的公开情报对同样重要：

EmergingThreats.net	Snort rules used for network monitoring purposes	Approved (for Free data)
PremiumDrops.com	Daily newly registered domains to alert analysts to suspicious domains worth investigating for malicious activity	Approved
verisign.com	Monthly updates of newly registered domains to alert analysts to suspicious domains worth investigating for malicious activity	Approved
MalwareDomainList.com	General malware tracking resource	Approved
twitter.com	Real-time alerting to new security issues reported by known security professionals, or planned activity by hacking groups e.g. Anonymous. For more information about the sources currently being brought into the building see source list on the LOVELY HORSE wiki	Approved
ContagioMiniDump.com	Most recommended blog by CDO analysts. Highly regarded for malware analysis relevant to APT investigations. Can be useful to declassify information for reporting purposes	Approved
metasploit.com	Access to new zero-day exploits for the malware team to analyse	Approved (for free data)
exploit-db.com	Access to an archive of exploits and vulnerable software. Exploits from submittals and mailing lists collected into one database.	Approved

[edit] Vulnerability Intelligence

Knowledge required	Available from	Update frequency	Filtering	Volumetrics	Comments
twitter traffic for vulnerabilities	use twitter API in standard way	hourly?	by twitter names of known malware/vulnerability researchers	very small (MB)	Current work is BIRD SEED. JTRIG's BIRDSTRIKE provides the scraping already, but only for handfuls using Cyber Cloud, and has OPP-LEG approval already.
certain blogs and CERT web sites for vulnerabilities	direct web scrape (if allowed). MHS OSINT pages have examples?	hourly?	by list of specific sites/pages	small (GB)	TR-CISA have previously run several contracts looking at this problem, with a view to delivery to CNE. Fir source information such that machines matching those rule (vulnerabilities) can be found in passive. Wanted
certain CERT IRC chatrooms for vulnerabilities	direct IRC access (if allowed)	hourly?	by list of specific IRCs	v.small (MB)	NB: Assume will include some encrypted IRCs. Wanted by GovCERT. Maybe a MARBLE POLLS source
certain CERT email lists for vulnerabilities	direct reception	hourly?	by list of specific mailing lists	v.small (MB)	NB: Assume will include some encrypted email (including PGP). Wanted by GovCERT. Maybe a MARBL
Commits to open source code repositories and security patch check-ins	GitHub etc.	daily?	by specific code projects, presumably	small (GB)	Requested by NDR [REDACTED].
Emerging Threats 'Open'	Scraped via SHORTFALL framework	Daily?	By updated Snort rules	???	Approval granted from OP-LEG to scrape info.

RedHat 的漏洞风险评估

Red Hat 的公开漏洞信息库：

https://access.redhat.com/security/cve/

关注”Important”和”Critical”：

CVE DATABASE

Red Hat vulnerabilities by CVE name

The Common Vulnerabilities and Exposures (CVE) project, maintained by The MITRE Corporation, is a list of standardized names for vulnerabilities and security exposures.

2015 | 2014 | 2013 | 2012 | 2011 | 2010 | 2009 | 2008 | 2007 | 2006 | 2005 | 2004 | 2003 | 2002 | 2001 | 2000 | 1999

Show 10 entries Filter:

CVE	Synopsis	Impact	Public Date
CVE-2015-0138	GSKit in IBM Tivoli Directory Server (ITDS) 6.0 before 6.0.0.73-ISS-ITDS-IF0073, 6.1 before 6.1.0.66-ISS-ITDS-IF0066, 6.2 before 6.2.0.42-ISS-ITDS-IF0042, and 6.3 before 6.3.0.35-ISS-ITDS-IF0035 and IBM Security Directory Server (ISDS) 6.3.1 before 6.3.1.9-ISS-ISDS-IF0009 does not properly restrict TLS state transitions, which makes it easier for remote attackers to conduct cipher-downgrade attacks to EXPORT_RSA ciphers via crafted TLS traffic, related to the "FREAK" issue, a different vulnerability than CVE-2015-0204.	Moderate	2015-03-11
CVE-2015-0192	** RESERVED ** This candidate has been reserved by an organization or individual that will use it when announcing a new security problem. When the candidate has been publicized, the details for this candidate will be provided.	Critical	2015-05-06
CVE-2015-0201	The Java SockJS client in Pivotal Spring Framework 4.1.x before 4.1.5 generates predictable session ids, which allows remote attackers to send messages to other sessions via unspecified vectors.	Low	2015-03-06

已知漏洞评估失误？

Cybersec consultant sucks? Who we should listen to? 0Ld sch00l hackers? Or xyz-certified-random-it-guy?

