

VXWORKS FUZZING 之道

VXWORKS 工控实时操作系统漏洞挖掘揭秘

吴少荣 (404安全研究员)

TDG (404安全研究员)

我们



KCon West 2016

本文内容

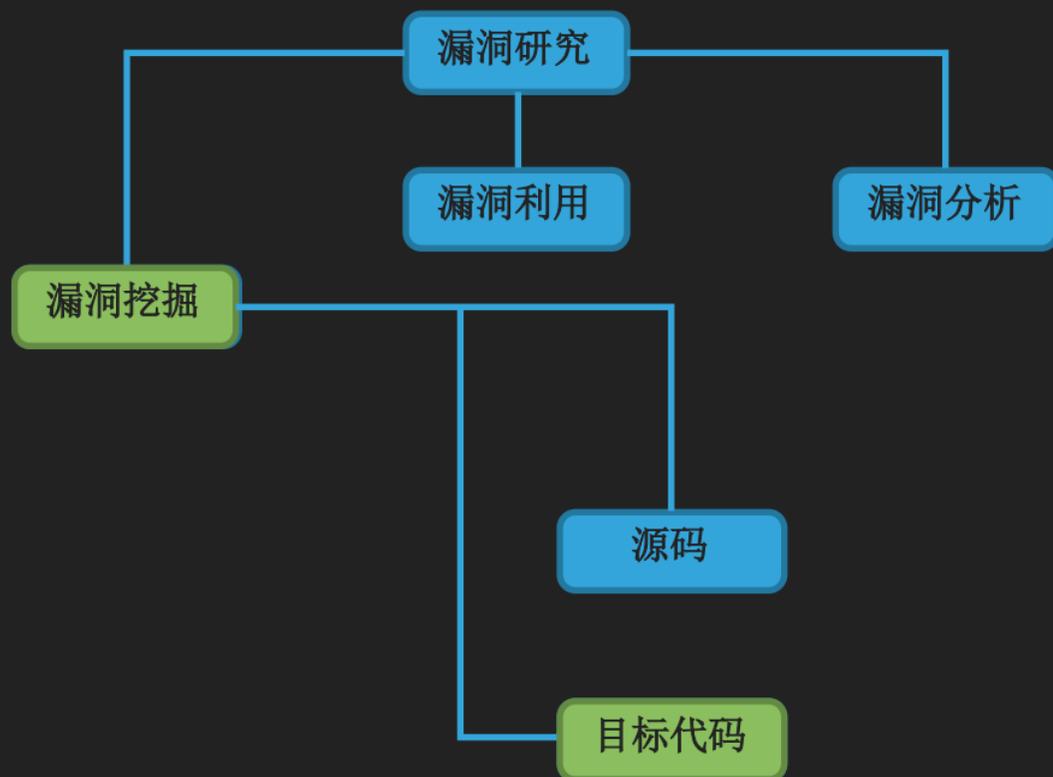
- ▶ 漏洞研究简介
- ▶ 研究环境准备
- ▶ 如何实现自动Fuzz（WDB、崩溃机制）
- ▶ 两个Fuzz实例
- ▶ Vxworks 系统组件无法调试问题
- ▶ 调查互联网上暴露的VxWorks WDB RPC服务
- ▶ 总结

漏洞研究-简介

未知漏洞探索

综合应用各种技术和工具

尽可能的找出潜在的漏洞

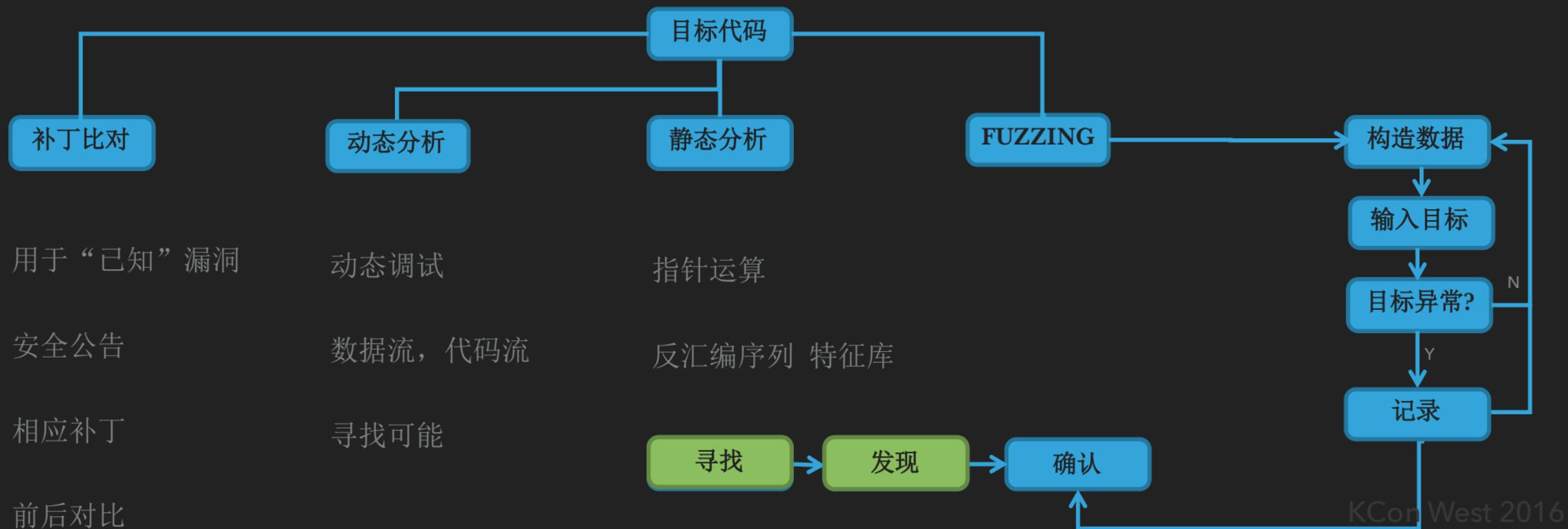


已发现漏洞的细节进行深入分析

漏洞利用

补救措施

漏洞研究-简介



实现目标





SULLEY 安装

- <http://www.freebuf.com/news/93201.html>
- Python灰帽子 第9章 Sulley
- 网络资料

环境准备



关于VXWORKS

美国WindRiver, 于1983年设计开发。已宣称15亿台设备, 使用广泛

支持几乎所有现代市场上的嵌入式CPU架构

x86、MIPS、PowerPC、SPARC、SH-4、ARM、StrongARM、xScale CPU

其市场范围跨越所有安全关键领域

- 1) 火星好奇心流浪者
- 2) 波音787梦幻客机
- 3) 网络路由器

KCon West 2016

特殊应用场景, 高危性质使的VxWorks安全被高度关注。

环境准备



VXWORKS 安装

无法涉及所有细节，提供资料以供参考

✓ [VxWorks 5.5 模拟环境搭建](#)

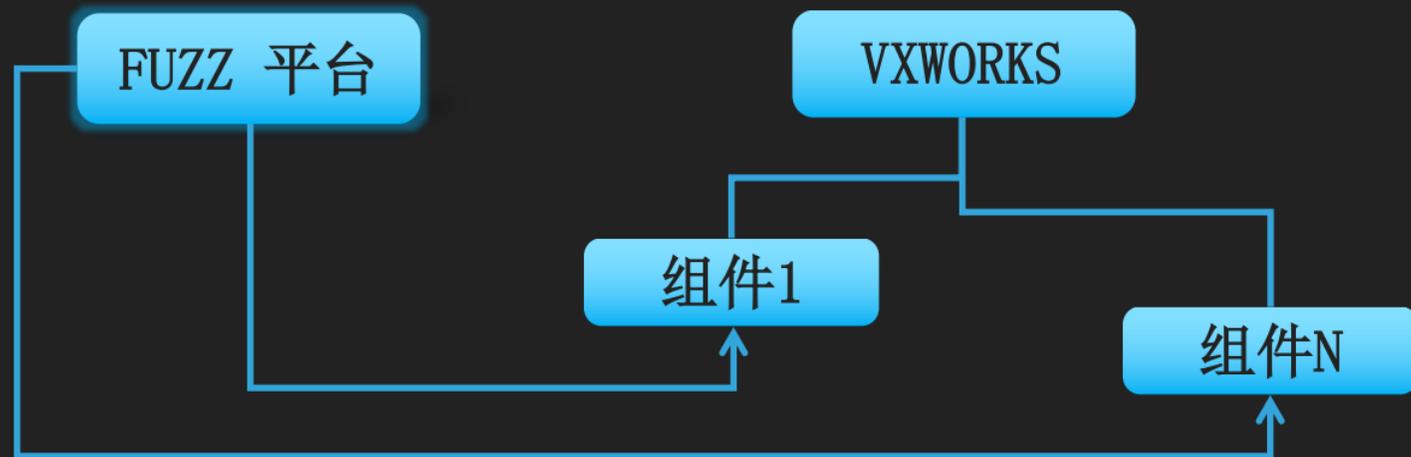
✓ [VxWorks 6.6 模拟环境搭建](#)

✓ [VmWare上运行VxWorks\(5.5\)](#)

2011年发布 | frank

<https://github.com/knownsec/VxPwn> VxWorks漏洞挖掘相关

实现目标



实现自动Fuzzing

- ✓ 构造半随机数据
- ✓ 检测组件状态
- ✓ 记录信息
- ✓ 输入目标组件
- ✓ 获取组件异常信息
- ✓ 目标组件环境复原

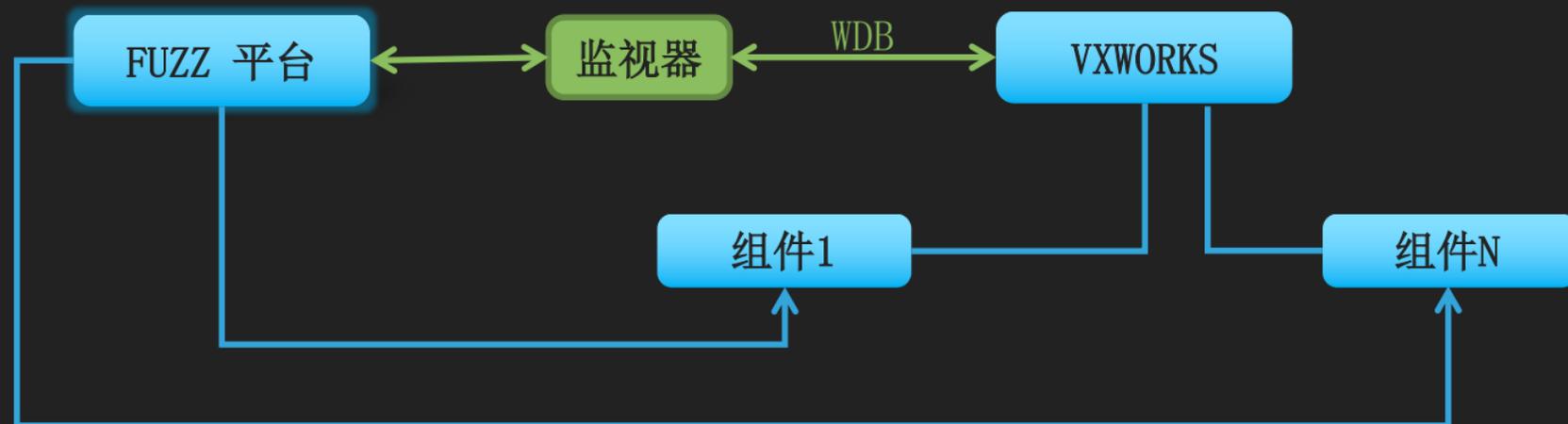
2015-9月

44CON



Yannick Formaggio介绍了他对VxWorks安全研究的心得，他采用了Fuzzing框架Sulley对VxWorks系统的多个协议进行了Fuzzing，挖掘到一些漏洞，并结合VxWorks的WDB RPC实现了一个远程调试器(监视器)。

