方程式组织SPARC架构样本分析调试

安天安全研究与应急处理中心（CERT）白淳升

课程提纲Contents

- 方程式组织回顾
- SPARC架构分析
- 动态调试技术

方程式组织回顾

方程式组织

- Equation 恶意软件星系的死星
- Fanny Equation 我是你的父亲, Stuxnet 蠕虫
- 善意的休斯顿Equation组织
- Equation组织问与答
- EquationDrug间谍平台剖析
- 修改硬盘固件的木马 探索方程式（EQUATION）组织的攻击组件
- 从方程式到“方程组”EQUATION攻击组织高级恶意代码的全平台能力解析

Equation Group: from Houston with love

February 19, 2015, 9:00 am. GReAT

f 45 g+ 12 29724 1

In 2009, an international scientific conference was held in Houston. The organizers sent out a post-meeting CDROM. The disk used in the Houston attack represents a rare and unusual operation for the Equation Group. Read Full Article

A Fanny Equation: “I am your father, Stuxnet”

February 17, 2015, 9:00 am. GReAT

f 107 g+ 13 41976 4

During our 2014 research into the Equation group, we created a special detection for the group’s exploitation library, codenamed “PrivLib”. To our surprise, this detection triggered a worm from 2008 that used the Stuxnet LNK exploit to replicate, codenamed Fanny. Read Full Article

Equation: The Death Star of Malware Galaxy

February 16, 2015, 6:55 pm. GReAT

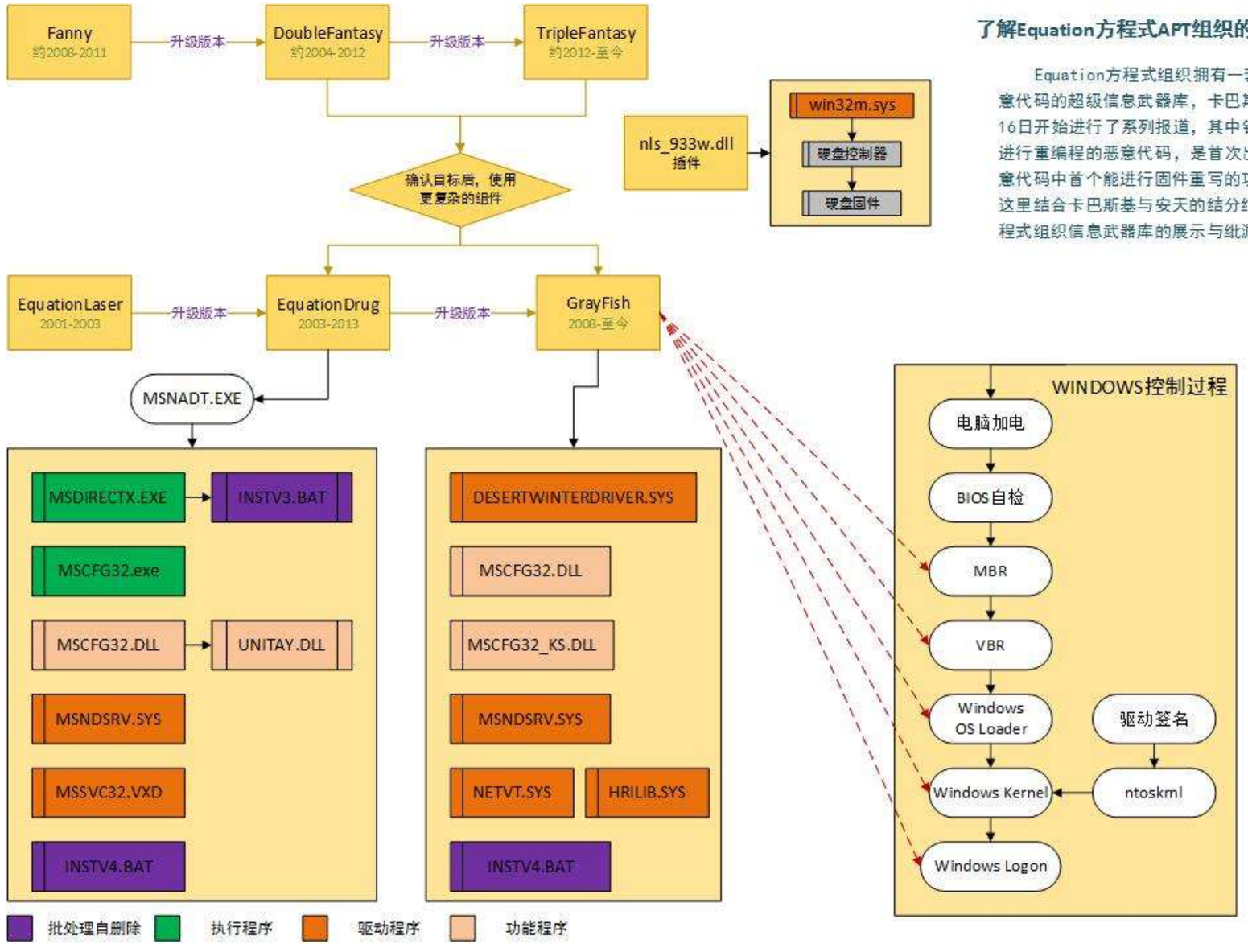
f 950 g+ 138 237724 28

The Equation group is a highly sophisticated threat actor that has been engaged in multiple CNE (computer network exploitation) operations dating back to 2001, and perhaps as early as 1996. It is probably one of the most sophisticated cyber attack groups in the world. Read Full Article

https://securelist.com

攻击平台-组件

- Laser（激光器）
- Drug（麻醉药）
- Fantasy（幻想曲）
- Double Fantasy（双重幻想）
- Triple Fantasy（三重幻想）
- Fanny（范妮）
- GrayFish（灰鱼）



了解Equation方程式APT组织的信息武器库

Equation方程式组织拥有一套用于植入恶意代码的超级信息武器库，卡巴斯基在在2月16日开始进行了系列报道，其中针对硬盘固件进行重编程的恶意代码，是首次出现，也是恶意代码中首个能进行固件重写的功能被发现。这里结合卡巴斯基与安天的结分结果，进行方程式组织信息武器库的展示与纰漏。

感染

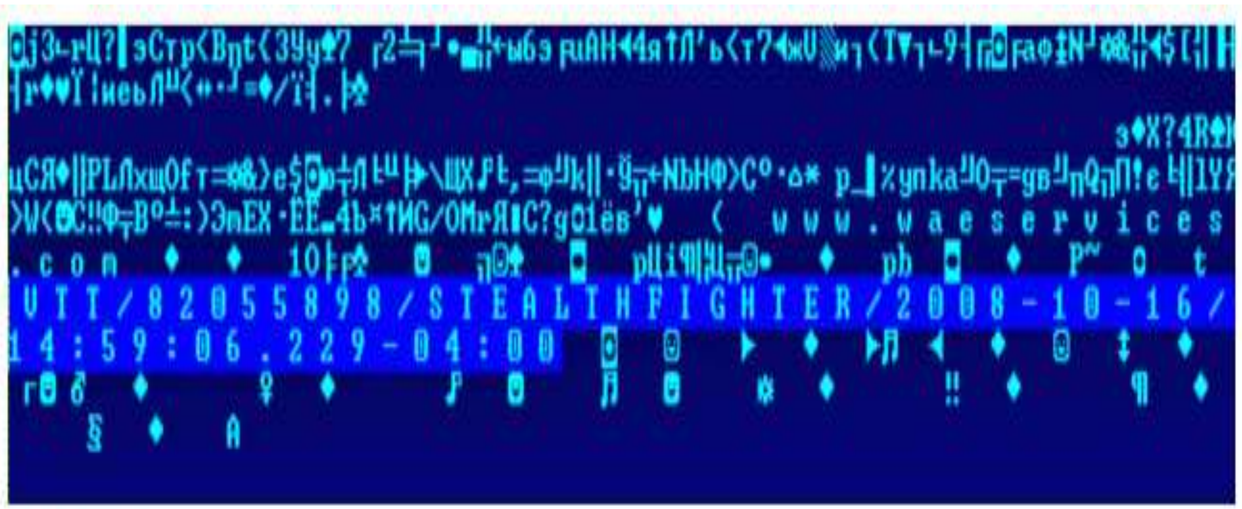
- 自我复制（蠕虫）代码 – Fanny
- USB存储器+漏洞
- 基于Web的漏洞
- 物理介质，CD-ROM

NSA、TAO、物流链攻击

- 首先，**截获**运往世界各地目标地点的计算机网络设备（服务器、路由器等），将其**转运至某秘密地点**。
- 然后，**将攻击载荷植入直接安装到目标电子设备中**。
- 最后，这些设备再经过**重新打包转运至原目的地**。

BACKSNARF项目

- 关联NSA的新证据为字符串"BACKSNARF_AB25"，该字符串于几天前在新发现的"EquationDrug"平台中发现。
- NSA陈述中第19页规定，"BACKSNARF"是一个关联到NSA的TAO(Tailored Access Operation，专门接入行动)的名称。



Enlarge

Kaspersky Lab

硬盘固件修改 80AA nls_933w.dll

- 针对性
  - 不会轻易使用，只为特别重要的计算机准备
  - 卡巴披露500个受害者中只有5台被攻击
- 感染固件
  - 磁盘格式化和重新安装系统后仍存在
  - 作为一个不可见的、持久存储隐藏在硬盘内部
  - 主流厂商全覆盖
  - 反病毒程序无法查杀

- IoControlCode为0x870021D0时，nls_933w.dll对硬盘控制器发送ATA控制指令对硬盘控制器进行操作

```
0012F890  1000A221  CALL 到 DeviceIoControl 来自 83d14ce2.1000A21B
0012F894  000000A8  hDevice = 000000A8 (window)
0012F898  870021D0  IoControlCode = 0x870021D0
0012F89C  003C3E98  InBuffer = 003C3E98
0012F8A0  0000024C  InBufferSize = 24C (588.)
0012F8A4  003C3E98  OutBuffer = 003C3E98
0012F8A8  0000024C  OutBufferSize = 24C (588.)
0012F8AC  0012F8BC  pBytesReturned = 0012F8BC
0012F8B0  00000000  pOverlapped = NULL
0012F8B4  0012F970
```

获取硬盘信息

地址	HEX 数据																ASCII
003C3E98	00	00	00	00	00	00	00	00	01	00	00	00	00	00	00	00	........ƒ.......
003C3EA8	01	00	00	00	00	00	00	00	38	02	00	00	00	00	00	00	ƒ.......8¬.....
003C3EB8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
003C3EC8	00	00	00	00	01	00	00	00	00	01	00	00	02	00	00	00	....ƒ...ƒ.¬...
003C3ED8	00	00	00	00	06	00	00	00	00	00	07	EC	7A	42	FF	3F	....■.....■勒Bÿ?
003C3EE8	00	00	0F	00	00	00	00	00	3F	00	00	00	00	00	00	00	..■.....?.......
003C3EF8	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
003C3F08	30	30	31	30	00	00	40	00	00	00	30	30	30	30	30	30	0010..@...000000
003C3F18	31	30	4D	56	61	77	65	72	56	20	72	69	75	74	6C	61	10MVawerV riutla
003C3F28	49	20	45	44	48	20	72	61	20	64	72	44	76	69	20	65	I EDH ra drDvi e
003C3F38	20	20	20	20	20	20	20	20	20	20	40	80	00	00	00	2F	@■.../
003C3F48	00	00	00	02	00	00	07	00	43	44	0F	00	3F	00	53	FB	...¬.■.CD■.?.S∅
003C3F58	FB	00	40	01	00	00	00	05	00	00	07	00	03	00	78	00	?@ƒ..¥.■. .x.
003C3F68	78	00	A0	00	78	00	00	00	00	00	00	00	00	00	00	00	x.?x............
003C3F78	00	00	00	00	00	00	00	00	00	00	00	00	1E	00	17	00	............■.■.
003C3F88	08	40	08	40	00	41	08	40	00	00	00	41	07	04	00	00	■@■@.A■@...A■ ¦..
003C3F98	00	00	00	40	00	00	00	00	00	00	00	00	00	00	00	00	...@............
003C3FA8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
003C3FB8	00	00	00	00	00	50	9E	C2	55	BA	29	9B	00	00	00	00	.....P灺U??...