虽然是否出现公开漏洞利用的代码和漏洞利用成功的危害程度是重要的评估要素，，但并不代表说没有公开的漏洞利用就不存在风险，这里举一个 BadiRET 漏洞的例子:

Linux 内核代码文件 arch/x86/kernel/entry_64.S 在 3.17.5 之前的版本都没有正确的处理跟 SS（堆栈区）段寄存器相关的错误，这可以让本地用户通过触发一个 IRET 指令从错误的地址空间去访问 GS 基地址来提权。 这个编号为 CVE-2014-9322,漏洞于 2014 年 11 月 23 日被 Linux 内核社区修复，之后的几个礼拜里没有出现公开的利用代码甚至相关的讨论。 当人们快要遗忘这个威胁的时候，Rafal Wojtczuk 于 2015 年 2 月公布了分析文章 Exploiting “BadIRET” vulnerability 证实了这个漏洞虽然利用极其困难，但并非不可能。 Rafal 在 Fedora 20 64-bit GNU/Linux 发行版上完成了研究和测试工作，内核是 3.11.10-301，并且证明只有 PaX/Grsecurity 类似的 UDEREF 技术才能完全阻止此类漏洞利用。

另外一方面，Rafal 也谈到说这个如此严重的漏洞居然数月都没有公开讨论，但其实在 2014 年 12 月中旬俄文的安全社区就已经进行了详细讨论并最终给出了 PoC。 这是一次针对非英文世界的公开威胁情报分析的重大失误的典型案例。 既然俄文安全社区已经公布了 PoC 代码，那离稳定的漏洞利用就不远了，从这个案例可以看出，在做已知漏洞的风险评估时一定要考虑斯拉夫兵工厂的实力。

漏洞利用"BadIRET"分析(CVE-2014-9322, Linux内核提权)

http://hardenedlinux.org/system-security/2015/07/05/badiret-analysis.html

后续故事：数字军火级别的"BadIRET"漏洞利用(CVE-2014-9322)

http://hardenedlinux.org/system-security/2015/07/05/badiret-exp.html

已知漏洞评估

只关注公开漏洞利用的情况已经无法满足企业需求，多维度分析：

利用难度

是否有公开利用？

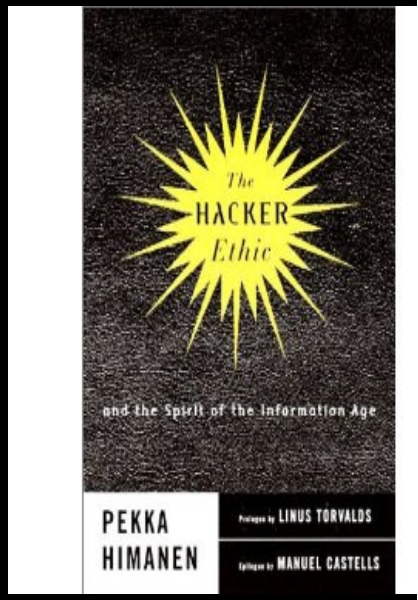
漏洞信息评估

军火成本

ADOBE READER	$5,000-$30,000
MAC OSX	$20,000-$50,000
ANDROID	$30,000-$60,000
FLASH OR JAVA BROWSER PLUG-INS	$40,000-$100,000
MICROSOFT WORD	$50,000-$100,000
WINDOWS	$60,000-$120,000
FIREFOX OR SAFARI	$60,000-$150,000
CHROME OR INTERNET EXPLORER	$80,000-$200,000
IOS	$100,000-$250,000

推荐读物

* For further reading/hacking.
* Book
  * The HACKER Ethic and the Spirit of Information Age
*Documentary film
  * Revolution OS
* The best security ezine
  * Phrack



关注我们

* Hardenedlinux 社区， http://hardenedlinux.org

* 使用自由软件的方案加固一切

* 狂热的自由软件玩家以及 Anarchy(Anarchy 翻译成“反权威主义”更准确)

* 关注企业安全

cat /proc/agenda

* **********
* **********
* **********
* **********
* 加固案例

网络产品加固

场景	加固方案
硬件 : x86_64	3.14.x with UDEREF
Debian 8 GNU/Linux 物理机安全	常规安全部署 + STIG-complianced
所有应用运行在 container/Docker 中，防御逃逸	容器的逃逸成本远远低于虚拟机，所以使用 PaX/Grsecurity + RBAC
ElasticSearch 防护	Sheild 插件

未完成的版本：

http://hardenedlinux.org/system-security/2015/09/06/hardening-es-in-docker-with-grsec.html

感谢 & 引用

感谢 Phreaker 分享的塔防理论和 Thomas Biege 提供的部分素材。

http://hardenedlinux.org/system-security/2015/06/09/debian-security-chklist.html

http://hardenedlinux.org/cryptography/2015/07/28/ssl-tls-deployment-1.4.html

http://grsecurity.net/

https://www.nth-dimension.org.uk/pub/BTL.pdf

http://www.defensecode.com/public/DefenseCode_Unix_WildCards_Gone_Wild.txt

https://www.ibm.com/developerworks/rational/library/11-proven-practices-for-peer-review/

Questions?

Thanks ! Any ideas about free software security ? Drop me a line:

<Shawn the R0ck, citypw(AT)gmail.com>