实现目标



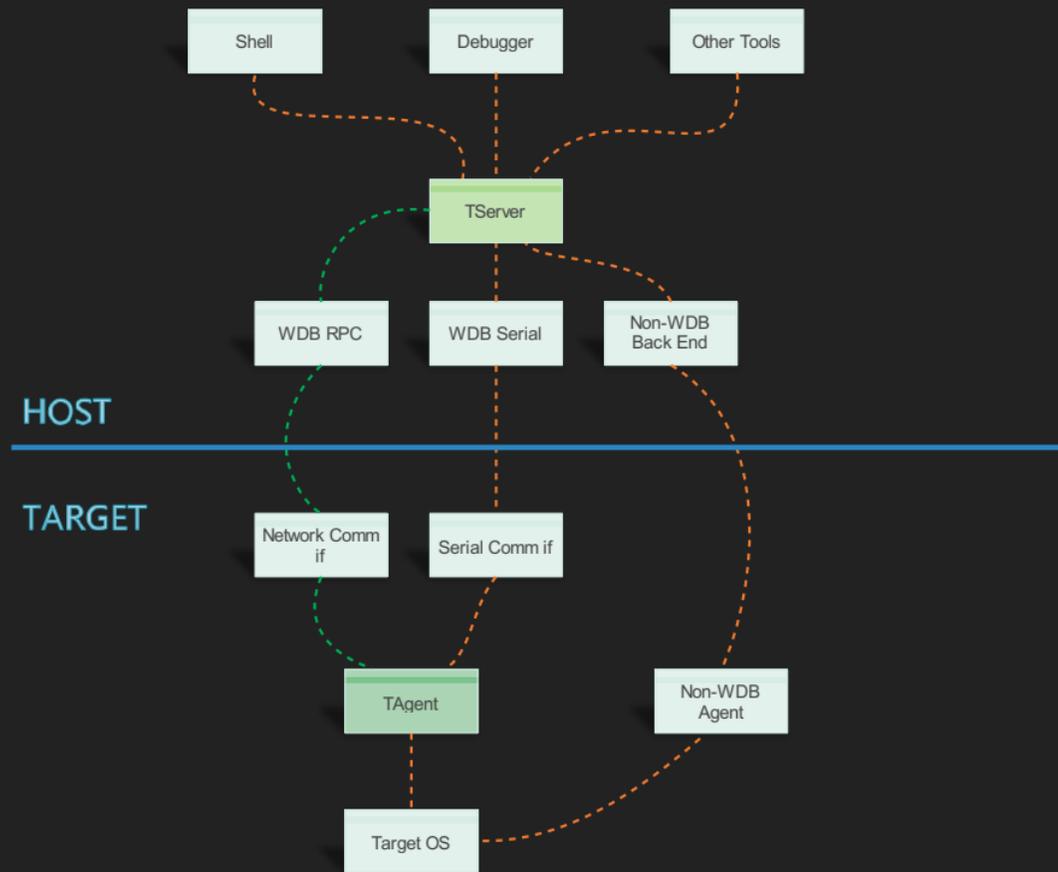
实现自动Fuzzing

- ✓ 构造半随机数据
- ✓ 检测组件状态
- ✓ 目标组件环境复原
- ✓ 输入目标组件
- ✓ 获取组件异常信息
- ✓ 记录信息

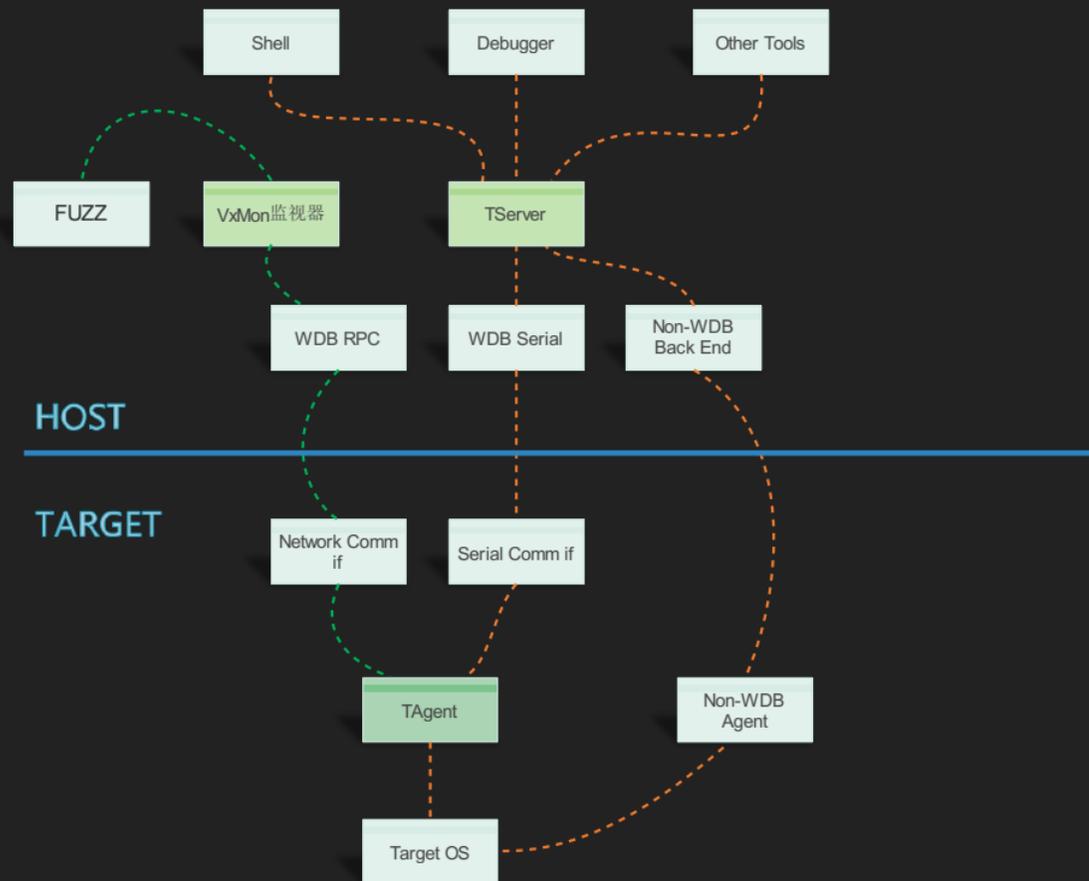
WDB RPC 协议

- 调试接口
- 基于SUN-RPC协议
 - ✓ 直接读写系统内存
 - ✓ 感知系统组件状态
- 服务运行在UDP协议的17185端口上
- WDB RPC被包含在VxWorks TAgent模块中
- 版本 V1, V2

VXWORKS 调试通信框架



FUZZ框架思路



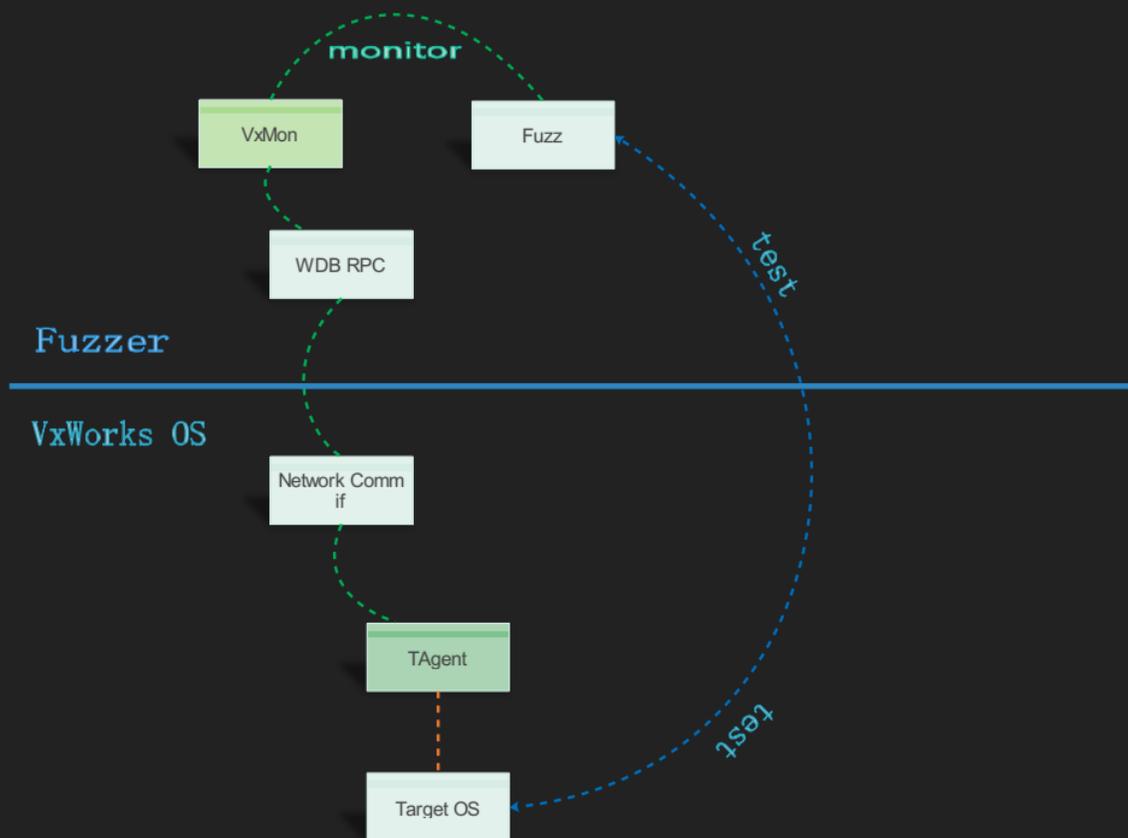
VxMon 充当调试器

模拟Debugger与TAgent通信

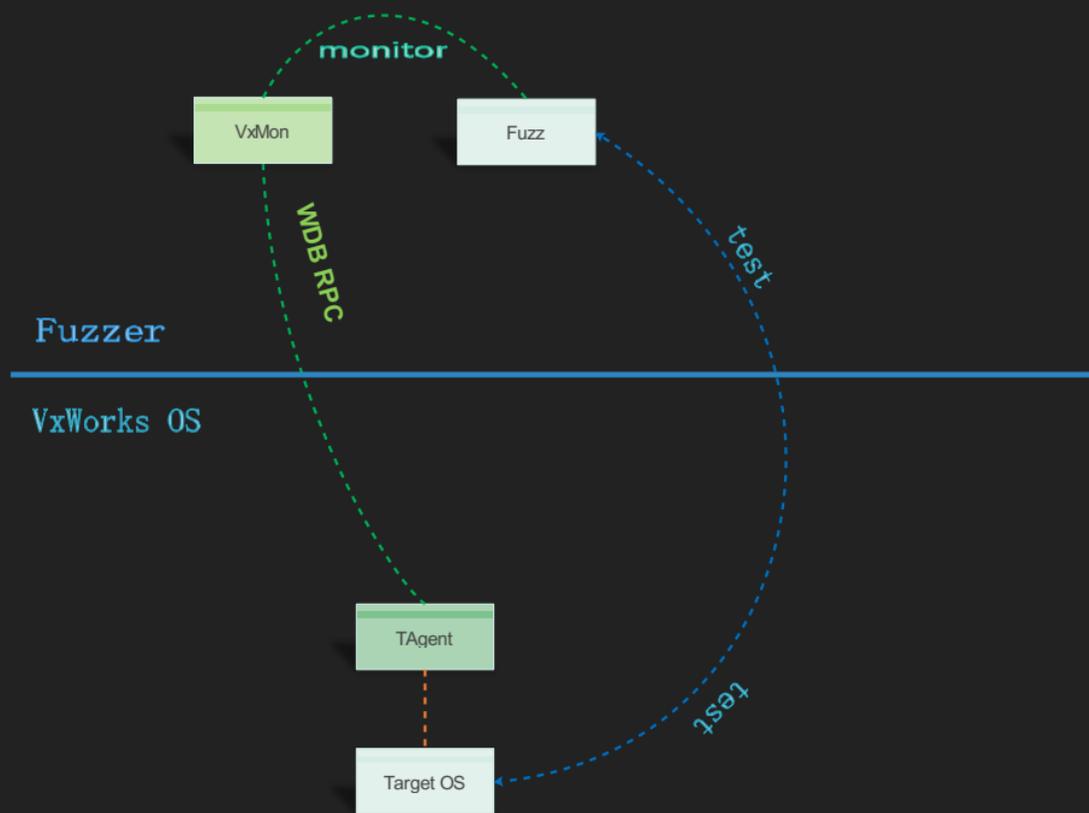
VxMon从TAgent获得异常通知

解决技术难点

FUZZ框架



FUZZ框架



WDB PRC 请求数据包

IP HEADER 20 bytes	----- IP Header ----- IP: Version = 4 IP: Header length = 20 bytes IP: Type of service = 0x00 IP: Total length = 96 bytes IP: Identification = 13825 IP: Flags = 0x0 IP: Fragment offset = 0 bytes IP: Time to live = 128 seconds/hops IP: Protocol = 17 (UDP) IP: Header checksum = 3d7e IP: Source address = 147.11.80.136 IP: Destination address = 147.11.80.111 IP: No options
UDP HEADER 8 bytes	----- UDP Header ----- UDP:Source port = 690 UDP:Destination port = 0x4321 (Sun RPC) UDP:Length = 76 UDP:Checksum = F729
RPC REQUEST HEADER 40 bytes (10 32-bit words)	----- SUN RPC Header ----- RPC:Transaction id = 1033B54674 RPC:Type = 0 (Call) RPC:RPC version = 2 RPC:Program = 1431655765, vers = 1, proc = 1 RPC:Credentials: Flavor = 0, len = 0 bytes RPC:Verifier : Flavor = 0, len = 0 bytes
XDR Stream WDB PARAMETER WRAPPER 12 bytes	Checksum Packet Size Sequence #
XDR encoded PARAMETERS (common information WDB_CORE and function input parameters)	--