硬盘厂商名

- 加密的硬盘厂商名
- 西部数据、三星、迈拓、希捷等主流硬盘厂商

```
ROM:1002F2A7                db  7Eh ; ~
ROM:1002F2A8 aWdcWd_0       db 'WDC WD',0           ; DATA XREF: sub_1
ROM:1002F2A8                                        ; sub_10007090+20↑r
ROM:1002F2AF                db    0
ROM:1002F2B0 unk_1002F2B0   db    1                 ; DATA XREF: sub_1
ROM:1002F2B1                db    8
ROM:1002F2B2                db    0
ROM:1002F2B3                db 0EEh ;
ROM:1002F2B4 aSamsung       db 'SAMSUNG',0          ; DATA XREF: sub_1
ROM:1002F2B4                                        ; sub_100073C0+11↑r
ROM:1002F2BC byte_1002F2BC  db 0                    ; DATA XREF: sub_1
ROM:1002F2BD                align 10h
ROM:1002F2C0 unk_1002F2C0   db    1                 ; DATA XREF: sub_1
ROM:1002F2C1                db    7
ROM:1002F2C2                db    0
ROM:1002F2C3                db 0B4h ;
ROM:1002F2C4 aWdcWd         db 'WDC WD',0           ; DATA XREF: sub_1
ROM:1002F2C4                                        ; sub_10007550+26↑r
ROM:1002F2CB                db    0
ROM:1002F2CC unk_1002F2CC   db    1                 ; DATA XREF: sub_1
ROM:1002F2CD                db  0Bh
ROM:1002F2CE                db    0
ROM:1002F2CF                db 0AAh ;
ROM:1002F2D0 aMaxtorStm_0   db 'Maxtor STM',0       ; DATA XREF: sub_1
ROM:1002F2D0                                        ; sub_10007740+1B↑r
ROM:1002F2DB                db    0
ROM:1002F2DC unk_1002F2DC   db    1                 ; DATA XREF: sub_1
ROM:1002F2DD                db    3
ROM:1002F2DE                db    0
ROM:1002F2DF                db  34h ; 4
ROM:1002F2E0 unk_1002F2E0   db  53h ; S             ; DATA XREF: sub_1
ROM:1002F2E1                db  54h ; T
ROM:1002F2E2                db    0
ROM:1002F2E3                db    0
ROM:1002F2E4 unk_1002F2E4   db    1                 ; DATA XREF: sub_1
ROM:1002F2E5                db  0Bh
ROM:1002F2E6                db    0
ROM:1002F2E7                db  83h ;
ROM:1002F2E8 aMaxtorStm     db 'Maxtor STM',0       ; DATA XREF: sub_1
ROM:1002F2E8                                        ; sub_10007910+36↑r
```

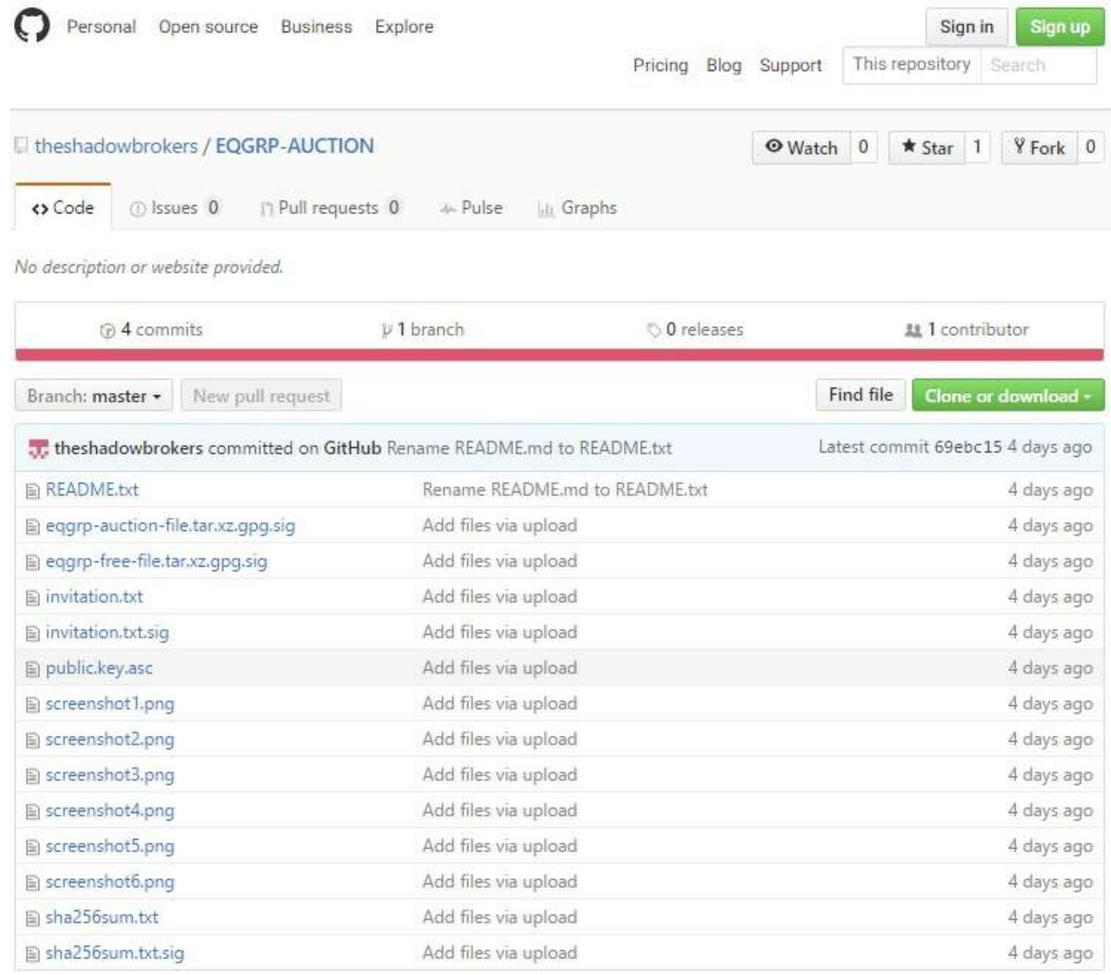
方程式还有什么能力 – 跨平台?

- Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_2) AppleWebKit/536.26.17 (KHTML, like Gecko) Version/6.0.2 Safari/536.26.17
- Mozilla/5.0 (Macintosh; Intel Mac OS X 10.8; rv:21.0) Gecko/20100101Firefox/21.0
- Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_3) AppleWebKit/536.28.10 (KHTML, like Gecko) Version/6.0.3 Safari/536.28.10

- **Agent信息表明方程式组件存在其他架构、系统**
- **Windows、Linux、Mac OS X、Android、iOS**

影子经纪人

- 2016年8月13日，黑客组织Shadow Brokers声称攻破了“为NSA开发网络武器的美国黑客团队Equation Group”
  - “eqgrp-free-file.tar.xz.gpg”
  - “eqgrp-auction-file.tar.xz.gpg”



泄露文件内容

- 通过对泄露文件中CPU的架构分析，我们发现文件的架构有X86、MIPS、PowerPC、SPARC架构，其中X86最多，SPARC最少，都是针对防火墙系统的攻击程序。

文件夹名称	猜测功能
BANANAGLEE	针对思科、瞻博防火墙的后门工具
BARGLEE	针对junos防火墙的攻击框架
BLATSTING	针对Fortinet和TOPSEC的攻击工具
BUZZDIRECTION	针对Fortinet的攻击工具
EXPLOITS	针对防火墙溢出攻击的工具集合
OPS	进行攻击的一些自动化脚本集合
SCRIPTS	一些攻击的使用方法和记录
TOOLS	进行攻击的一些常用攻击集合
TURBO	一些其他攻击工具

入侵清单

- 2016年11月，影子经纪人公开了曾一份遭受入侵的服务器清单，清单的日期戳显示，各系统遭受入侵的时间在2000年到2010年之间，受影响的国家包括中国、日本、韩国、西班牙、德国、印度等。

域名/主机名	IP地址	系统	时间戳
mail.a-1.net.cn	210.77.147.84	sparc-sun-solaris2.7	20010313-132258
noya.bupt.edu.cn	202.112.96.2	sparc-sun-solaris2.7	20010810-101659
noya.bupt.edu.cn	202.112.96.2	sparc-sun-solaris2.7	20011022-132036
cad-server1.ee.nctu.edu.tw	140.113.212.150	sparc-sun-solaris2.5.1	20040322-113315
mail.tccn.edu.tw	203.64.35.108	sparc-sun-solaris2.8	20041018-114628
ns.chining.com.tw	202.39.26.50	i386-pc-solaris2.8	20041022-114055
mail.ncue.edu.tw	163.23.225.100	sparc-sun-solaris2.8	20041022-122302
ns.bigobe.net.tw	202.166.255.98	sparc-sun-solaris2.9	20041022-125517
mail.must.edu.tw	203.68.220.40	sparc-sun-solaris2.8	20041022-133501
ultra10.nanya.edu.tw	203.68.40.6	sparc-sun-solaris2.8	20041227-123134
mail.et.ntust.edu.tw	140.118.2.53	sparc-sun-solaris2.8	20050315-111913

从方程式到“方程组”

EQUATION 攻击组织的全平台载荷能力解析

安天实验室

首次完成时间：2016 年 05 月 23 日 15 时 00 分
本版本更新时间：2016 年 11 月 02 日 9 时 00 分

SPARC架构分析

- 汇编语言
- 架构特点

SPACR 寄存器

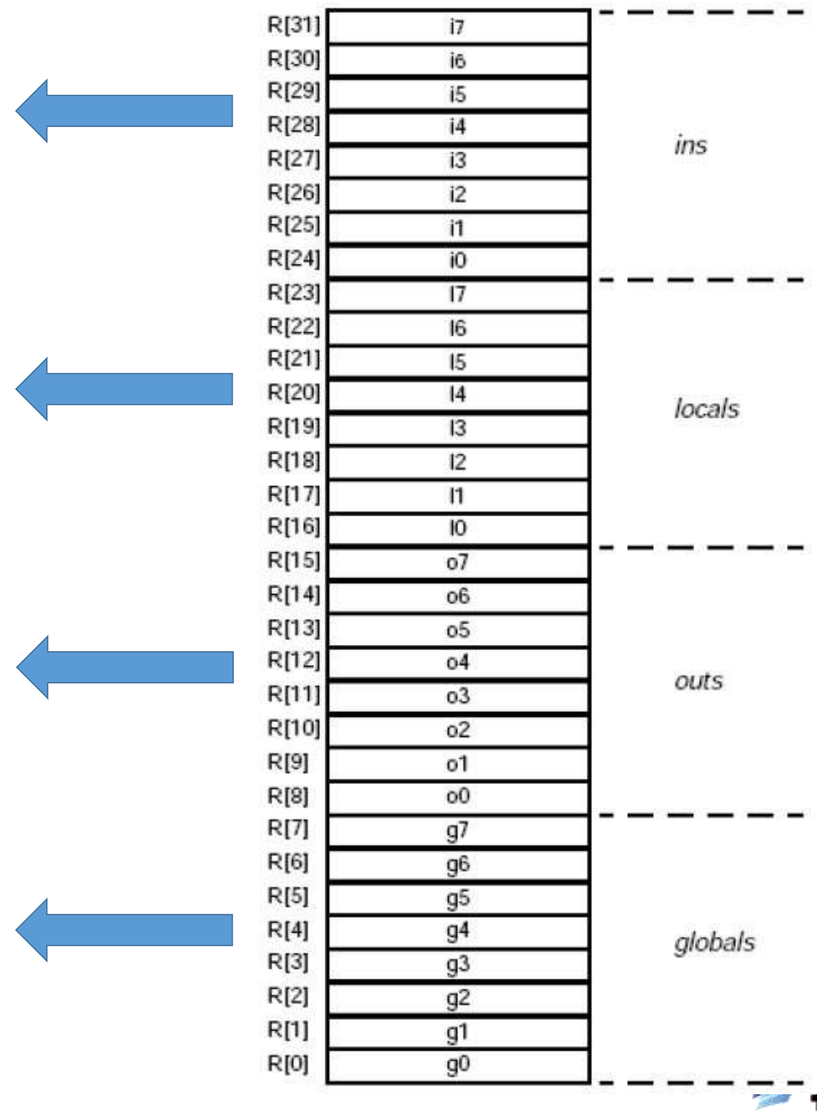
Sparc的通用寄存器不同于X86，共有32个寄存器 r0-r31

输入寄存器(8个) - 命名为 %i0 ~ %i7，
等同于 %r24 ~ %r31

局部寄存器(8个) - 仅本函数可见
命名为 %l0 ~ %l7，等同于 %r16 ~ %r23

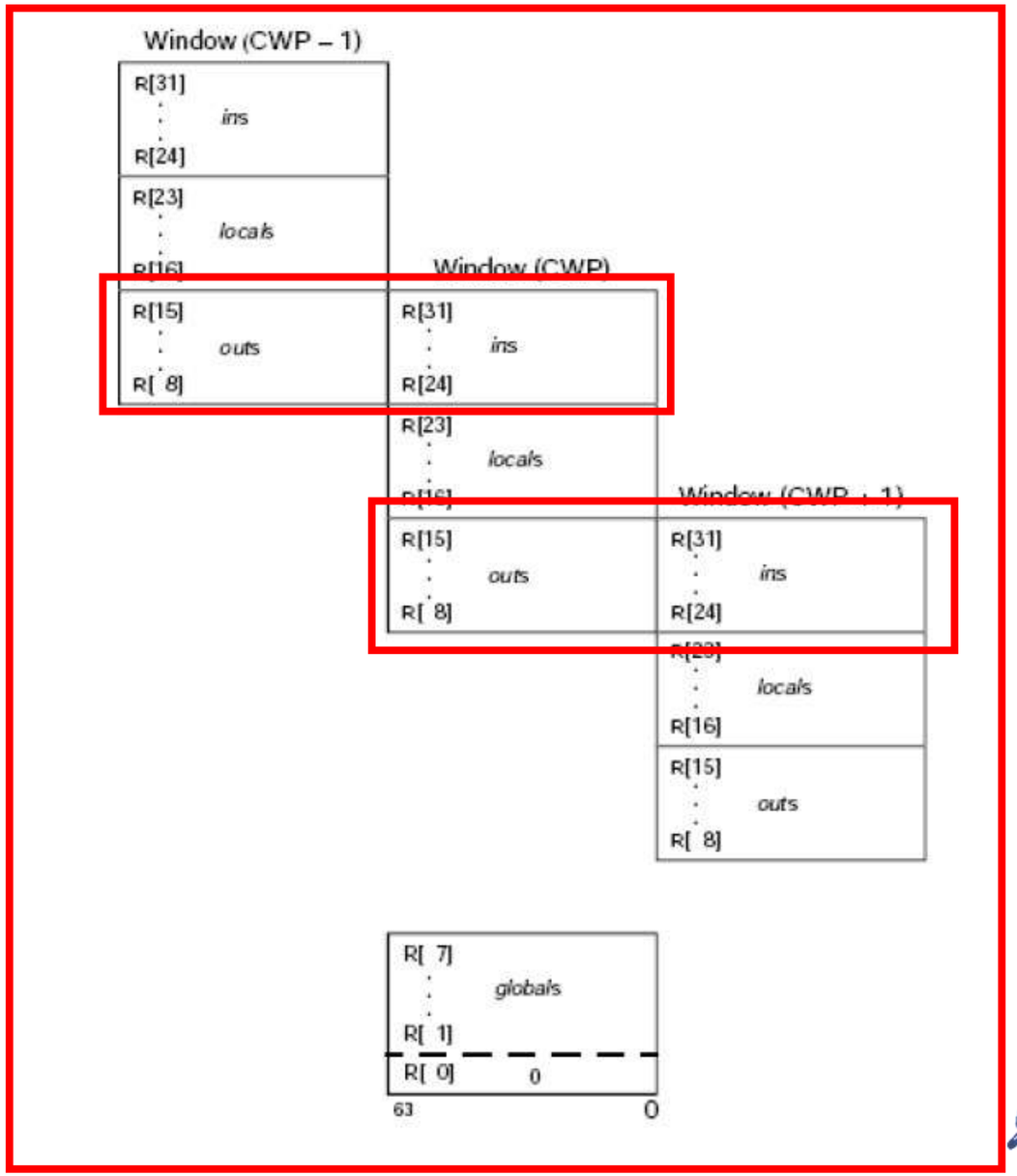
输出寄存器(8个) - 函数返回值
命名为 %o0 ~ %o7，等同于 %r8 ~ %r15

全局寄存器(8个) - 对所有函数可见
命名为 %g0 ~ %g7 等同于%r0 ~ %r7



窗口管理

- 当前窗口的 i0 ~ i7和上一窗口的 o0 ~ o7 对应于同一组物理寄存器；
- 当前窗口o0 ~ o7作为参数传入下层窗口i0 ~ i7
- 当前窗口返回值 i0 ~ i7返回上层窗口的o0 ~ o7
- 所有窗口的local寄存器绝对独立在自己窗口呢

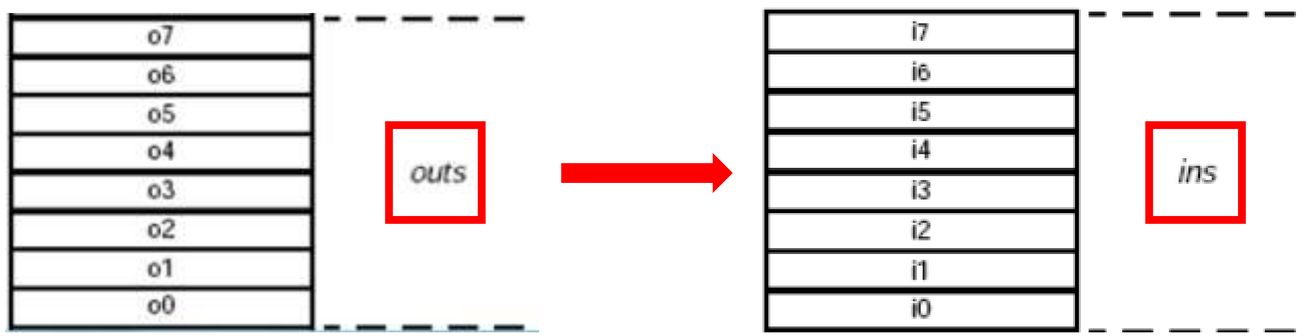


窗口管理-转动

- 为了支持窗口的管理，SPARC使用 CWP (Current Window Pointer)指定当前使用的窗口。

```
00017430      .global QRWLs0NcGpGET9w4OEEm09
00017430      QRWLs0NcGpGET9w4OEEm09:
00017430
00017430      var_18= -0x18
00017430
00017430 000 save     %sp, -0x78, %sp
00017434 078 mov      0, %o4
00017438 078 mov      0, %o5
0001743C 078 sethi    %hi(loc_1BC00), %l7
00017440 078 call     sub_17428
00017444 078 set      loc_1BE2C, %l7  ! 3336C
00017448 078 cmp      %i0, 1
0001744C 078 ble      loc_17534
00017450 078 std      %o4, [%fp+var_18]
```

```
0001760C
0001760C      locret_1760C:
0001760C 078 ret
00017610 078 restore
00017610      ! End of function
```



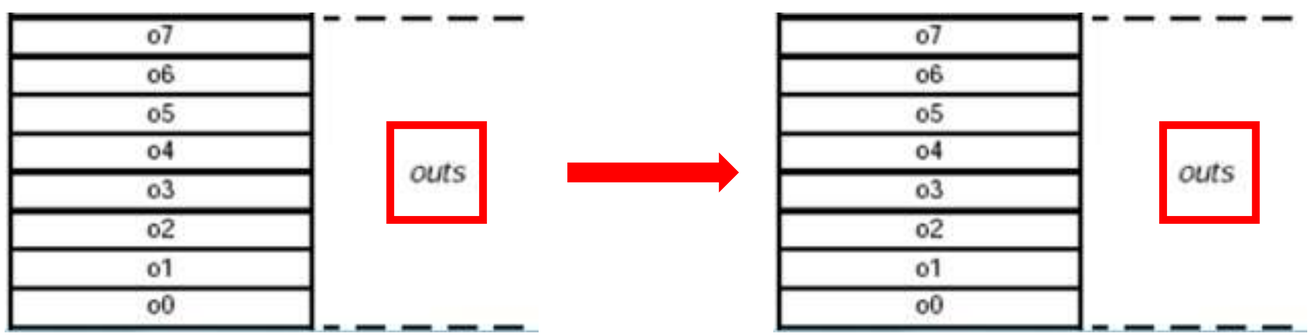
有效防护 价值输出

窗口管理-不转动

- 特定不转动窗口

```
0001A920          .global ogU8PAD0zX3sNtQHqyQ
0001A920          ogU8PAD0zX3sNtQHqyQ:
0001A920 000 cmp        %o0, 0
0001A924 000 be         locret_1A938
```

```
0001A518          .global JiOse7OIYqEafNTnSVnTHTg
0001A518          JiOse7OIYqEafNTnSVnTHTg:
0001A518 000 ld         [%o0+0x3C], %g1
0001A51C 000 sethi      %hi(0x30000000), %o5
0001A520 000 bset       %o5, %g1
0001A524 000 sethi      %hi(-0x40000000), %o5
0001A528 000 bset       %o5, %g1
0001A52C 000 sethi      %hi(0x8000000), %o3
```



有效防护 价值输出

过程调用

•参数

父函数写入o0-o5，子函数使用i0-i5接受。

父函数

```
00017720 3B30 mov     %l2, %o1          ! 33bd8 这是k
00017724 3B30 add     %fp, var_620, %i3
00017728 3B30 mov     %i3, %o0          ! key1
0001772C 3B30 mov     %l2, %o1          ! key0
00017730 3B30 call    CreateKey         ! CreateKey
```

子函数

```
0001B054
0001B054 000 save     %sp, -0x80, %sp
0001B058 080 mov      0, %o4
0001B05C 080 mov      0, %o5
0001B060 080 inc      0x18, %i1         ! [3339C]+18=33BD8+18=33BF0
0001B064 080 add      %fp, var_20, %l0
0001B068 080 mov      %i0, %o0          ! key1
0001B06C 080 mov      %l0, %o1          ! 存key1尾部10字节
0001B070 080 mov      %i1, %o2          ! key0
```

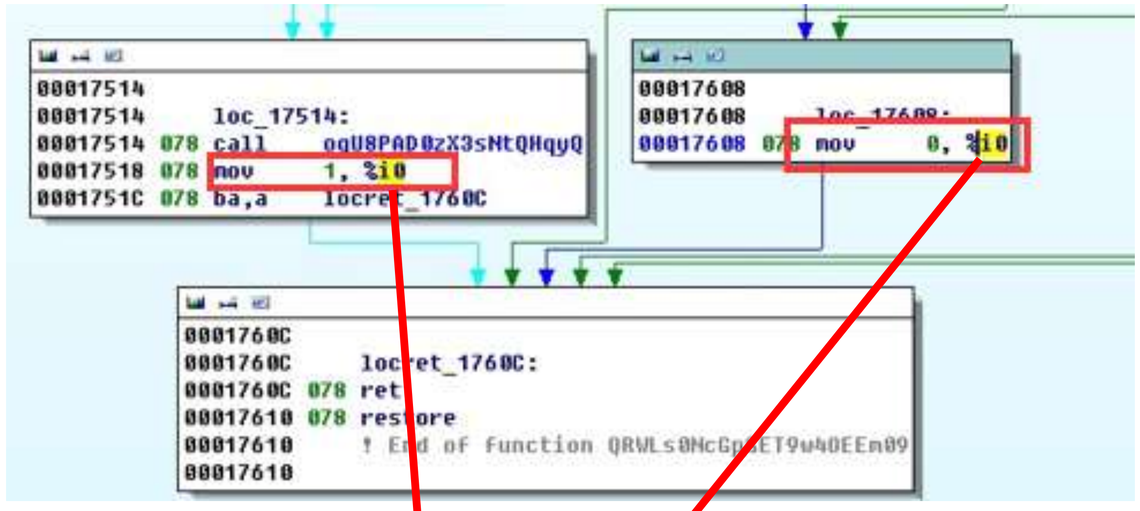
1 2

过程调用

•返回值

子函数写入i0，父函数o0接受。

子函数



父函数

```
000176BC 3B30 mov     %i0, %o0
000176C0 3B30 add     %fp, %o2, %o2  ! fp-0x3AC0
000176C4 3B30 add     %fp, %o3, %o3  ! fp-0x3AC4
000176C8 3B30 add     %fp, %o4, %o4  ! fp-0x3AC8
000176CC 3B30 call    QRWLs0NcGpGET9w4OEEm09
000176D0 3B30 mov     %i1, %o1
000176D4 3B30 cmp     %o0, 1
000176D8 3B30 mov     0, %l5
```

有效防护 价值输出

延迟槽 Delay slots

- 分支延迟槽 (Branch delay slot)，简单地说就是位于分支指令后面的一条指令，不管分支发生与否其总是被执行，而且位于分支延迟槽中的指令先于分支指令执行
  - 存在延迟槽的架构：MIPS、PA-RISC、ETRAX CRIS、SuperH、SPARC等
  - 不存在延迟槽的架构：X86、PowerPC、ARM、DEC Alpha