* IP Header

* UDP Header

* RPC Request Header

* WDB Parameter Wrapper

* Function input parameters

重点内容: WDB Parameter Wrapper内容包含整个请求包的大小, 校验和及请求系列号, Function input parameters 为请求功能号的携带辅助信息。

WDB PRC 应答数据包

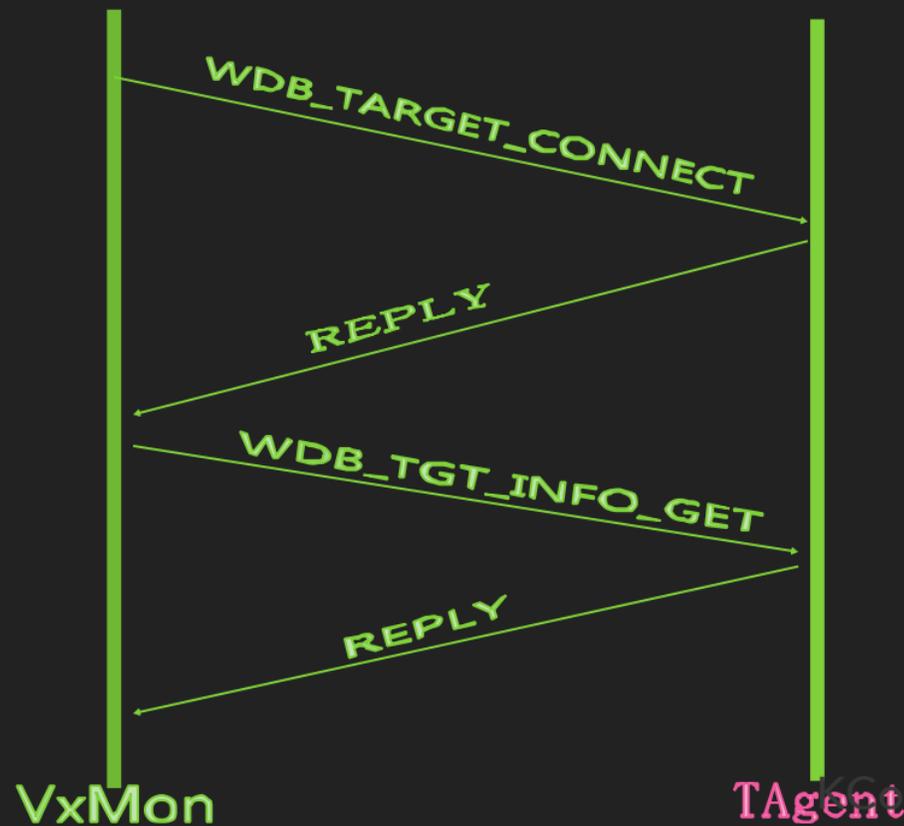
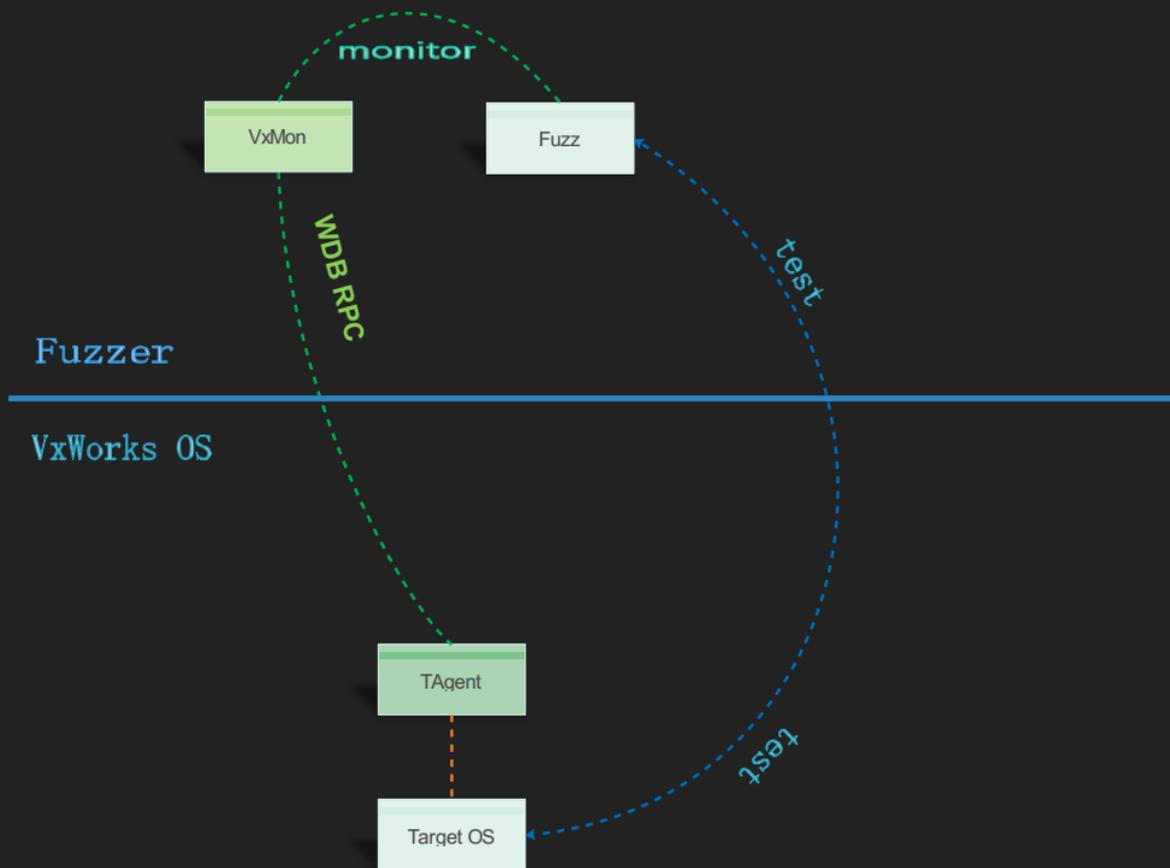
IP HEADER 20 bytes	----- IP Header ----- IP: Version = 4 IP: Header length = 20 bytes IP: Type of service = 0x00 IP: Total length = 64 bytes IP: Identification = 340 IP: Flags = 0x0 IP: Fragment offset = 0 bytes IP: Time to live = 32 seconds/hops IP: Protocol = 17 (UDP) IP: Header checksum = d24b IP: Source address = 147.11.80.111 IP: Destination address = 147.11.80.136 IP: No options
UDP HEADER 8 bytes	----- UDP Header ----- UDP:Source port = 0x4321 UDP:Destination port = 974 (Sun RPC) UDP:Length = 44 UDP:Checksum = D7FD
RPC REPLY HEADER 24 BYTES (6 32-bit words)	----- SUN RPC Header ----- RPC:Transaction id = 1251462914 RPC:Type = 1 (Reply) RPC:This is a reply to frame 28 RPC:Status = 0 (Accepted) RPC:Verifier : Flavor = 0, len = 0 bytes RPC:Accept status = 0 (Success)
XDR Stream WDB REPLY WRAPPER 12 bytes	Checksum Packet Size WDB Error Status
XDR encoded REPLY (function output)	--

- * IP Header
- * UDP Header
- * RPC Reply Header
- * WDB Reply Wrapper
- * Function output

重点内容: WDB Parameter Wrapper内容包含整个请求包的大小、校验和及应答系列号(在每个请求与应答中, 应答与请求系列号一致), Function output包含应答的输出信息, 为请求功能号的返回信息。

V2版本的WDB RPC与V1版本最大的区别在于，在发送各类请求（如获取VxWorks版本BSP信息等的请求WDB_TGT_INFO_GET）时，V1只用发送对应的请求包即可。而V2维护了一种类似Session的机制，在发送各类请求前，需要发送一个连接请求包（WDB_TARGET_CONNECT）以成功连接至TAgent，对于每个Session中的多个请求包（包括连接请求包），它们的SUN RPC -> Transaction ID字段及WDB RPC -> sequence字段的值需是连续递增的，否则就会收到包含错误的响应包。

建立通信-举例



建立通信-连接请求

```
//WDB WDB_TARGET_CONNECT 请求包
5784ac6a //Transaction ID
00000000 //Type is call
00000002 //RPC version
55555555 //Program
00000001 //ver
0000007a //function id = WDB_TARGET_CONNECT ( )

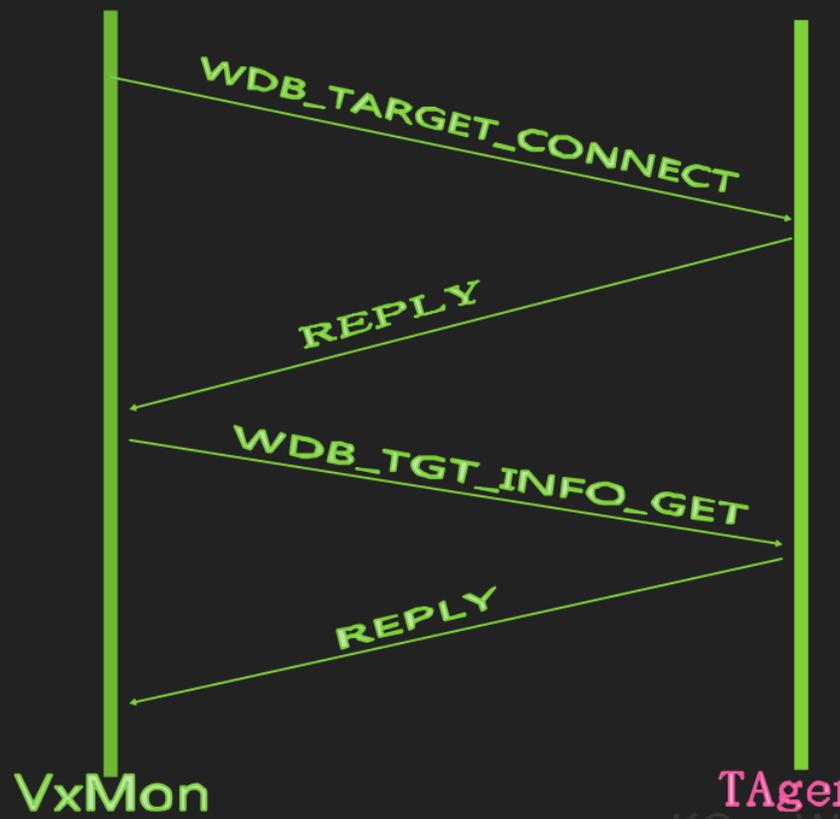
00000000
00000000
00000000
00000000

ffffd0ff //checksum
00000060 //packet size
0f100001 //sequence

00000002
00000000
00000000

00000001 //Function input parameters
00000019 //length "Vxworks6x_192.168.102.88"
5678576f726b7336785f3139322e3136382e3136322e383800000000
```

```
0000 00 0c 29 b3 26 8c 00 0c 29 50 06 45 08 00 45 00 ..).&... )P.E..E.
0010 00 80 14 79 00 00 40 11 18 4a c0 a8 66 01 c0 a8 ...y.0. .J.f...
0020 66 58 02 83 43 21 00 6c 67 08 57 84 ac 6a 00 00 fx..cl.] g...j...
0030 00 00 00 00 00 02 55 55 55 55 00 00 00 01 00 00 :z.....
0040 00 7a 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0050 00 00 ff ff 00 ff 00 00 00 60 0f 10 00 01 00 00 .....
0060 00 02 00 00 00 00 00 00 00 00 00 00 01 00 00 .....
0070 00 19 56 78 57 6f 72 6b 73 36 78 5f 31 39 32 2e ..Vxwork s6x_192.
0080 31 36 38 2e 31 30 32 2e 38 38 00 00 00 00 00 00 168.102. 88....
```