延迟槽 Delay slots

•指令

call

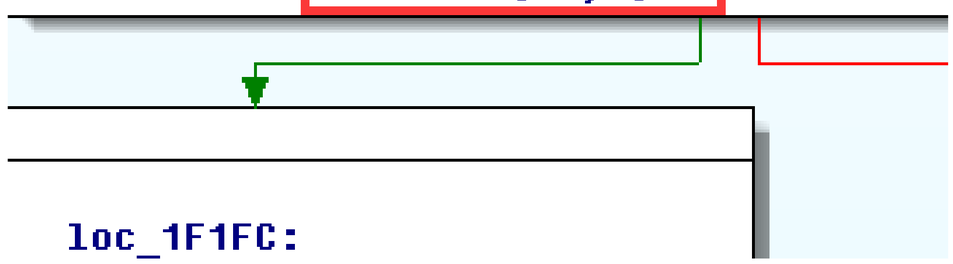
```
00017704 3B30 mov     0x200, %o1      ! size
00017708 3B30 call    _getcwd         ! 获取
0001770C 3B30 add     %i0, 0x18, %o0
```

return

```
00020560 080 inc     %l0
00020564 080 st      %l0, [%i3]
00020568 080 ret
0002056C 080 restore %g0, 1, %o0
```

jump

```
0001F27C 090 mov     %l0, %o0
0001F280 090 cmp     %o0, 0
0001F284 090 be      loc_1F1FC
0001F288 090 mov     %i3, %o1
```



延迟槽 Delay slots

nop

```
00018058
00018058          loc_18058:
00018058 078 orcc     %g1, %o3, %g0
0001805C 078 bne      locret_180B4
00018060 078 nop
```

, a

```
0001AA1C 070 nop
0001AA20 070 orcc     %o0, 0, %i0
0001AA24 070 be,a     locret_1AA2C
0001AA28 070 st       %l0, [%i1]        ! 344D8
0001AA2C
0001AA2C          locret_1AA2C:
0001AA2C 070 ret
```

有效防护 价值输出

过程调用

call： 将自身地址写入o7

父函数

```
000176A8 3B30 std     %o0, [%fp+%o2] ! clr -0x3AC0
000176AC 3B30 clr     [%fp+%o4]      ! clr -0x3AC8
000176B0 3B30 sethi   %hi(loc_1BC00), %l7
000176B4 3B30 call    sub_17428
000176B8 3B30 set     loc_1BCB8, %l7 ! l7=3336C
```

子函数

```
00017428
00017428
00017428          sub_17428:
00017428 000 retl
0001742C 000 add      %o7, %l7, %l7
0001742C          ! End of function sub_17428
```

$l7=176B4+1BCB8=3336C$

汇编指令

- neg reg = sub %g0,reg, reg
- not reg = xnor reg,%g0,reg
- btst bits, reg = andcc reg, bits, %g0
- bset bits, reg = or reg, bits, reg
- bclr bits, reg = andn reg, bits, reg
- btog bits, reg = xor reg, bits, reg

解密函数


```
00022C94      .global decode_str_sll
00022C94      decode_str_sll:
00022C94 000 save     %sp, -0x70, %sp ! _9moHttzb2
00022C98 070 call     _strlen
00022C9C 070 mov      %i0, %o0          ! size
00022CA0 070 mov      %o0, %l0
00022CA4 070 call     _malloc
00022CA8 070 inc      %o0
00022CAC 070 cmp      %o0, 0
00022CB0 070 be       loc_22CE4
00022CB4 070 mov      0, %o4
```

密文

长度

```
00022CE4
00022CE4      loc_22CE4:
00022CE4 070 clrb     [%o0+%l0]
```

```
00022CB8
00022CB8      loc_22CB8:                ! l0=密文长度
00022CB8 070 cmp      %o4, %l0
00022CBC 070 bge,a    locret_22CE8
```

```
00022CC0 070 clrb     [%o0+%l0]         ! 清空数据尾
00022CC4 070 ldsb     [%i0+%o4], %o5    ! 逐字节取密文
00022CC8 070 sll      %o5, 3, %g1       ! *8
00022CCC 070 sub      %g1, %o5, %g1     ! -1
00022CD0 070 sll      %g1, 2, %g1       ! *3
00022CD4 070 add      %g1, %o5, %g1     ! 解密操作
00022CD8 070 stb      %g1, [%o0+%o4]
00022CDC 070 ba       loc_22CB8         ! l0=密文长度
00022CE0 070 inc      %o4
```

```
00022CE8
00022CE8      locret_22CE8:
00022CE8 070 ret
00022CEC 070 restore %g0, %o0, %o0
00022CEC      ! End of function decode_str_sll
00022CEC
```

(byte * 8 - byte) * 4 + byte = byte * 29

解密数据地址

- 配置解密

```
870 sethi   %hi(0), %g1
870 sethi   %hi(0x14000), %l7
870 call    sub_1E7E4
870 set     0x14230, %l7
870 set     0x174, %g1
870 call    decode_str_sll  !
870 ld      [%l7+%g1], %o0
870 cmp     %g0, %i0
870 subc    %g0, -1, %o5
```

```
rodata:00023118     unk_23118:      .byte 0xBB !
rodata:00023119                     .byte 0x86 !
rodata:0002311A                     .byte 0x91 !
rodata:0002311B                     .byte 0xFB !
rodata:0002311C                     .byte 0x42 ! B
rodata:0002311D                     .byte 0xBD !
rodata:0002311E                     .byte 0x5C ! \
rodata:0002311F                     .byte 0x5C ! \
rodata:00023120                     .byte 0x15
rodata:00023121                     .byte 0xBB !
rodata:00023122                     .byte 0x1E
rodata:00023123                     .byte 0xBD !
rodata:00023124                     .byte 0x9A !
rodata:00023125                     .byte 0xE9 !
rodata:00023126                     .byte 0x1E
rodata:00023127                     .byte 0xFB !
rodata:00023128                     .byte 0xD8 !
rodata:00023129                     .byte 0xBB !
```

- 参数o0，~~0x14230+0x174=0x143A4~~

```
    sub_1E7E4:
000 retl
000 add     %o7, %l7, %l7
```

- l7=0x14230+0x1F13C=0x3336C
- o0=[0x3336C+0x174]
- =[0x334E0]

```
.got:000334E0
.got:000334E4
.got:000334E8
.got:000334EC
.got:000334F0
```

```
.word unk_23118
.word unk_34650
.word unk_34850
.word unk_34854
.word unk_23130
```

密文和明文

- 0xBB*0x1D(29) = 0x152F 2F = ‘/’
- 0x86*0x1D = 0xF2E 2E = ‘.’
- /.mozilla/firefox/

```
rodata:00023118     unk_23118:      .byte 0xBB !
rodata:00023119                     .byte 0x86 !
rodata:0002311A                     .byte 0x91 !
rodata:0002311B                     .byte 0xFB !
rodata:0002311C                     .byte 0x42 ! B
rodata:0002311D                     .byte 0xBD !
rodata:0002311E                     .byte 0x5C ! \
rodata:0002311F                     .byte 0x5C ! \
rodata:00023120                     .byte 0x15
rodata:00023121                     .byte 0xBB !
rodata:00023122                     .byte 0x1E
rodata:00023123                     .byte 0xBD !
rodata:00023124                     .byte 0x9A !
rodata:00023125                     .byte 0xE9 !
rodata:00023126                     .byte 0x1E
rodata:00023127                     .byte 0xFB !
rodata:00023128                     .byte 0xD8 !
rodata:00023129                     .byte 0xBB !
```

```
Text
        call    decode_str_sll  ! /bin/false
        call    decode_str_sll  ! /sbin/nologin
-48     call    decode_str_sll  ! '\r\n\r\n'
...     call    decode_str_sll  ! 100  ' 200 Connection established'
        call    decode_str_sll  ! 'user_pref("'
        call    decode_str_sll  ! 'network.proxy.http'
        call    decode_str_sll  ! 'network.proxy.http_port'
        call    decode_str_sll  !  'network.proxy.ssl'
        call    decode_str_sll  !  'network.proxy.ssl_port'
l...    call    decode_str_sll  ! '/.netscape'
l...    call    decode_str_sll  ! '/preferences.js'
        call    decode_str_sll  ! '/.mozilla/'
        call    decode_str_sll  !  '/.mozilla/firefox/'
y...    call    decode_str_sll  ! '/bin/false'
y...    call    decode_str_sll  ! '/sbin/nologin'
```

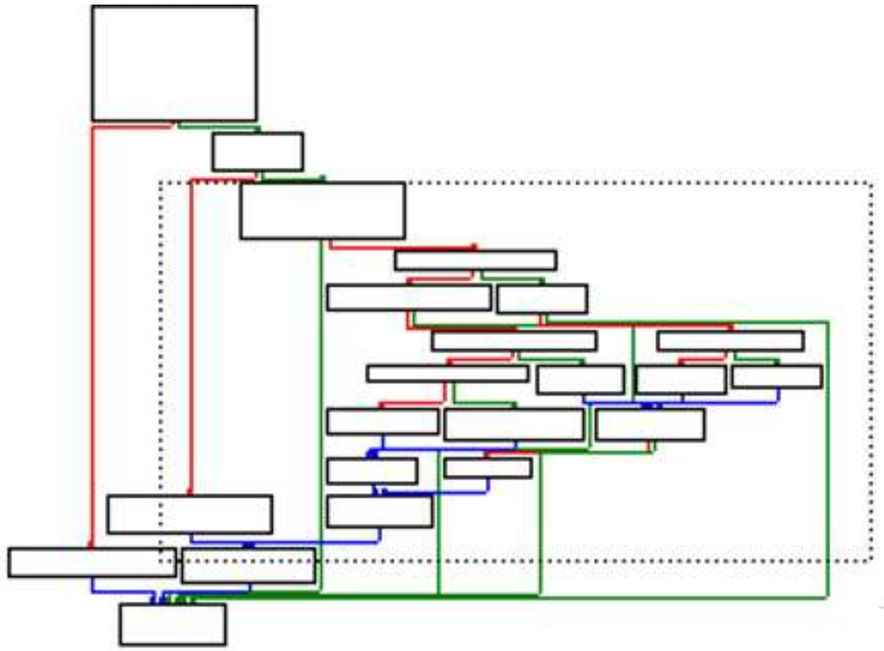
有效防护 价值输出

控制指令

- 数据包第一字节

```
0001CF84 070 mov     %i0, %o0
0001CF88 070 ldub    [%i0+0xF4], %g1 ! buffer第一字节
0001CF8C 070 cmp     %g1, 0x60       ! 判断是否是60指令
0001CF90 070 be      end
0001CF94 070 mov     7, %o5
```

- 判断是哪条指令，根据指令返回值跳转不同功能函数



```
0001CFA0 070 cmp      %g1, 0x4A
0001CFA4 070 be       end
0001CFA8 070 mov      0xF, %o5
```

```
0001D024
0001D024     end:
0001D024 070 ret
0001D028 070 restore %g0, %o5, %o0
```

```
00017CA4 3B30 ba      loc_17AC0
00017CA8 3B30 mov     %o0, %15
00017AC0     loc_17AC0:
00017AC0 3B30 cmp     %15, 0x1F
00017AC4 3B30 bgu,a   loc_17ACC
```

```
00017AC8 3B30 mov     0xC, %15
```

```
00017EC8 0xFFFFFC20
00017ECC 0xFFFFFEC4
00017ED0 0xFFFFFFF0
00017ED4 0xFFFFFE8C
00017ED8 0xFFFFFEA4
00017EDC 0xFFFFFFF0
00017EE0 0xFFFFFDC8
00017EE4 0xFFFFFDE4
00017EE8 0xFFFFFE54
00017EEC 0xFFFFFE00
00017EF0 0xFFFFFEE0
00017EF4 0xFFFFFFA4
00017EF8 0xFFFFFF7C
00017EFC 0xFFFFFF60
00017F00 0xFFFFFFF0
00017F04 0xFFFFFE70
00017F08 0xFFFFFFF0
00017F0C 0xFFFFFFF0
00017F10 0xFFFFFFF0
00017F14 0xFFFFFFF0
00017F18 0xFFFFFDAC
00017F1C 0xFFFFFD00
00017F20 0xFFFFFD50
00017F24 0xFFFFFCB8
00017F28 0xFFFFFF50
00017F2C 0xFFFFFFF0
00017F30 0xFFFFFFF0
00017F34 0xFFFFFFF0
00017F38 0xFFFFFFF0
00017F3C 0xFFFFFF50
00017F40 0xFFFFFE1C
00017F44 0xFFFFFE38
```

```
    loc_17AD0:
3B30 set      loc_1B4A4, %g1
3B30 sub      %17, %g1, %o5
3B30 sll      %15, 2, %g1
3B30 ld       [%o5+%g11], %g1
3B30 jmp      %g1+%o5
```

```
! 3336C-1B4A4=17EC8
! 15=
! 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
! 0, 4, 8, C, 10,14,18,1C,20,24
!
! A, B, C, D, E, F, 10,11,12,13
! 28,2C,30,34,38,3C,40,44,48,4C
!
! 14,15,16,17,18,19,1A,1B,1C,1D
! 50,54,58,5C,60,64,68,6C,70,74
!
! 1E,1F
! 78,7C
```

```
! 17AE8,17D8C,17EB8,17D54,17D6C,17EB8,17C90,17CAC,17D1C,
! 17DA8,17E6C,17E44,17E28,17EB8,17D38,17EB8,17EB8,17EB8,
! 17C74,17BC8,17C18,17B80,17E18,17EB8,17EB8,17EB8,17EB8,
! 17CE4,17D00
```

创建文件并执行


```
00018FB0 8B8 ld      [%fp+var_1C], %o1
00018FB4 8B8 add     %l1, %o1, %o1   ! s2
00018FB8 8B8 call    _memcpy
00018FBC 8B8 ld      [%fp+var_18], %o2
00018FC0 8B8 ld      [%fp+var_24], %o1
00018FC4 8B8 add     %l1, %o1, %o1   ! s2
00018FC8 8B8 ld      [%fp+n], %o2    ! n
00018FCC 8B8 call    _memcpy
00018FD0 8B8 mov     %l4, %o0
00018FD4 8B8 add     %fp, -0x844, %o1 ! file handle
00018FD8 8B8 add     %fp, var_848, %o2
00018FDC 8B8 mov     %l3, %o0        ! filename
00018FE0 8B8 call    CreateFile1
00018FE4 8B8 add     %fp, var_84C, %o3 ! int
00018FE8 8B8 cmp     %o0, 1          ! 0成功
00018FEC 8B8 mov     0, %o2          ! int
00018FF0 8B8 mov     3, %o1
00018FF4 8B8 bne     loc_1900C
00018FF8 8B8 mov     %l2, %o0        ! int
```

创建文件

1

```
call    _fopen
ld      [%l7+%g1], %o1  ! 333A8 a+
mov     %o0, %l1
mov     2, %o2          ! whence
mov     0, %o1          ! offset
cmp     %o0, 0
be      locret_18774
mov     1. %o5
```

```
save    %sp, -0x70, %sp
add     %i0, 1, %o0     ! ptr
mov     1, %o1          ! size
mov     %i1, %o2        ! nitems
call    _fwrite         ! only write
mov     %i2, %o3        ! file
cmp     %o0, %i1
mov     %i2, %o0        ! stream
bl      locret_187AC
mov     1, %i0
```

```
0001900C
0001900C     loc_1900C:             ! int
0001900C 8B8 ld      [%fp+var_84C], %o1
00019010 8B8 call    return_createfile ! 创建成功将结果回传
00019014 8B8 mov     %l2, %o0       ! key1
00019018 8B8 cmp     %o0, 1         ! 0成功
0001901C 8B8 mov     %l3, %o4
00019020 8B8 mov     %l4, %o5       ! filename2
00019024 8B8 bne     loc_19044
00019028 8B8 mov     %l2, %o0       ! stream
```

回传结果

2

```
0001902C 8B8 call    _fclose
00019030 8B8 ld      [%fp+filename1], %o0 ! p
00019034 8B8 call    _unlink
00019038 8B8 mov     %l3, %o0
0001903C 8B8 ba      locret_19060
00019040 8B8 mov     0x1D, %i0
```

```
00019044
00019044     loc_19044:
00019044 8B8 ldub    [%fp+var_37], %g1
00019048 8B8 st      %g1, [%sp+0x8B8+var_85C]
0001904C 8B8 ld      [%fp+var_848], %o1
00019050 8B8 ld      [%fp+var_84C], %o2
00019054 8B8 call    execute_file_0  ! 写入数据后执行文件
00019058 8B8 ld      [%fp+filename1], %o3 ! filename1
0001905C 8B8 mov     %o0, %i0
```

写入文件并执行

3

```
loc_18978:
mov     %l0, %o3
call    _execle
mov     0, %o2
```

有效防护

数据包结构-文件名


```
00018F90 8B8 mov     %l1, %o1          ! recv_buffer
00018F94 8B8 mov     %l0, %o2          ! n
00018F98 8B8 add     %fp, var_840, %o4 ! int
00018F9C 8B8 call    analytic_recvdata ! encode filename
00018FA0 8B8 add     %fp, var_83C, %o3
00018FA4 8B8 cmp     %o0, 1
00018FA8 8B8 be      locret_19060
00018FAC 8B8 mov     %l3, %o0          ! s1
00018FB0 8B8 ld      [%fp+var_1C], %o1
00018FB4 8B8 add     %l1, %o1, %o1     ! s2
00018FB8 8B8 call    _memcpy
00018FBC 8B8 ld      [%fp+var_18], %o2
00018FC0 8B8 ld      [%fp+var_24], %o1
00018FC4 8B8 add     %l1, %o1, %o1     ! s2
00018FC8 8B8 ld      [%fp+n], %o2      ! n
00018FCC 8B8 call    _memcpy
00018FD0 8B8 mov     %l4, %o0
00018FD4 8B8 add     %fp, -0x844, %o1 ! file handle
00018FD8 8B8 add     %fp, var_848, %o2
00018FDC 8B8 mov     %l3, %o0          ! filename
00018FE0 8B8 call    CreateFile1
00018FE4 8B8 add     %fp, var_84C, %o3 ! int
00018FE8 8B8 cmp     %o0, 1            ! 0成功
00018FEC 8B8 mov     0, %o2            ! int
00018FF0 8B8 mov     3, %o1
00018FF4 8B8 bne     loc_1900C
00018FF8 8B8 mov     %l2, %o0          ! int
```

o1是文件名

o2是文件名长度

```
add     %i1, 0x12, %o1  ! s2
mov     4, %o2          ! n
call    _memcpy
add     %i0, 0x1C, %o0  ! s1 i0=-38
```

1

2

i0=-0x38　i1=buffer

-0x38+0x1C = -0x1C

数据包偏移0x12是文件名偏移

3

```
add     %i1, 0x16, %o1  ! s2
mov     4, %o2          ! n
call    _memcpy
add     %i0, 0x20, %o0
```

4

-0x38+0x20=-0x18

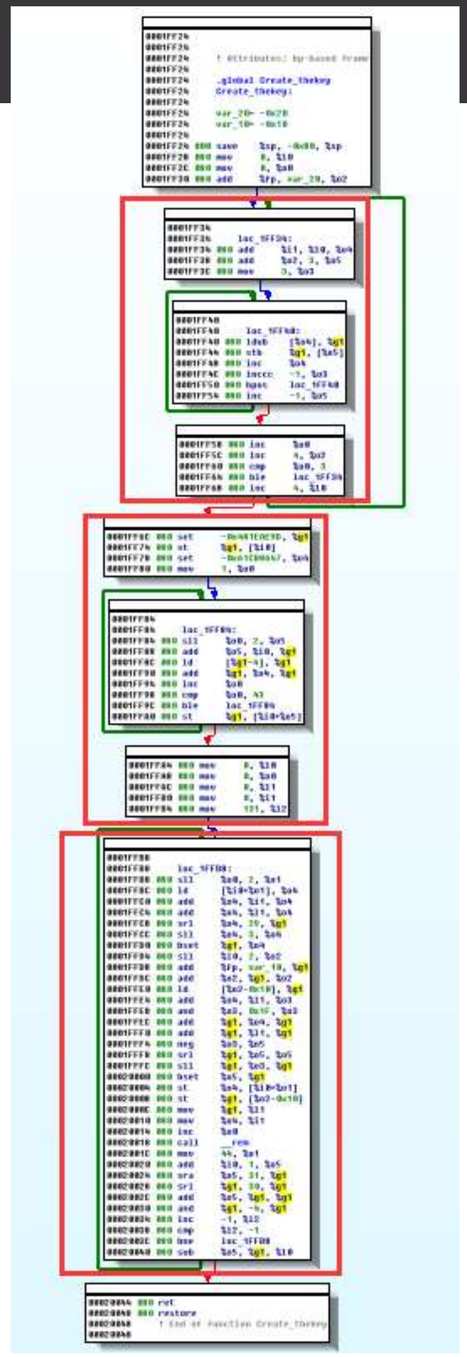
偏移0x16对应文件名的长度

数据包格式

- 创建文件
  - 偏移0x12的值是文件名的偏移，偏移0x16指向的值是文件名的长度
- 写入文件
  - 偏移0x1指向的数据是要写入的数据
- 执行文件
  - 偏移0xA指向的值是文件名，偏移0xE指向的值是文件名长度

通信 – 密钥生成

- 密钥生成函数
  - **一个参数key0**
  - **三部分代码**
    - 取key0数据
    - 生成随机种子
    - key0与种子运算生成密钥



key0推导


```
000176B4 3B30 call     sub_17428
000176B8 3B30 set      loc_1BCB8, %17
```

0x1BCB8+0x176B4 = 0x3336C

```
3B30 set      0x30, %g1
3B30 add      %fp, var_10, var10
3B30 or       %l4, 0x3A0, %o5 ! l4=c80
3B30 or       %i5, 0x158, %l3
3B30 ld       [%17+%g1], %l2 ! [3339C]
```

0x3336C+0x30 = 0x3339C

```
inc     0x18, %i1       ! [3339C]+18=33BD8+18=33BF0
```

[0x3339C] = 0x33BD8

0x33BD8+0x18 = 0x33BF0

```
:00033BF0          .byte 0x66 ! f
:00033BF1          .byte 0x39 ! 9
:00033BF2          .byte 0x71 ! q
:00033BF3          .byte 0x3C ! <
:00033BF4          .byte  0xF
:00033BF5          .byte 0x85 !
:00033BF6          .byte 0x99 !
:00033BF7          .byte 0x81 !
:00033BF8          .byte 0x20
:00033BF9          .byte 0x19
:00033BFA          .byte 0x35 ! 5
:00033BFB          .byte 0x43 ! C
:00033BFC          .byte 0xFE !
:00033BFD          .byte 0x9A !
:00033BFE          .byte 0x84 !
:00033BFF          .byte 0x11
```