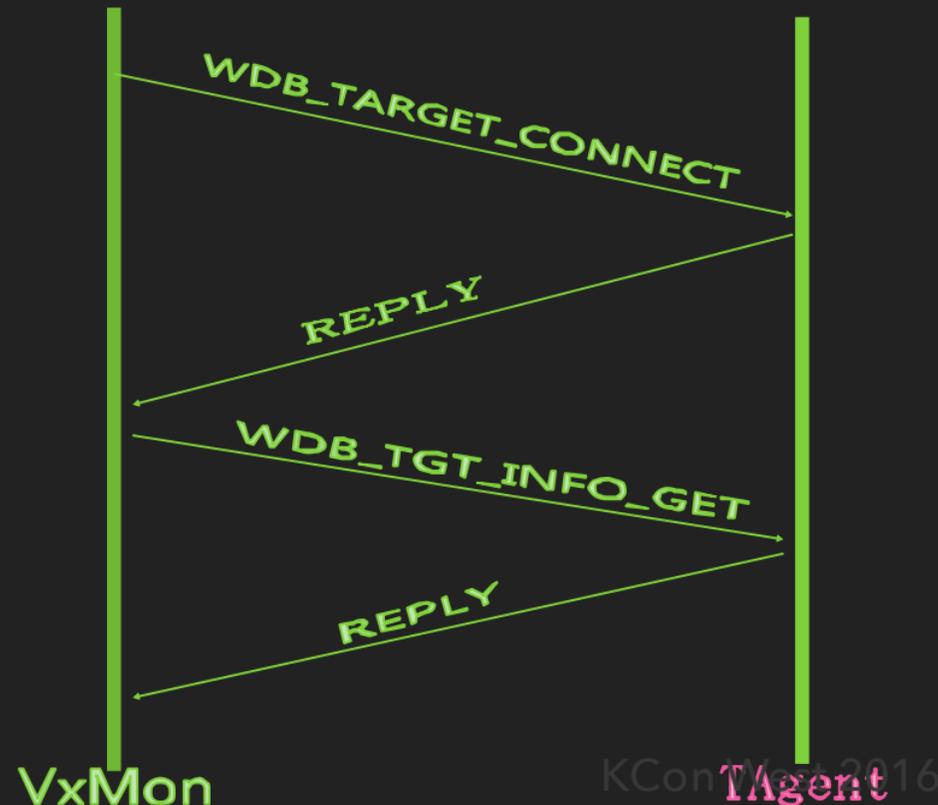
建立通信-连接应答

```
////WDB WDB_TARGET_CONNECT 应答包
5784ac6a      //Transaction ID
00000001      //Type is reply
00000000
00000000
00000000
00000000

ffff273a     //checksum
0000004c     //packet size
00000000     //WDB status

00000004     //WDB_TGT_INFO
352e3000     //"5.0"
00000200
00000003
00000002
00000008     //length
5678576f726b7300 //"Vxworks"
00000004
00000004
eeeeeeee
```

0000	00 0c 29 50 06 45 00 0c 29 b3 26 8c 08 00 45 00	..)P.E..)&...E.
0010	00 6c 00 00 00 00 20 11 4c d7 c0 a8 66 58 c0 a8	.l.... L...fX..
0020	66 01 43 21 02 83 00 58 00 00 57 84 ac 9a 00 00	f.c!...x ..w..j..
0030	00 01 00 00 00 00 00 00 00 00 00 00 00 00 00
0040	00 00 ff ff 27 3a 00 00 00 4c 00 00 00 00 00	...5.0... ..
0050	00 04 35 2e 30 00 00 00 02 00 00 00 03 00 00vx works...
0060	00 02 00 00 00 08 56 78 57 6f 72 6b 73 00 00
0070	00 04 00 00 00 04 ee ee ee ee



建立通信-获取信息请求

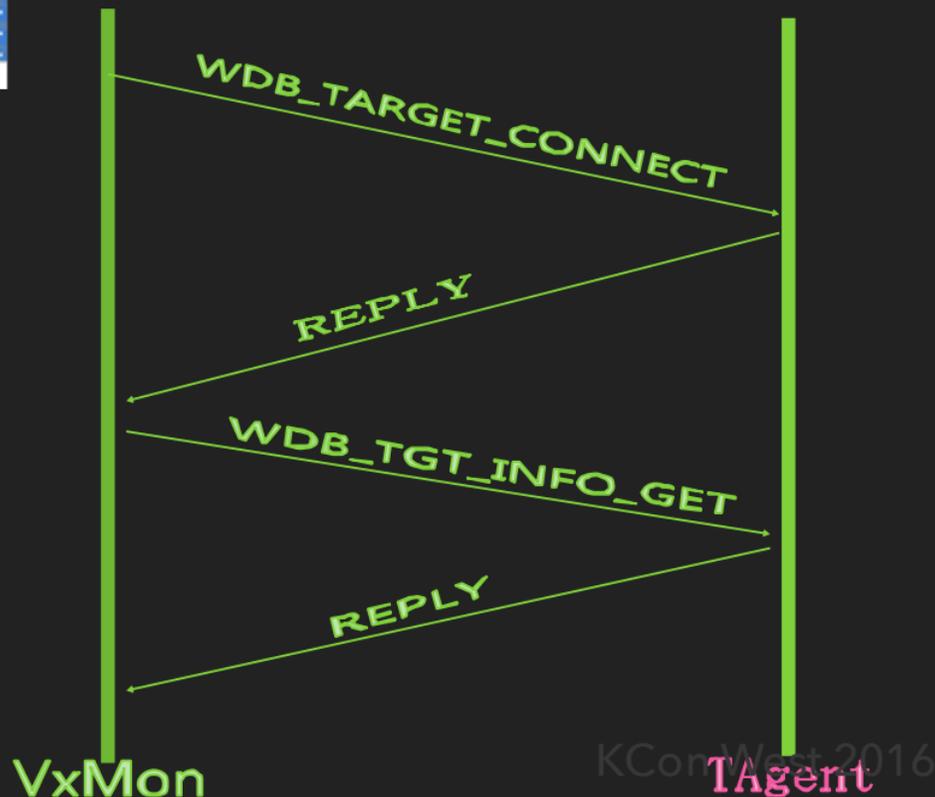
```
//WDB_TGT_INFO_GET 请求包
5884ac6a
00000000
00000002
55555555
00000001
0000007b //function id = WDB_TGT_INFO_GET

00000000
00000000
00000000
00000000

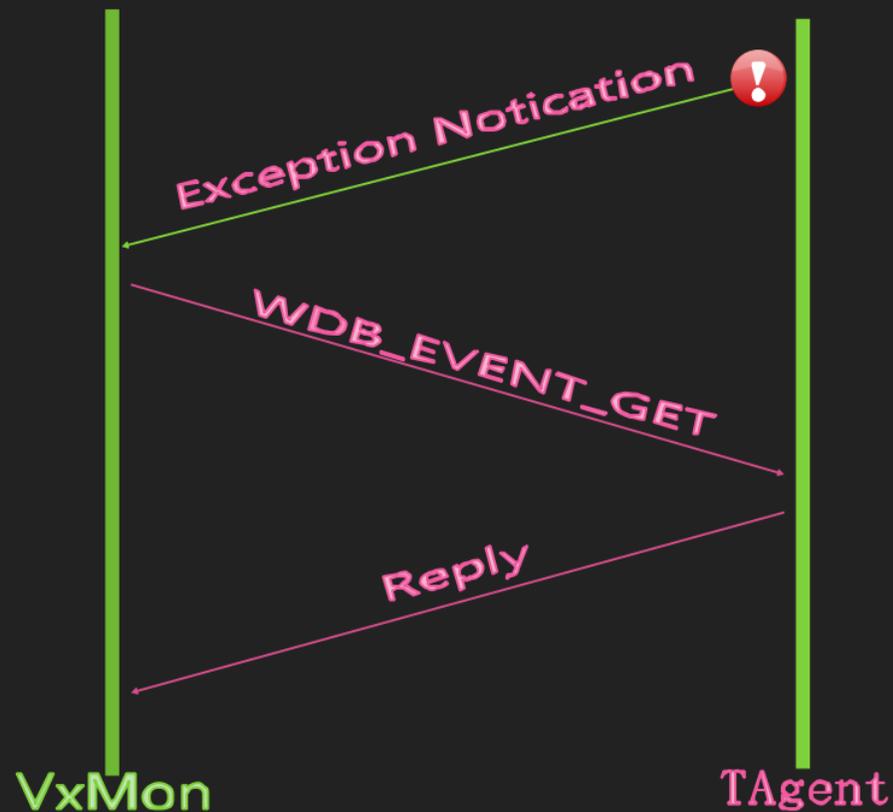
ffff457a //checksum
00000044 //packet size
0f100002 //sequence

00000003
00000000
00000000
00000004
00000000
```

0000	00 02 29 b3 26 8c 00 0c	29 50 06 45 08 00 45 00	..).&... JP.E..E.
0010	00 64 14 7a 00 00 40 11	18 65 c0 a8 66 01 c0 a8	.d.2..@. .e..f...
0020	66 58 02 83 43 21 00 50	67 10 58 84 ac 6a 00 00	fx..c!.P g.X.].
0030	00 00 00 00 00 02 55 55	55 55 00 00 00 01 00 00UU UU.....
0040	00 7b 00 00 00 00 00 00	00 00 00 00 00 00 00 00	{..... ..
0050	00 00 ff ff 45 7a 00 00	00 44 0f 10 00 02 00 00	...Ez.. .D.....
0060	00 03 00 00 00 00 00 00	00 00 00 00 00 04 00 00
0070	00 00		= ..



崩溃机制检测



崩溃机制检测

特征:

a) 循环通知

b) WDB_EVENT_GET包确认

```
⊕ User Datagram Protocol, Src Port: 17185 (17185), Dst Port: 49366 (49366)
⊖ Data (24 bytes)
  Data: 00000000000000000000000001fffffffffeeeeeee00000005
  [Length: 24]

0000  00 0c 29 4a 92 d0 00 0c 29 b3 26 8c 08 00 45 00  ..)J.... ).&...E.
0010  00 34 03 00 00 00 20 11 4a 04 c0 a8 66 58 c0 a8  .4.... . J...fX..
0020  66 0c 43 21 c0 d6 00 20 00 00 00 00 00 00 00 00  f.c!... ..
0030  00 00 00 00 00 01 ff ff ff ff ee ee ee ee 00 00  .....
0040  00 05
```

崩溃机制检测-异常信息请求

```
//WDB_EVENT_GET 请求包
11112224 //Transaction ID
00000000 //Type is call
00000002 //RPC version
55555555 //Program
00000001 //ver
00000046 //function id = WDB_EVENT_GET( )

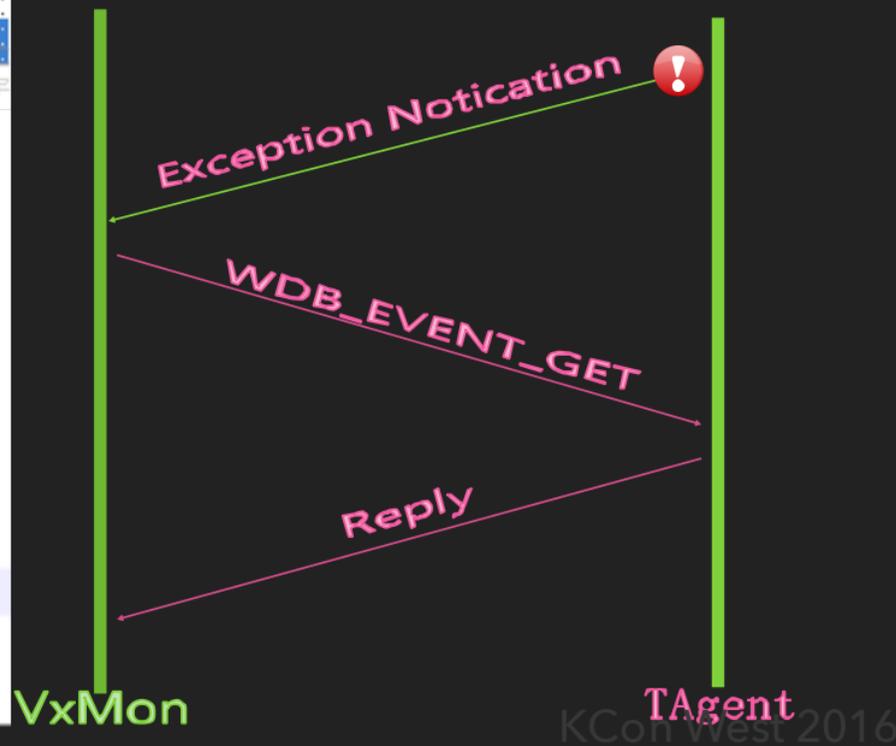
00000000
00000000
00000000
00000000

00000000
00000030 //packet size
33334446 //sequence
```

Hex dump (Data: 52 bytes, Length: 52):

0000	00 0c 29 b3 26 8c 00 0c	29 4a 92 d0 08 00 45 00	..).&...)J...E.
0010	00 50 01 98 00 00 80 11	eb 4f c0 a8 66 0c c0 a8	.P.....O..f...
0020	66 58 c0 d6 43 21 00 3c	57 f6 11 11 22 24 00 00	fX..Cl.< w..f...\$
0030	00 00 00 00 02 55 55 55	55 55 00 00 00 01 00 00UU UU.....
0040	00 46 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.F.....0330F
0050	00 00 00 00 00 00 00 00	00 30 33 33 44 46	

KNOWNSSEC 知道创宇



崩溃机制检测

```
vxworks
UxWorks 6.6
KERNEL: WIND version 2.11
Copyright Wind River Systems, Inc., 1984-2007

CPU: PC PENTIUM3, Processor #0.
Memory Size: 0xe6a000. BSP version 2.0/9.
Created: Dec 31 2015, 14:05:00
ED&R Policy Mode: Deployed
MDB Comm Type: MDB_COMM_END
MDB: Ready.

->
Page Fault
Page Dir Base : 0x00779000
Esp0 0x008f4440 : 0x008f44a4, 0x008f4468, 0x007a0024, 0x00000200
Esp0 0x008f4450 : 0x003d8600, 0x008f44a4, 0x008f4468, 0x00000200
Program Counter : 0x003dc6ef
Code Selector : 0x00000000
Eflags Register : 0x00010202
Error Code : 0x00000000
Page Fault Addr : 0x226a69e0
Task: 0x79622c "tPortmapd"
0x79622c (tPortmapd): task 0x79622c has had a failure and has been stopped.
0x79622c (tPortmapd): fatal kernel task-level exception!
```



更多...

接下来主机请求更多的信息，如崩溃时寄存器内容，内存区域，异常代码。

通过VxMon发送WDB_REGS_GET请求，可以获取异常寄存器内容。

通过VxMon发送WDB_MEM_READ请求，可以获取异常地址的执行代码。

```
bok = target.pub_Connect()

while 1:
    #get exception information
    bok,info = target.pub_IsAbort()
    time.sleep(1)

    if bok :
        print "\n-----event infomation-----\n"
        print "%s" % info ['eve']
        print "\n-----reg infomation-----\n"
        print "%s" % info['reg']
        print "\n-----code infomation-----\n"
        print "%s" % info['asm']
        break
```



Debug I/O (stdin, stdout, stderr) appears below

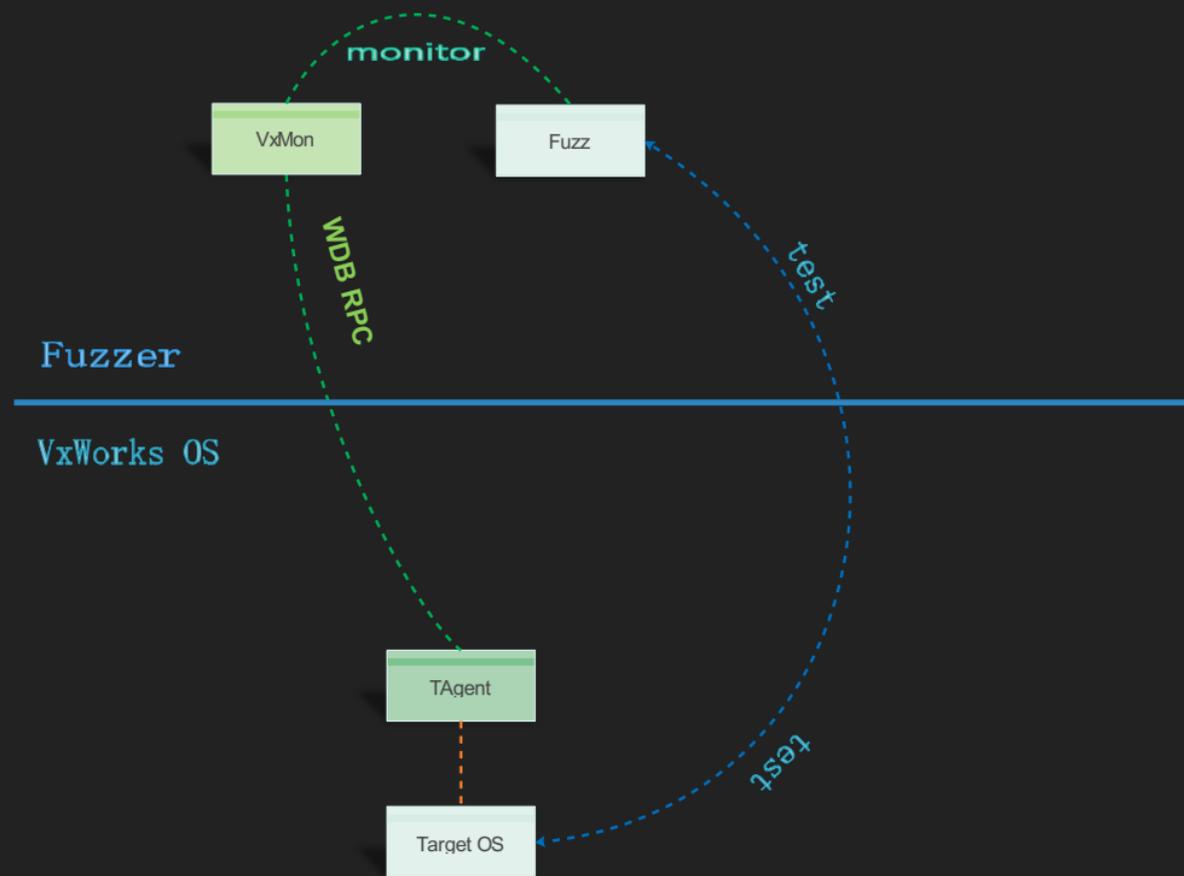
```
-----reg infomation-----
{'esp': 9389120, 'edi': 9389812, 'pc': 4048623, 'eax': 9389220, 'ebp': 9389132, 'eflags': 642, 'edx': 17, 'ebx': 9389132}

-----code infomation-----
0x3dc6ef: call dword ptr [ebx*4 + 0x4847c0]
0x3dc6f6: add esp, 8
0x3dc6f9: jmp @x3dc700
0x3dc6fb: mov eax, 2
0x3dc700: pop ebx
0x3dc701: mov esp, ebp
0x3dc703: pop ebp
0x3dc704: ret
0x3dc705: nop
```