0x6639713C,0x0F859981,
0x20193543,0xFE9A8411

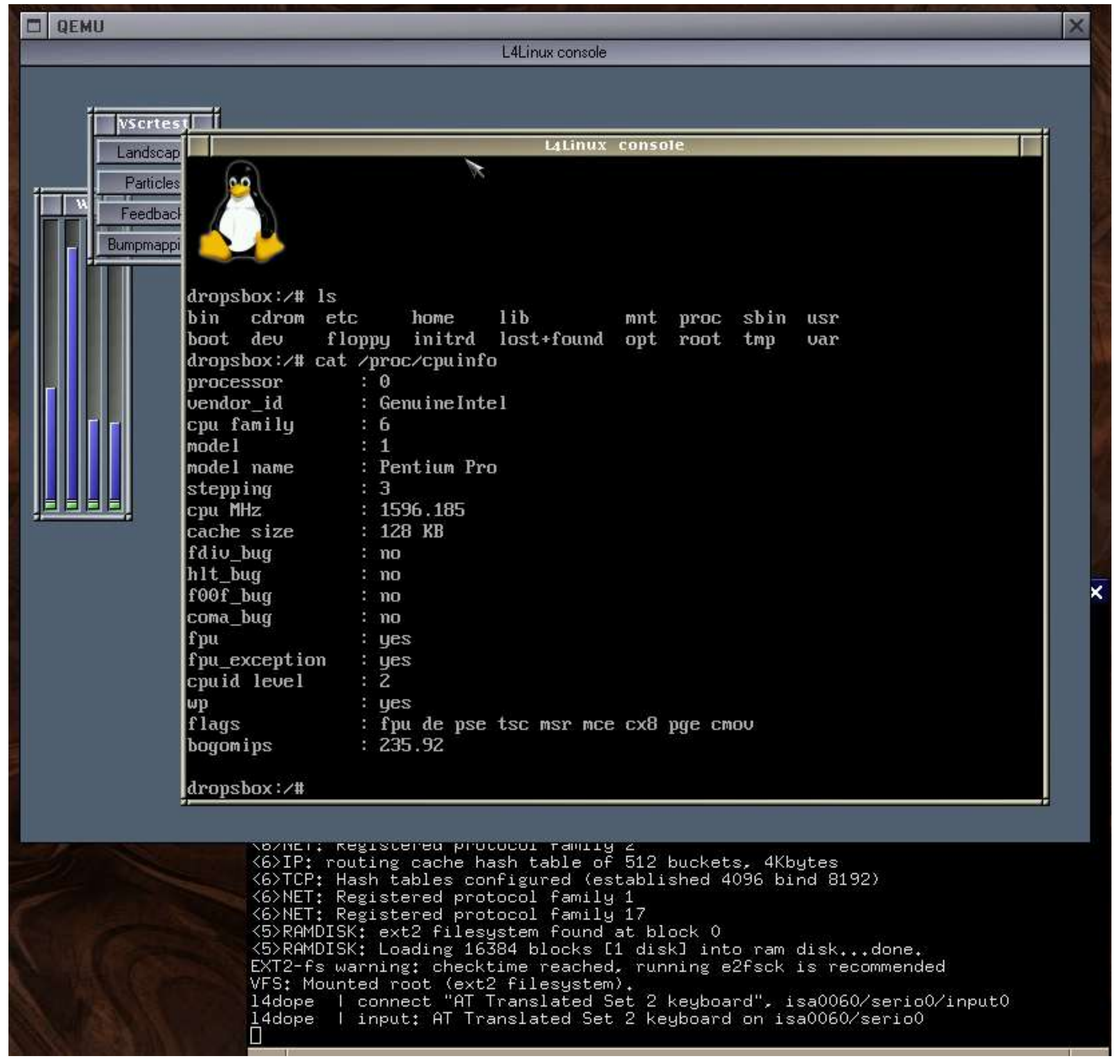
有效防护 价值输出

动态调试技术

- 虚拟机
- 调试器

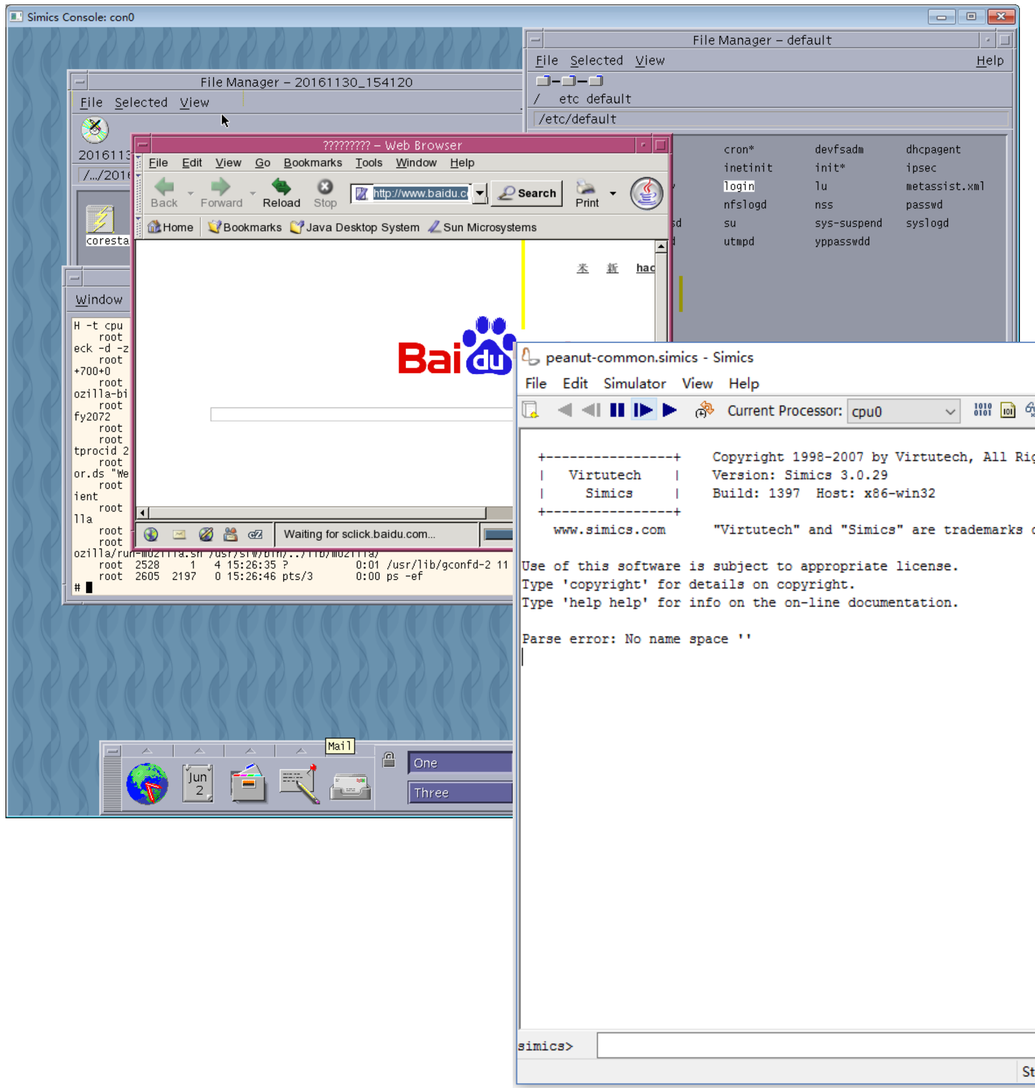
QEMU

- QEMU是一套由Fabrice Bellard所编写的模拟处理器的自由软件。它与Bochs，PearPC近似，但其具有某些后两者所不具备的特性，如高速度及跨平台的特性。经由kqemu这个开源的加速器，QEMU能模拟至接近真实电脑的速度。
- **优点：**
- 可以模拟 IA-32 (x86)个人电脑，AMD64个人电脑， MIPS R4000, 升阳的 SPARC sun3 与 PowerPC (PReP 及 Power Macintosh)架构
- 支持其他架构，不论在主机或虚拟系统上增加了模拟速度，某些程式甚至可以实时运行
- 可以在其他平台上运行Linux的程序
- 可以储存及还原运行状态(如运行中的程序)
- 可以虚拟网络卡
- 可模拟多CPU



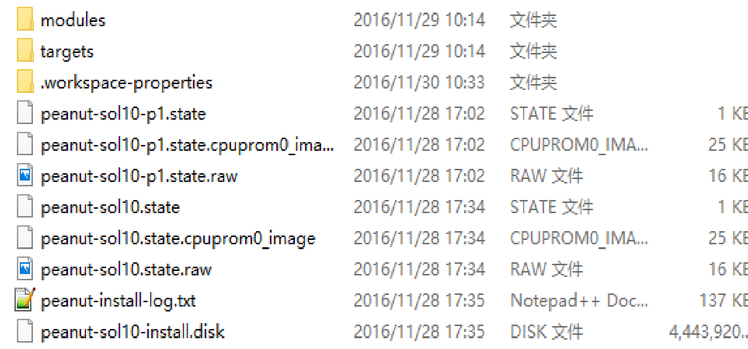
Virtutech Simics

- Simics 是一种完整系统模拟技术，为软件和系统开发人员、架构师、测试工程师提供为各种目的构建和使用虚拟系统或创建多个虚拟连接系统的方法。
- Simics提供了功能丰富的接口和数据结构，使得研究者可以方便定义设备模块。
- Simics4.0实现了可逆执行及调试功能，使得程序不用重启就可以恢复到以前的执行状态。



安装Solaris系统

1. peanut-common.simics
   - $memory_megs = 1024
2. peanut-sol10-cd-install1.simics
   - $cdrom_path = "D:\\sol-10-u2-ga-sparc-v1.iso “
   - 安装很慢...
   - sol-10-u2-ga-sparc-v1.iso 改为 sol-10-u2-ga-sparc-v3.iso



3. new session->workspace\targets\sunfire，完成后续安装
   - peanut-sol9-cd-install3.simics
   - peanut-common.simics 增加一行：$os = "solaris10"
4. 启动安装好的系统
   - new session 选workspace\targets\sunfire\peanut-common.simics

- connect-real-network IP（hostIP）

```
  +----------------+    Copyright 1998-2007 by Virtutech, All Rights Reserved
  |   Virtutech    |    Version: Simics 3.0.29
  |     Simics     |    Build: 1397  Host: x86-win32
  +----------------+
    www.simics.com      "Virtutech" and "Simics" are trademarks of Virtutech AB

Use of this software is subject to appropriate license.
Type 'copyright' for details on copyright.
Type 'help help' for info on the on-line documentation.

simics> connect-real-network 10.10.0.5
NAPT enabled with gateway 10.10.0.1 on link link0.
Host TCP port 4021 -> 10.10.0.5:21 on link link0
Host TCP port 4023 -> 10.10.0.5:23 on link link0
Host TCP port 4080 -> 10.10.0.5:80 on link link0
Real DNS enabled at 10.10.0.1 on link link0.
simics> c
simics> stop
[cpu0] v:0x000000000105d8d4 p:0x0003905d8d4  lduw [%o0 + 36], %g5
simics> enable-real-time-mode 80
simics> c
simics> c
Simics is already running.
simics> stop
[cpu0] v:0x000000000105d8f4 p:0x0003905d8f4  lduw [%i0 + 0], %i1
simics> enable-real-time-mode 90
simics> c
simics> c
Simics is already running.
simics> enable-real-time-mode 100
simics> c
```

peanut-common.simics

```
script-branch {
    wait-for-variable machine_defined
    $pcibrd = (create-sunfire-pci-board
mac_address = "10:10:10:10:10:14")
    $pgx64 = (create-sun-pci-pgx64)
    $gfxcon = (create-std-graphics-console)
    $keyboard = (create-sun-type5-keyboard)
    $mouse = (create-sun-type5-mouse)
    $scsi_bus1 = (create-std-scsi-bus)

    $system.connect slot2 $pcibrd
    $pcibrd.connect pci-slot0 $pgx64
    $pcibrd.connect $scsi_bus1
    $system.connect keyboard $keyboard
    $system.connect mouse $mouse
    $pgx64.connect console $gfxcon
    $gfxcon.connect keyboard $keyboard
    $gfxcon.connect mouse $mouse
}
```

peanut-setup.include

```
if $os != none {
    #load-persistent-state prefix = (get-component-prefix) $state
    # user override
    $system.set-nvram-hostid $hostid
    $system.set-nvram-mac $mac_address
}
```

sunfire-6500-system.include

```
#$console = (create-std-text-console)
#$system.connect ttya $console
```

启动系统

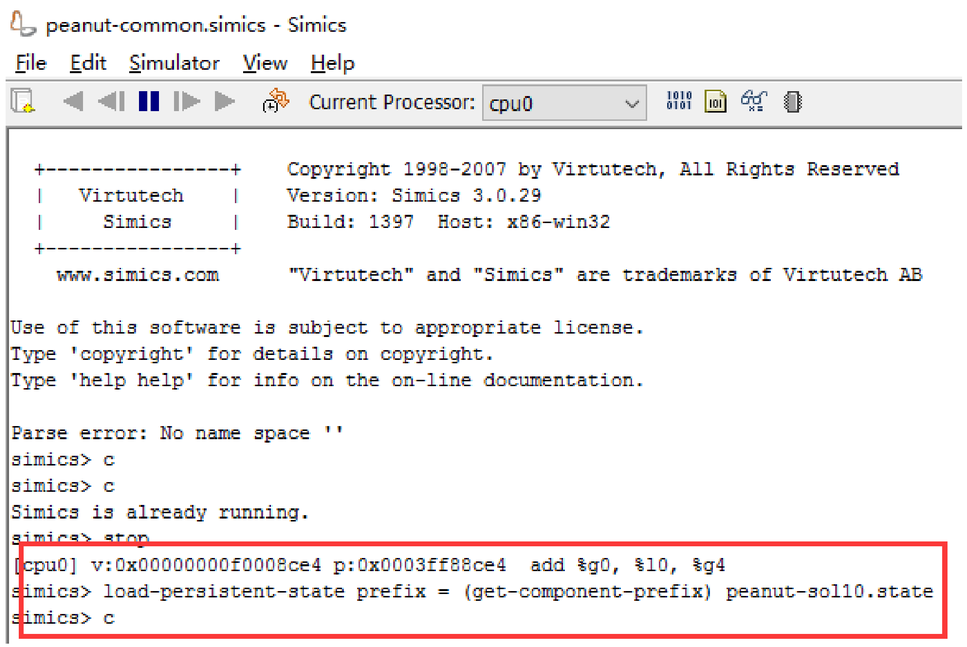
- 新建会话启动 peanut-common.simics，在simics界面输入c命令
- 出现报错”Bad magic number in disk label”

```
Boot device: /sbus@3,0/SUNW,fas@3,8800000/sd@1,0  File and args: -v
Bad magic number in disk label
Can't open disk label package
Evaluating: boot disk1 -v

Can't open boot device

ok boot disk1 -rv
```