wbdbog.py

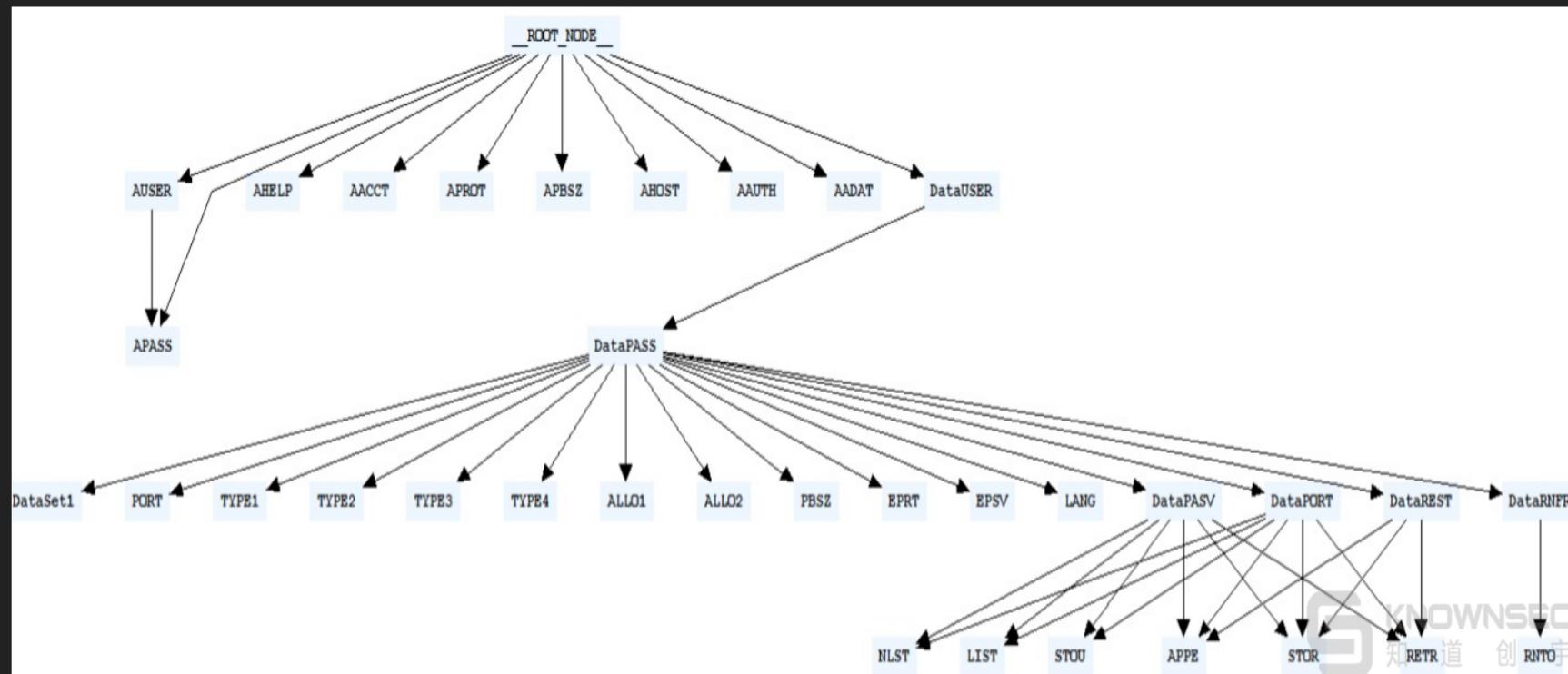
KNOWNSEC 知道创造 KCon West 2016

FUZZ框架



FTP协议 (TCP/21) FUZZING

FTP协议中很多命令需要在登录后才能执行，我们主要关注未登录的情况。fuzz的协议字段节点图如下：



FTP协议 (TCP/21) FUZZING

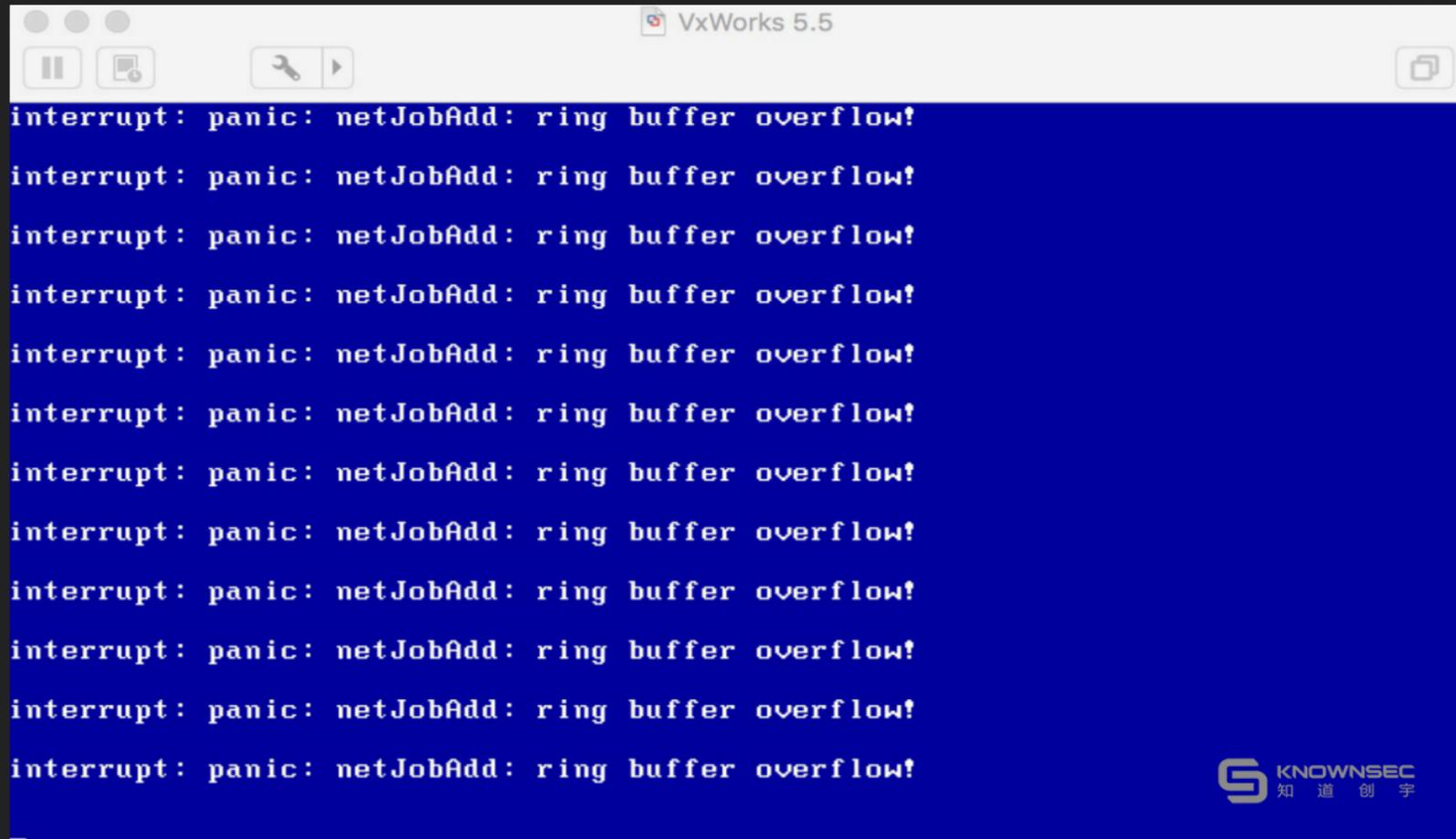
FTP协议中很多命令需要在登录后才能执行，我们主要关注未登录的情况。

fuzz结果：

* 6.6版本无影响。

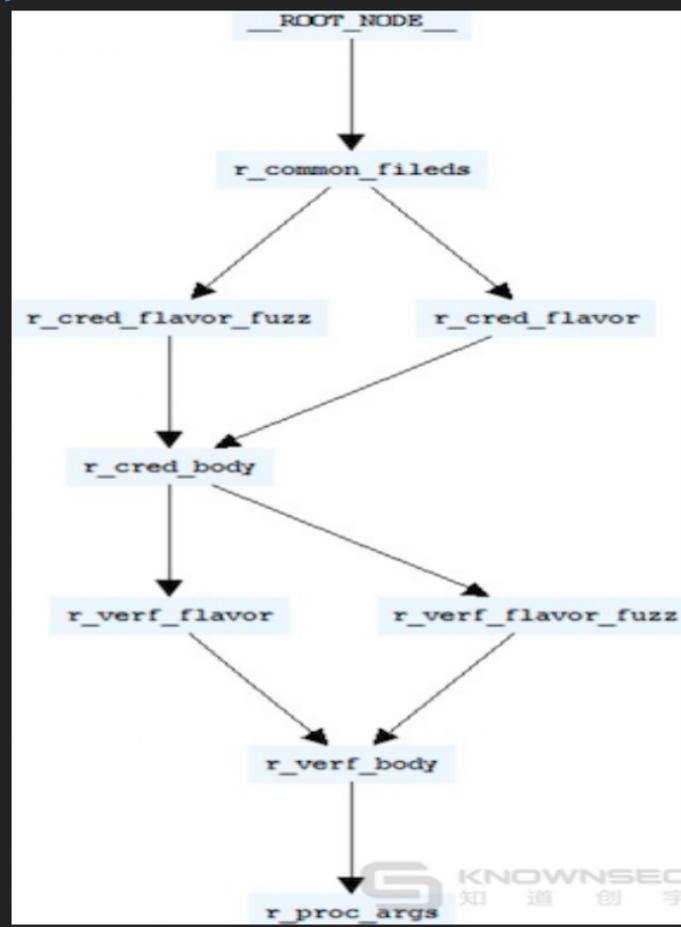
* 5.5连续发送极大的FTP请求包时，会造成ring buffer overflow，导致VxWorks无法进行网络通信。该问题也属于上文中已经提到的网络栈问题，不属于FTP协议问题。

网络栈问题



```
interrupt: panic: netJobAdd: ring buffer overflow!  
interrupt: panic: netJobAdd: ring buffer overflow!
```

SUNRPC协议 RPCBIND服务 FUZZING



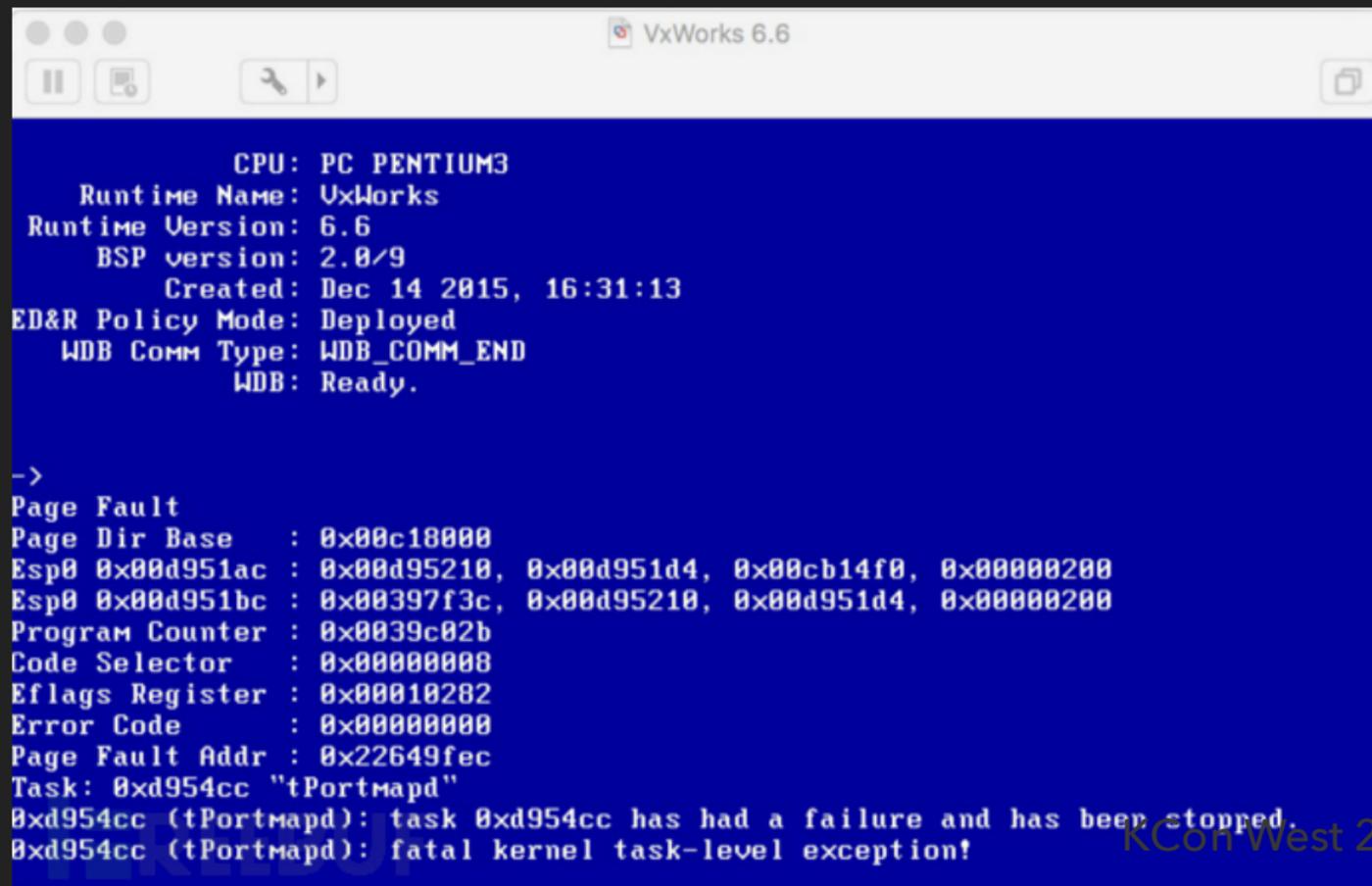
SUNRPC协议 RPCBIND服务 FUZZING

Fuzzing结果:

5.5及6.6版本均测试出18处崩溃点, (Payload 存在Github) 通过观察结果中的寄存器状态, 都属于一类, 该漏洞仅造成tPortmapd服务崩溃, 对其他服务没有影响。

RPCBIND 服务问题

rpcbind服务是SUN-RPC的一部分，在VxWorks系统中该服务监听在tcp/111及udp/111端口，攻击者向该端口发送经过特殊构造的数据包，可使rpcbind服务崩溃，精心构造的请求可能可以造成任意代码执行。终端会给出错误信息，报错信息如下图：



```
CPU: PC PENTIUM3
Runtime Name: UxWorks
Runtime Version: 6.6
BSP version: 2.0/9
Created: Dec 14 2015, 16:31:13
ED&R Policy Mode: Deployed
WDB Comm Type: WDB_COMM_END
WDB: Ready.

->
Page Fault
Page Dir Base : 0x00c18000
Esp0 0x00d951ac : 0x00d95210, 0x00d951d4, 0x00cb14f0, 0x00000200
Esp0 0x00d951bc : 0x00397f3c, 0x00d95210, 0x00d951d4, 0x00000200
Program Counter : 0x0039c02b
Code Selector : 0x00000008
Eflags Register : 0x00010282
Error Code : 0x00000000
Page Fault Addr : 0x22649fec
Task: 0xd954cc "tPortmapd"
0xd954cc (tPortmapd): task 0xd954cc has had a failure and has been stopped.
0xd954cc (tPortmapd): fatal kernel task-level exception!
```

VXWORKS 6.6-调试

VxWorks6x_192.168.102.88 (2)

tPortmapd : 0x7ad28c

```
003ddf76: mov    eax,dword ptr [_null_auth]
003ddf7c: mov    dword ptr [ebx+0x20],eax
003ddf7f: mov    dword ptr [ebx+0x28],0
003ddf86: mov    eax,dword ptr [ebp+8]
003ddf89: mov    ebx,dword ptr [eax+0xC]
003ddf8c: cmp    ebx,2
003ddf8f: jnle  0x3DDFA3
003ddf91: push  dword ptr [ebp+0xC]
003ddf94: push  dword ptr [ebp+8]
003ddf97: call  near [ebx*4+svcauthsw]
003ddf9e: add    esp,8
003ddfa1: jmp   0x3DDFA8
003ddfa3: mov    eax,2
003ddfa8: pop    ebx
003ddfa9: mov    esp,ebp
003ddfab: pop    ebp
003ddfac: ret
003ddfad: nop
003ddfae: nop
003ddfaf: nop
    _svcauth_null:
003ddfb0: push  ebp
```

Debug

VxWorks6x_192.168.102.88 (2) [Attach to Target]

- Pentium3: VxWorks 6.6 (Task Mode)
 - tPortmapd : 0x7ad28c (Stopped - Exception)
 - _authenticate() - 0x003ddf97
 - svc_getreqset() - 0x003d9ea3
 - svc_run() - 0x003da078
 - portmapd() - 0x003d8d57
 - 0x0032cc6f

(*)= Variables 1010 Registers X Expressions Memory

Name	Value
General	
1010 0101 eax	0x009074A4
1010 0101 ecx	0x007B7554
1010 0101 edx	0x00000011
1010 0101 ebx	0x88888888
1010 0101 esp	0x00907440
1010 0101 ebp	0x0090744C
1010 0101 esi	0x80000000
1010 0101 edi	0x009076F4
1010 0101 eflags	0x00000282
1010 0101 pc	0x003DDF97

VXWORKS 5.5-调试问题

The screenshot shows the Tornado debugger interface. The main window displays a disassembly of code for the address range 0x31c5eb to 0x31c7e6. The code includes instructions like TEST, JNE, RET, CMP, JE, and RET. A modal error dialog box is overlaid on the disassembly, titled "Tornado". The dialog contains the text "attach 0xf6bb94" and "Failed: cannot attach to task with VX_UNBREAKABLE option set". A "确定" (OK) button is visible at the bottom of the dialog.

Tornado - [Disassembly [0x31c5eb - 0x31c7e6]]

File Edit View Project Build Debug Tools Window Help

192.168.102.1@MICROSC

```
0x31c5eb +0x139: TEST EAX, EAX
0x31c5ed +0x141: JNE semQPut + 0x90
0x31c5ef +0x143: RET
0x31c5f0 +0x144: CMP EAX, 1
0x31c5f3 +0x147: JE semQPut + 0xb0
0x31c5f5 +0x149: RET
0x [ESI+0]
0x [EDI+0]
0x xit
0x 0xffffffff
0x [ESI+0]
0x 8]
0x 9cc, EAX
0x
0x31c61b +0x187: ADD ESP, 4
0x31c61e +0x190: MOV [ESP+8], EAX
0x31c622 +0x194: JMP semTake
0x31c627 +0x199: MOV ESI, ESI
0x31c629 +0x201: LEA EDI, [EDI+0]
0x31c630 semOTake: PUSH -1
0x31c632 +0x002: MOV ECX, [ESP+8]
0x31c634 +0x004: MOV
```

Tornado

attach 0xf6bb94

Failed: cannot attach to task with VX_UNBREAKABLE option set

确定

VXWORKS 5.5-调试问题

```
.text:003BE350 taskSpawn proc near ; CODE XREF: wdbSp+36f
.text:003BE350 ; sub_342D30+95f
.text:003BE350 ; ftpdInit+1BBf
.text:003BE350 ; muxPollStart+F5f
.text:003BE350 ; netLibInit+D1f
.text:003BE350 ; rpcInit:loc_351DE2f
.text:003BE350 ; telnetdIoTasksCreate+F4f
.text:003BE350 ; telnetdIoTasksCreate+174f
.text:003BE350 ; telnetdStart+163f
.text:003BE350 ; tFtpChildTaskSpawn+36f
.text:003BE350 ; tFtpdInit+A9f
.text:003BE350 ; tFtpdTask:loc_3566E2f
.text:003BE350 ; dcacheDevCreate+1FFf
.text:003BE350 ; excInit:loc_382B03f
.text:003BE350 ; logInit:loc_387C73f
.text:003BE350 ; shellInit+70f
.text:003BE350 ; sub_39A950+61f
.text:003BE350 ; taskRestart+A5f
.text:003BE350 ; DATA XREF: .data:standTbl
.text:003BE350 var_18 = dword ptr -18h
.text:003BE350 arg_0 = dword ptr 8
.text:003BE350 arg_4 = dword ptr 0Ch
.text:003BE350 arg_8 = dword ptr 10h
.text:003BE350 arg_C = dword ptr 14h
.text:003BE350 arg_10 = dword ptr 18h
.text:003BE350 arg_14 = dword ptr 1Ch
.text:003BE350 arg_18 = dword ptr 20h
.text:003BE350 arg_1C = dword ptr 24h
.text:003BE350 arg_20 = dword ptr 28h
.text:003BE350 arg_24 = dword ptr 2Ch
.text:003BE350 arg_28 = dword ptr 30h
.text:003BE350 arg_2C = dword ptr 34h
.text:003BE350 arg_30 = dword ptr 38h
.text:003BE350 arg_34 = dword ptr 3Ch
.text:003BE350 arg_38 = dword ptr 40h
.text:003BE350
.text:003BE350 push ebp
.text:003BE351 mov ebp, esp
.text:003BE353 sub esp, 14h
```

000B63B0 003BE350: taskSpawn

```
.text:00351DC7 push 0
.text:00351DC9 push 0
.text:00351DCB push offset portmapd
.text:00351DD0 push eax
.text:00351DD1 mov eax, portmapdOptions
.text:00351DD6 push eax
.text:00351DD7 mov eax, portmapdPriority
.text:00351DDC push eax
.text:00351DDD push offset gcc2_compiled__86
.text:00351DE2
.text:00351DE2 loc_351DE2: ; CODE XREF: .text:gcc2
.text:00351DE2 call taskSpawn
.text:00351DE7 add esp, 40h
.text:00351DEA cmp eax, 0FFFFFFFFh
.text:00351DED mov ds:portmapdId, eax
.text:00351DE2 iz short loc_351E23
```

VXWORKS 5.5-调试问题

```
003D6384  03 00 00 00 10 27 00 00 00 00 00 00 00 00 0
003D6394  00 00 00 00 00 00 00 00 00 00 00 00 00 00 0
003D63A4  00 00 00 00 00 00 00 00 00 00 00 00 00 00 0
003D63B4  00 00 00 00 FF 03 00 00 1E 00 00 00 00 36 0
003D63C4  03 00 00 00 10 27 00 00 00 00 00 00 00 00 0
003D63D4  00 00 00 00 00 00 00 00 00 00 00 00 00 00 0
003D63E4  F1 29 35 00 EA 29 35 00 E1 29 35 00 D8 2
003D63F4  CC 20 25 00 C1 20 25 00 B6 20 25 00 05 2
```

VXWORKS 5.5-调试问题

```
taskLib.h (h)
taskLib.h
#define VX_MAX_TASK_DELETE_RTNS 16 /* max task delete callout routines */
#define VX_MAX_TASK_CREATE_RTNS 16 /* max task create callout routines */

/* task option bits */

#define VX_SUPERVISOR_MODE 0x0001 /* OBSOLETE: tasks always in sup mode */
#define VX_UNBREAKABLE 0x0002 /* INTERNAL: breakpoints ignored */
#define VX_DEALLOC_STACK 0x0004 /* INTERNAL: deallocate stack */
#define VX_FP_TASK 0x0008 /* 1 = f-point coprocessor support */
#define VX_STDIO 0x0010 /* OBSOLETE: need not be set for stdio*/
...

003D6394 00 00 00 00 00 00 00 00 00 00 00 00 00
003D63A4 00 00 00 00 00 00 00 00 00 00 00 00 00
003D63B4 00 00 00 00 FF 03 00 00 1E 00 00 00 36
003D63C4 01 00 00 00 10 27 00 00 00 00 00 00 00
003D63D4 00 00 00 00 00 00 00 00 00 00 00 00 00
003D63E4 54 00 05 00 50 00 05 00 54 00 05 00 00
```

VXWORKS 5.5-调试问题

The screenshot shows the Tornado disassembler interface. The window title is "Tornado - [Disassembly [0x3ae95c - 0x3ae58]]". The menu bar includes File, Edit, View, Project, Build, Debug, Tools, Window, and Help. The toolbar contains various icons for file operations, navigation, and debugging. The address bar shows "192.168.102.1@MICROSC". The main display area shows assembly code with addresses, offsets, mnemonics, and operands.

```
0x3ae95c      +0x060:      MOV     EAX, [0x003d81d0+(EAXx4)]
0x3ae963      +0x067:      PUSH   EBX
0x3ae964      +0x068:      PUSH   ECX
0x3ae965      +0x069:      CALL   EAX
0x3ae967      +0x071:      JMP    _authenticate + 0x55
0x3ae969      +0x073:      LEA   ESI, [ESI+0]
0x3ae970      +0x080:      MOV   EAX, 0x2
0x3ae975      +0x085:      MOV   EBX, [EBP-24]
0x3ae978      +0x088:      MOV   ESP, EBP
0x3ae97a      +0x090:      POP   EBP
0x3ae97b      +0x091:      RET
0x3ae97c      +0x092:      LEA   ESI, [ESI+0]
0x3ae980      _svcauth_null:
0x3ae981      _svcauth_null:
0x3ae983      +0x003:      MOV   EBP, ESP
0x3ae985      +0x005:      MOV   ESP, EBP
0x3ae987      +0x007:      POP   EBP
0x3ae988      +0x008:      RET
0x3ae989      +0x009:      NOP
0x3ae98a      +0x010:      NOP
0x3ae98b      +0x011:      NOP
0x3ae98c      +0x012:      NOP
```

暴露在互联网中的VXWORKS WDB RPC V2服务!!!

WDB RPC的功能如此完备，就成了一把双刃剑。由于它本身没有身份认证的功能，因此能够与VxWorks主机17185端口通信就可以调用它。如果使用它的是黑客而非开发调试人员，就可能造成极大危害：

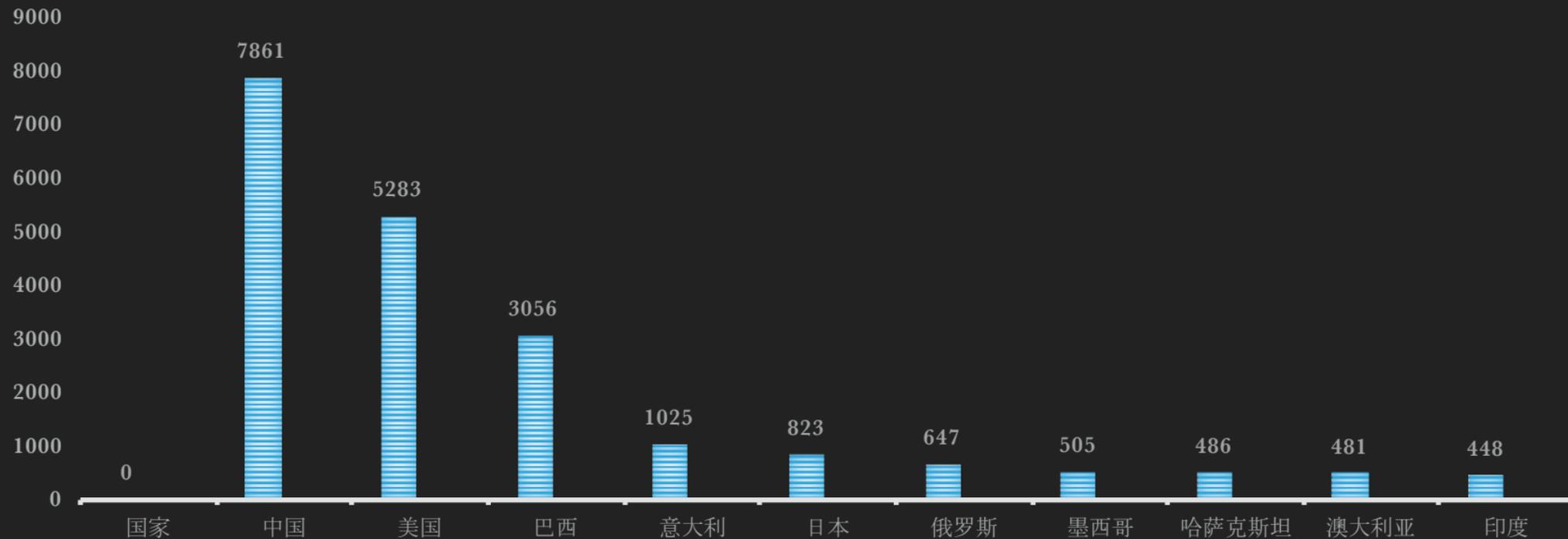
- * 监视所有组件（服务）状态
- * 恶意固件刷入、后门植入--探针
- * 重启VxWorks设备
- * 任意内存读写
- * 登陆绕过
- * ...

暴露在互联网中的VXWORKS WDB RPC V2服务!!!