1.在simics中，输入如下命令

stop

load-persistent-state prefix = (get-component-prefix) peanut-sol10.state



2.在虚拟机中输入下命令启动

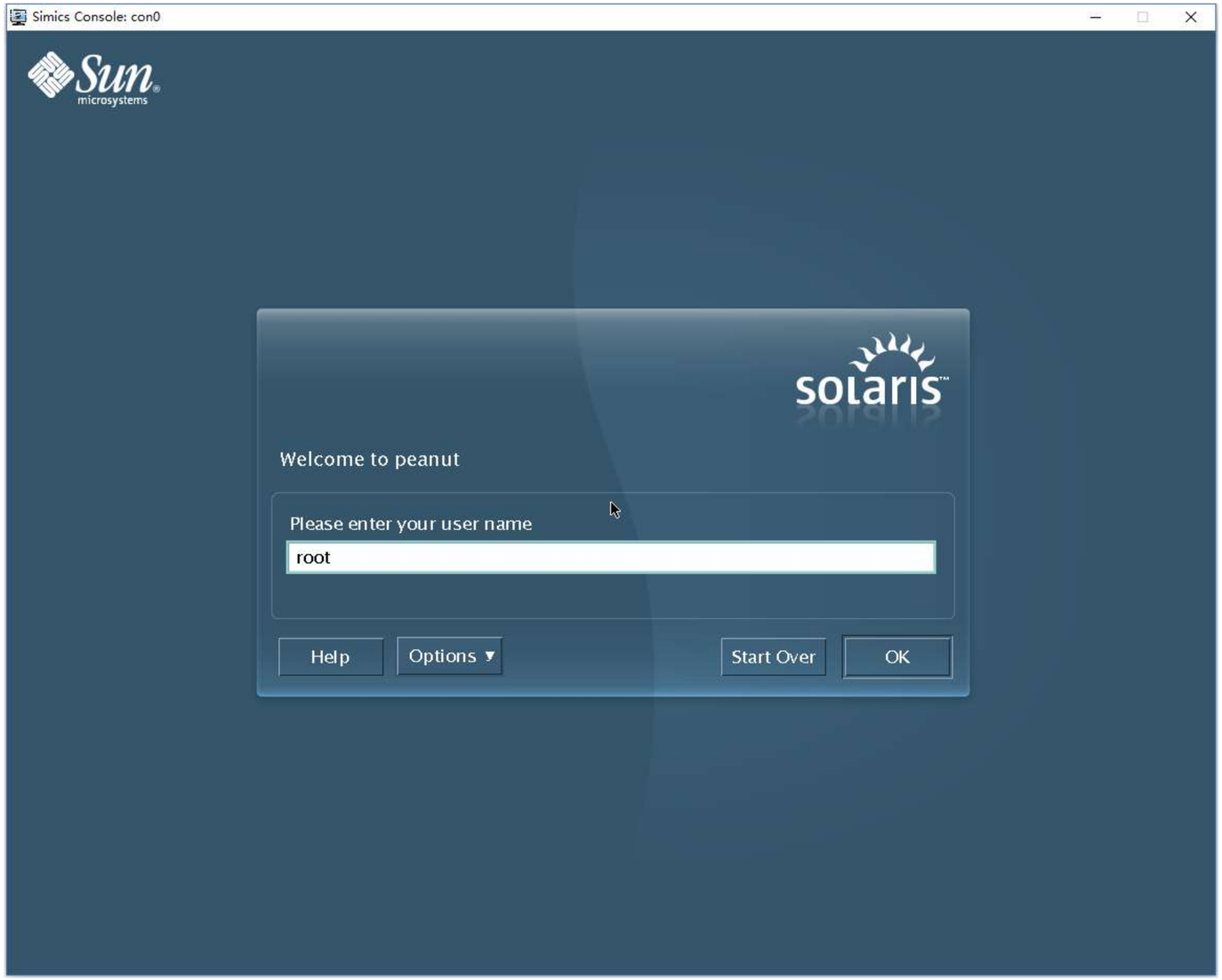
boot disk1 –rv

```
ok boot disk1 -rv
Boot device: /sbus@3,0/SUNW,fas@3,8800000/sd@1,0  File and args: -rv
ufs-file-system
Loading: /platform/SUNW,Ultra-Enterprise/boot_archive
Loading: /platform/sun4u/boot_archive
ramdisk-root ufs-file-system
Loading: /platform/SUNW,Ultra-Enterprise/kernel/sparcv9/unix
Loading: /platform/sun4u/kernel/sparcv9/unix
module /platform/sun4u/kernel/sparcv9/unix: text at [0x1000000, 0x10a358d] data
at 0x1800000
module /platform/sun4u/kernel/sparcv9/genunix: text at [0x10a3590, 0x126b757] da
ta at 0x1866800
module /platform/SUNW,Ultra-Enterprise/kernel/misc/sparcv9/platmod: text at [0x1
26b758, 0x126bd77] data at 0x18bc088
module /platform/sun4u/kernel/cpu/sparcv9/SUNW,UltraSPARC-II: text at [0x126bd80
, 0x1278187] data at 0x18bc880
SunOS Release 5.10 Version Generic_141444-09 64-bit
Copyright 1983-2009 Sun Microsystems, Inc.  All rights reserved.
Use is subject to license terms.
os-io
```

3.在simics中输入命令继续运行

c

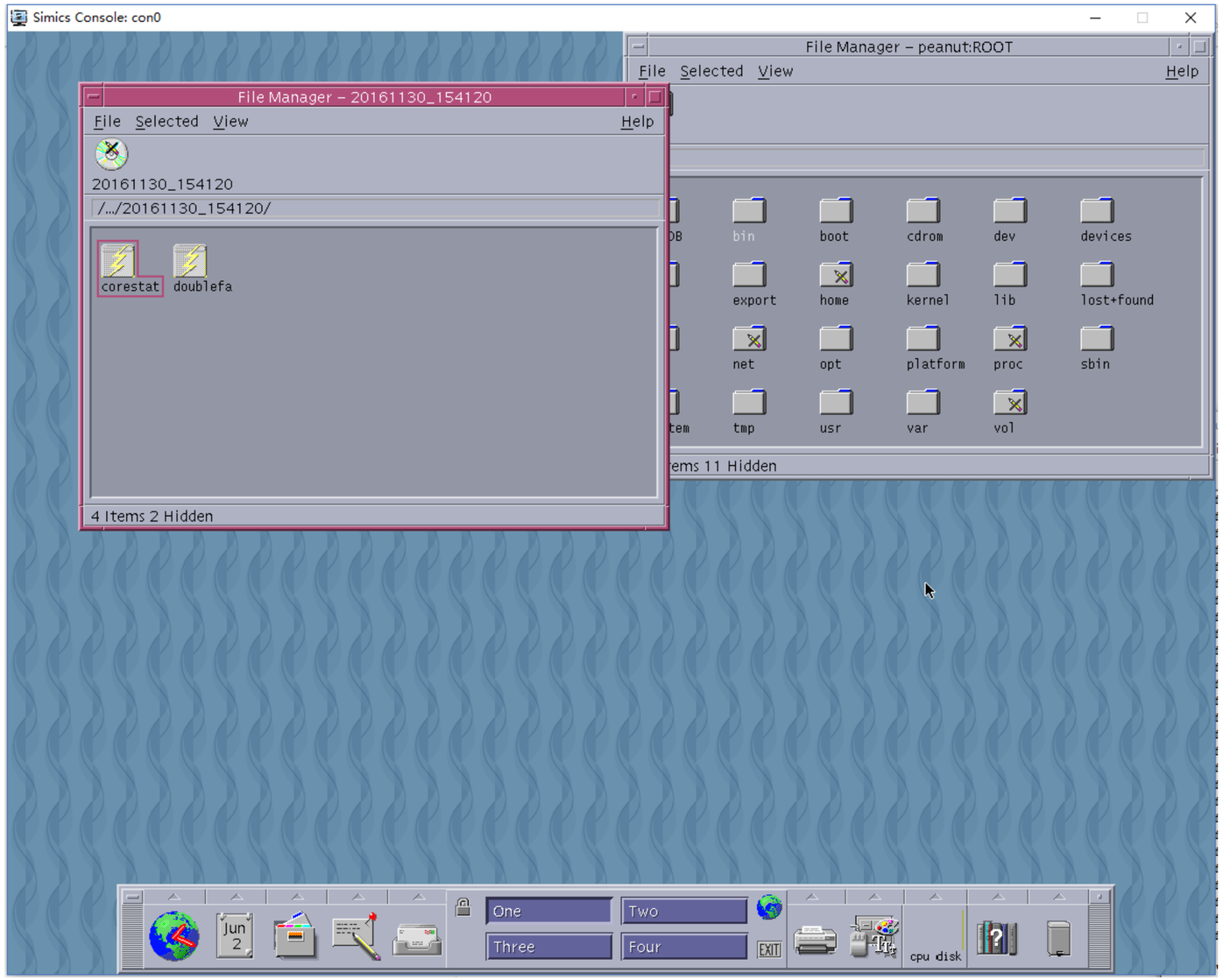
登录系统-选择CDE


Simics Console: con0
Sun microsystems
solaris
Welcome to peanut
Please enter your user name
root
Help
Options
Start Over
OK

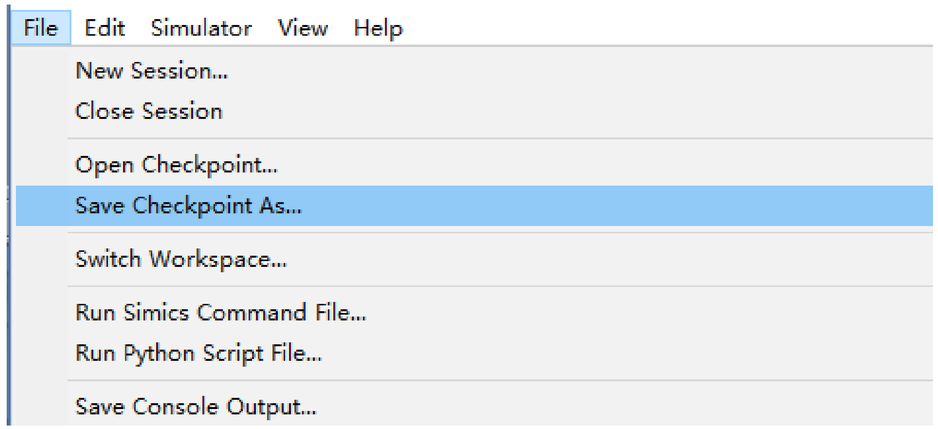

Welcome to Solaris. The following desktops are available. Which one would you like to use as your default desktop?
Note: You can change this default at any time by using the Session Menu located on the Desktop Login Screen's Options Menu.
Choose One:
Java Desktop System, Release 3
Common Desktop Environment (CDE)
OK
Cancel

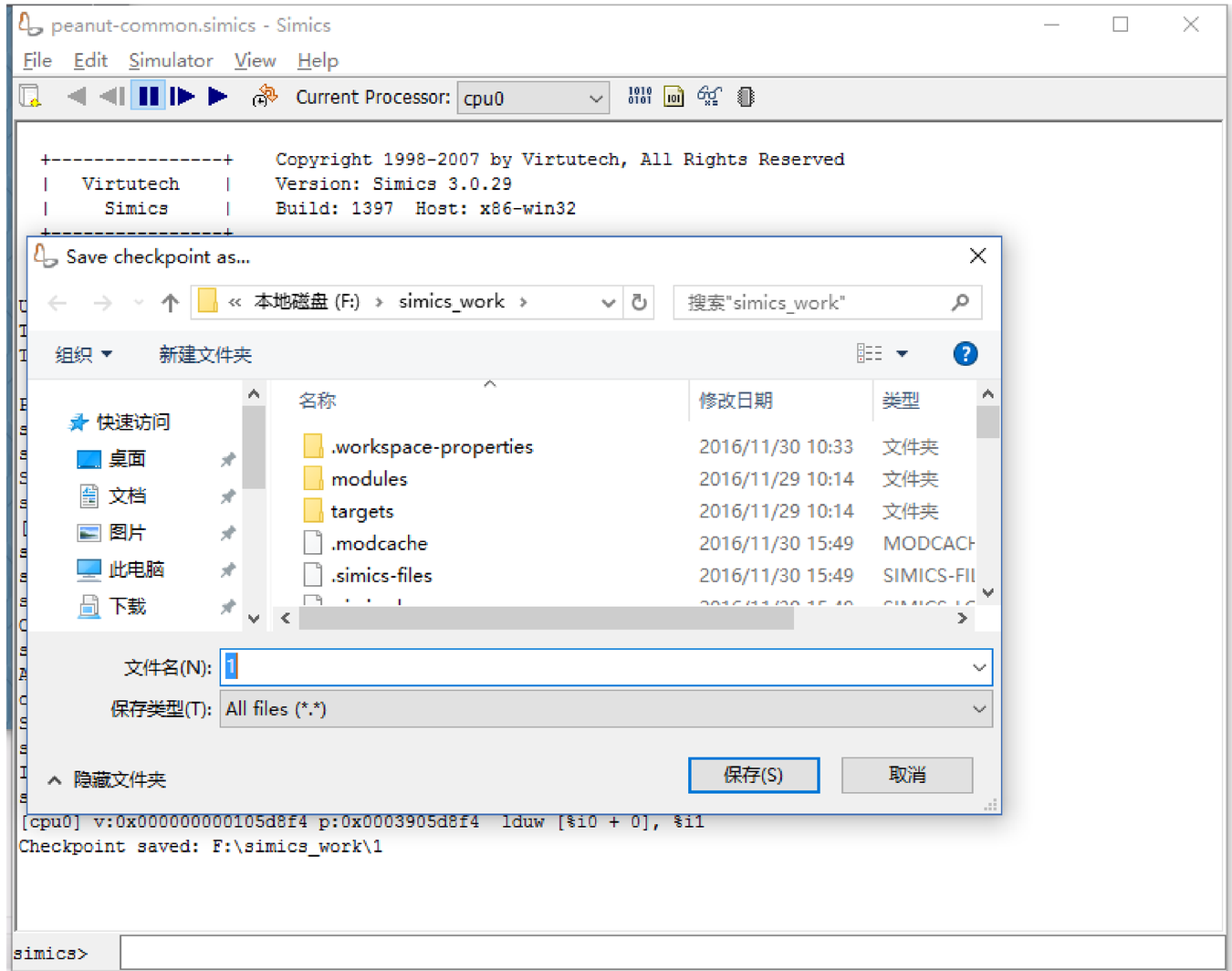
主机和虚拟机复制文件

1. 将要传输的文件制作为iso镜像
2. 将iso放到工作目录下
3. new-file-cdrom xxx.iso
4. cd0.insert xxx
5. ls /cdrom/cdrom0
6. Eject弹出，传输其他文件



- stop
- File->Save Checkpoint as

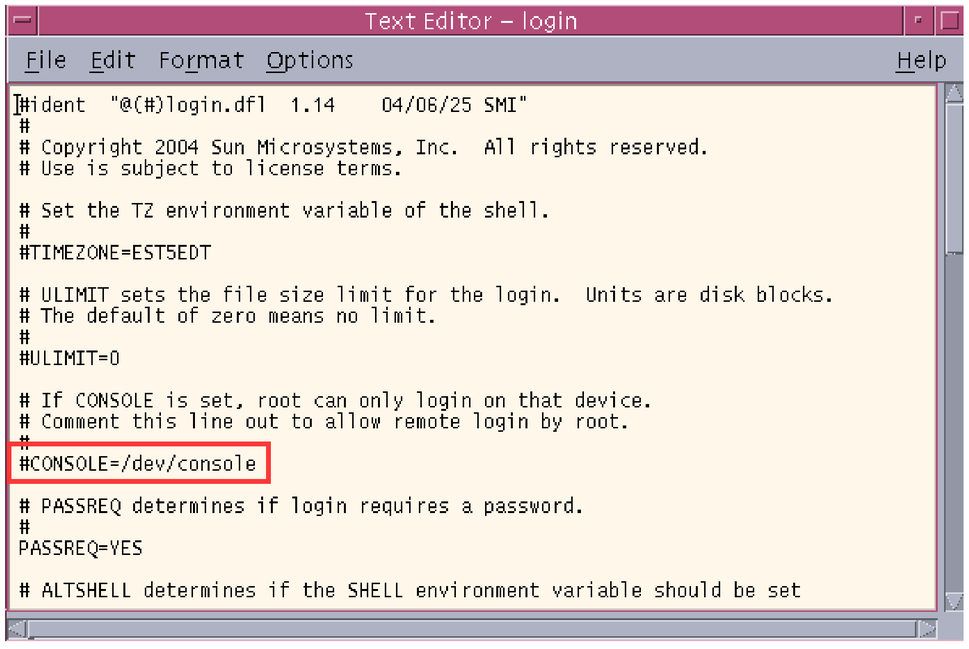


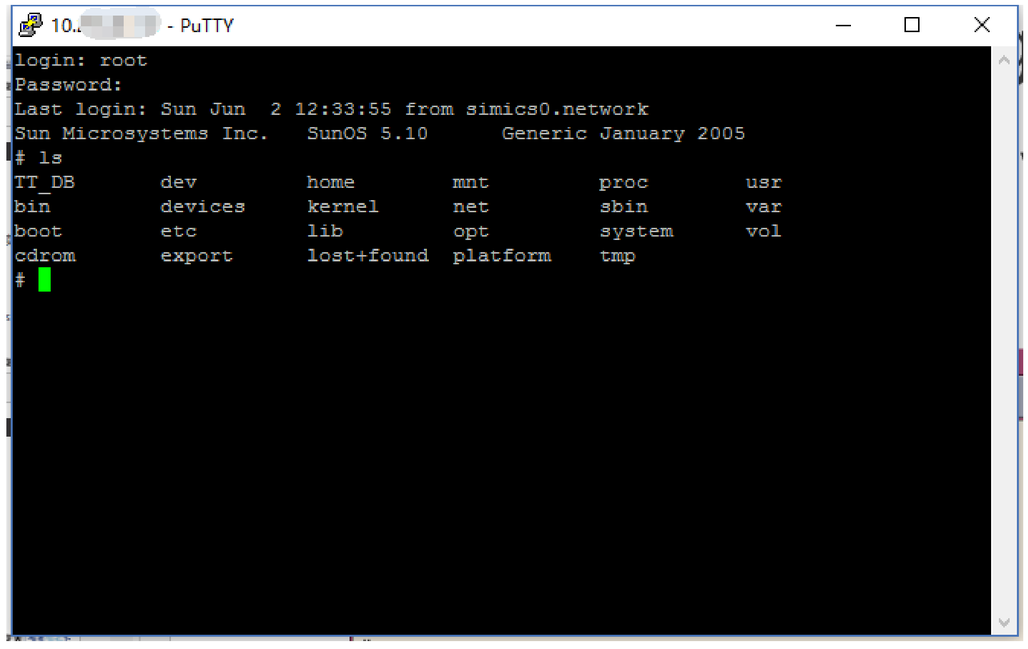


Telnet

- 修改/etc/default/login文件
- 注释掉CONSOLE=/dev/console

- Telnet 主机IP 4023





调试器

- MDB
  - 模块调试器（Modular Debugger）是用于solaris的通用调试工具，主要特性是可扩展性。
- DBX
  - UNIX下基于命令行界面的程序调试器，过交互执行子命令来达到调试的目的。
- DTrace
  - DTrace 内置在 Solaris 中，是一个全面的动态跟踪工具。用户可以使用DTrace 工具检查用户程序的行为或操作系统的行为。
- GDB

调试dump文件

调试执行的进程

```
# mdb -p 2244
Loading modules: [ ld.so.1 libc.so.1 ]
> ::stack
libc.so.1`_so_connect+4(1, ffbfb158, 10, 3c, 1, 34090)
6I6ScxUkWLR+0x84(1, d1bf7a46, 50, 0, ffbfb1e8, 1)
q6E5h2Y2rMI8XbuoscztppWHyjevBC4s6+0x74(ffbfc6f0, ffbfc004, ffbfc11c, ffbfbca4,
ffbfbcbc, 1)
B7p3zBP4s9qH8etyiNcBPezMQOuHWmiYhMP+0x30(ffbfc6f0, ffbfc004, 0, ffbfbc80, 0,
ffbfc6f0)
ZVmkso4Gv7msBh2CWiS3XgT2tERSgnO37ydnaVrMTT+0x4c(ffbfbc80, ffbfbc80, 0, ff2303a8
, dc884, 17ec8)
vsss+0x558(ffbfc198, ffbff66c, ffbff674, ffbfefe8, ffbff170, ffffc400)
_start+0x5c(0, 0, 0, 0, 0, 0)
> ::stack
libc.so.1`_so_connect+4(1, ffbfb158, 10, 3c, 1, 34090)
6I6ScxUkWLR+0x84(1, d1bf7a46, 50, 0, ffbfb1e8, 1)
q6E5h2Y2rMI8XbuoscztppWHyjevBC4s6+0x74(ffbfc6f0, ffbfc004, ffbfc11c, ffbfbca4, ffbfbcbc, 1)
B7p3zBP4s9qH8etyiNcBPezMQOuHWmiYhMP+0x30(ffbfc6f0, ffbfc004, 0, ffbfbc80, 0, ffbfc6f0)
ZVmkso4Gv7msBh2CWiS3XgT2tERSgnO37ydnaVrMTT+0x4c(ffbfbc80, ffbfbc80, 0, ff2303a8, dc884, 17ec8)
vsss+0x558(ffbfc198, ffbff66c, ffbff674, ffbfefe8, ffbff170, ffffc400)
_start+0x5c(0, 0, 0, 0, 0, 0)
> ::step
mdb: target stopped at:
libc.so.1`sigacthandler+4:      mov       %o7, %i5
> ::step
mdb: target stopped at:
libc.so.1`sigacthandler+8:      cmp       %i0, 0x24
>
```

GDB

- expat-2.0.1-sol10-sparc-local.gz
- gcc-3.4.6-sol10-sparc-local.gz
- gdb-6.8-sol10-sparc-local.gz
- libiconv-1.11-sol10-sparc-local.gz
- libintl-3.4.0-sol10-sparc-local.gz
- ncurses-5.6-sol10-sparc-local.gz

下载地址：https://ftp.ussg.iu.edu/solaris/freeware/sparc/5.10/

加载样本

- ./gdb sample

```
# ./gdb /tmp/doublefa
GNU gdb 6.8
Copyright (C) 2008 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "sparc-sun-solaris2.10"...
(no debugging symbols found)
(gdb) start
Breakpoint 1 at 0x173b0
Starting program: /tmp/doublefa
(no debugging symbols found)
(no debugging symbols found)
(no debugging symbols found)
(no debugging symbols found)
(no debugging symbols found)
(no debugging symbols found)
0x000173b0 in main ()
(gdb) Quit
(gdb)
```

```
000173B0 main:
000173B0                mov     %o7, %g1
000173B4                call    vsss
000173B8                mov     %g1, %o7
000173BC                nop
```

加密密钥


```
00017720 3B30 mov     %l2, %o1        ! 33bd8 这是key0（生成key1的key0）的偏移
00017724 3B30 add     %fp, var_620, var620 ! key1
00017728 3B30 mov     var620, %o0     ! key1
0001772C 3B30 mov     %l2, %o1        ! key0
00017730 3B30 call    CreateKey       ! CreateKey
00017734 3B30 or      %l4, 0x338, %l1 ! -34C8
00017738 3B30 add     var10, %l1, %l1 ! VAR10-34C8
```

```
(gdb) x /64xw 0xffbff7d8
0xffbff7d8:     0xe0cdbee9      0xdb4d9fa8      0x2bac42c3      0xcbab7724
0xffbff7e8:     0xf852c15a      0x78f03e5b      0x690a01cb      0x8c858f29
0xffbff7f8:     0xef7c9c03      0x8b0e365e      0x287640c0      0x24f29c9c
0xffbff808:     0x72029d81      0xb5bb6a4f      0x1473425b      0x7573f288
0xffbff818:     0x9837f98b      0x2b649f3b      0xc7ffc4a3      0xc167408a
0xffbff828:     0x54659f25      0xff483645      0x1a0586e2      0x2bac94f4
0xffbff838:     0x83e5d508      0xeead2cbe      0xcb98a6d0      0xeeed358d
0xffbff848:     0xf28cf0c4      0x0387bacd      0x133d2754      0x056a9ba7
0xffbff858:     0x213002c7      0x3b586705      0x0a44a1e6      0x863c1637
0xffbff868:     0x208bbce9      0x287e981a      0xcaf77fe6      0x31389ef7
0xffbff878:     0x932ff07f      0xf0282b11      0xfcb711ff      0xcb86631c
0xffbff888:     0x00000000      0x00000000      0x00000000      0x00000000
```

有效防护 价值输出

- 域名

```
(gdb) x /64b 0xffbff960
0xffbff960:     -1 '▒'  -65 '▒' -55 '▒' -120 '\210'     -1 '▒'  -65 '▒' -55 '▒'
32 ' '
0xffbff968:     0 '\0'  0 '\0'  0 '\0'  0 '\0'  -1 '▒'  -1 '▒'  -1 '▒'  -1 '▒'
0xffbff970:     0 '\0'  0 '\0'  0 '\0'  0 '\0'  0 '\0'  0 '\0'  0 '\0'  0 '\0'
0xffbff978:     0 '\0'  0 '\0'  0 '\0'  0 '\0'  0 '\0'  0 '\0'  0 '\0'  0 '\0'
0xffbff980:     ▒▒ ▒▒▒ ▒▒▒▒ ▒▒▒▒ ▒▒▒ ▒▒▒ ▒▒▒ ▒▒▒▒ ▒▒ ▒▒▒▒ ▒▒▒' 97 'a'  108 'l'
0xffbff988:     108 'l' 110 'n' 105 'i' 110 'n' 106 'j' 97 'a'  116 't' 101 'e'
0xffbff990:     99 'c'  104 'h' 46 '.'  99 'c'  111 'o' 109 'm' 0 '\0'  0 '\0'
0xffbff998:     0 '\0'  0 '\0'  0 '\0'  0 '\0'  0 '\0'  0 '\0'  0 '\0'  0 '\0'
```

- 硬编码IP

```
(gdb) x /32ub 0xffbffcb4
0xffbffcb4:     1▒▒     128     235     237     0       0       0       0
0xffbffcbc:     0       0       0       0       0       0       0       0
0xffbffcc4:     0       0       0       0       0       0       0       0
0xffbffccc:     0       0       0       0       0       0       0       0
```

谢谢