Kimon在其 [揭秘VxWorks——直击物联网安全罩门](#) 一文中详尽地介绍了各种利用WDB RPC的攻击方式，因此不再一一列举。文中Kimon还给出了z-0ne 2015-11 关于WDB RPC的全球详细统计：34000台

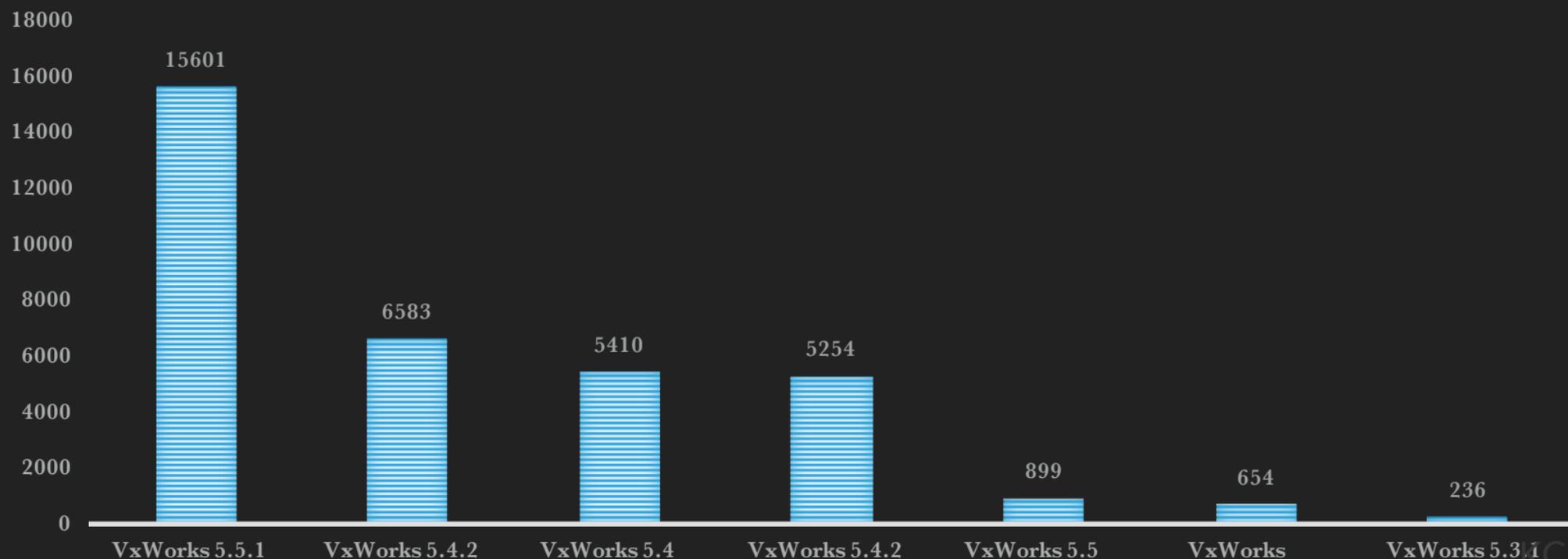
暴露在互联网中的VXWORKS WDB RPC V2服务!!!

TOP10 国家分布(V1)



暴露在互联网中的VXWORKS WDB RPC V2服务!!!

版本统计(V1)



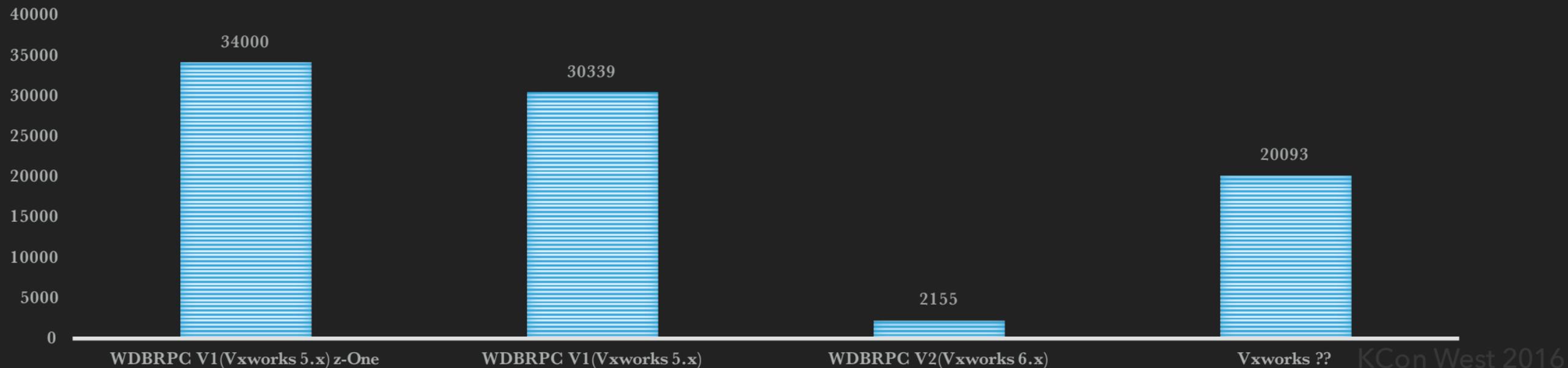
暴露在互联网中的VXWORKS WDB RPC V2服务!!!

其中受影响的PLC模块型号:

- a) 罗克韦尔Rockwell Automation 1756-ENBT固件版本为3.2.6、3.6.1及其他
- b) 西门子Siemens CP 1604、Siemens CP 1616
- c) 施耐德Schneider Electric 昆腾部分以太网模块

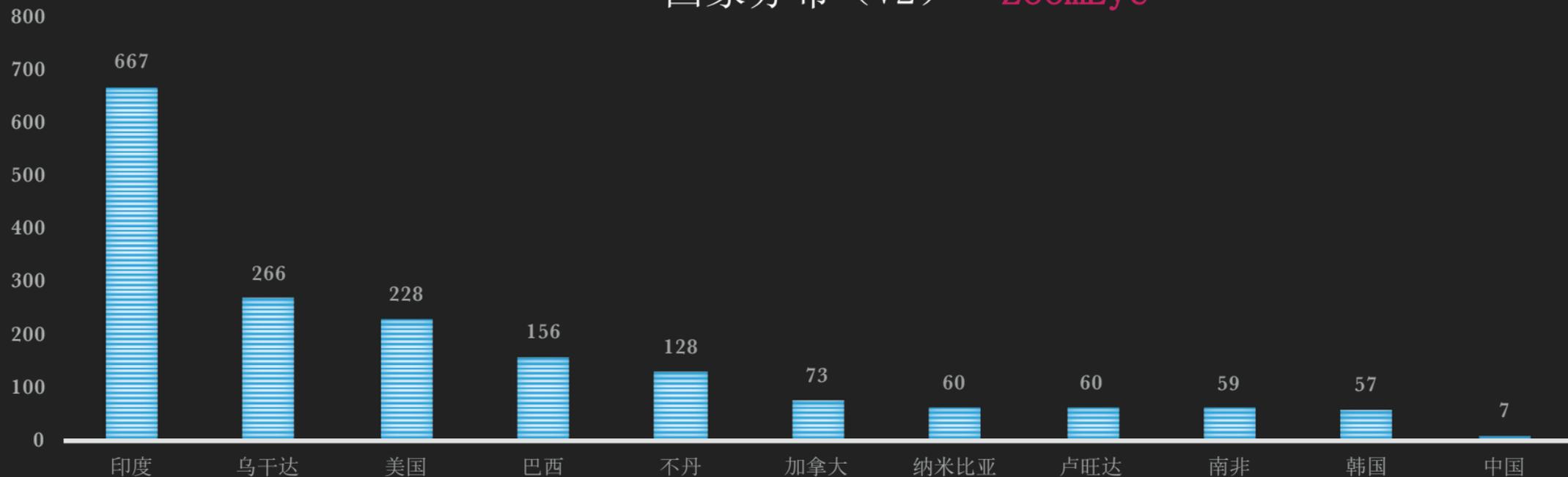
暴露在互联网中的VXWORKS WDB RPC V2服务!!!

ZoomEye团队探测全球IPv4网络空间结果：52586台Vxworks主机



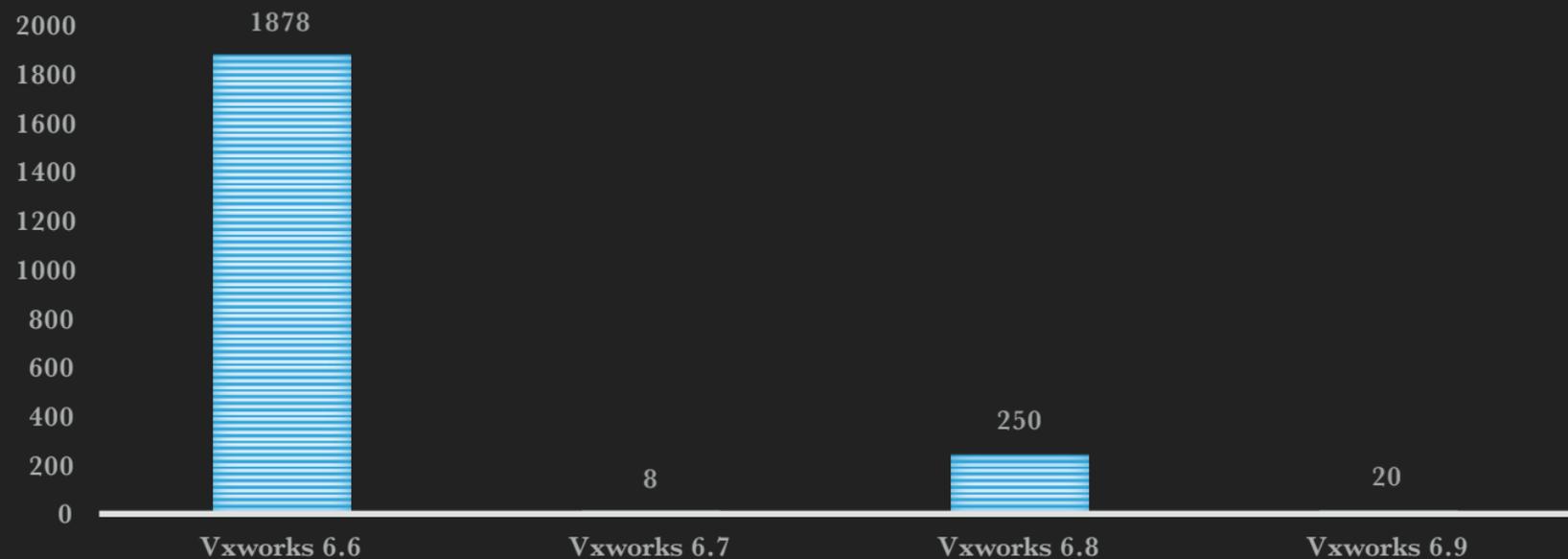
暴露在互联网中VXWORKS WDB RPC V2服务!!!

国家分布 (V2) --ZoomEye



暴露在互联网中VXWORKS WDB RPC V2服务!!!

版本统计 (V2) --ZoomEye



芯片/电路板 统计

Freescale P2020E - Security Engine	6	联网、电信、军事、工业
Freescale E300C3	6	网络、通信、工业控制
Intel(R) Pentium4 Processor SYMMETRIC IO MPTABLE	2	
IBM PowerPC [Fluke Odin] 405GPr Rev. 1.1	2	数码相机、调制解调器、机顶盒、手机、GPS、打印机、传真机、网卡、交换机、存储设备
RENESAS SH7751R 240MHz (BE)	2	路由器、PBX、LAN/WAN、打印机、扫描仪、PPC
Broadcom BCM91250A/swarm	2	Ethernet通信与交换
Xilinx Zynq-7000 ARMv7	2	高级驾驶员辅助系统、医疗内窥镜、小型蜂窝基带、专业照相机、机器视觉、电信级以太网回传、4K2K超高解析度电视、多功能打印机
BCM1190 A2	2	VoIP、宽带接入
Telvent HU_A ColdFire Board (MCF5485)	1	工业和嵌入式联网
RDL3000-SS - ARM11MPCore (ARM)	1	运载、SCADA、通信

利用WDB RPC V2，可以尝试进一步确定使用这些芯片或集成开发板的设备的品牌或型号，并对这些设备进行进一步控制，玩法与Kimon介绍的WDB RPC V1版本类似，有兴趣的同学可以继续深入。

总结

本次介绍了如何基于Fuzzing框架Sulley实现基于对VxWorks 5.5和6.6系统的FTP服务和Sun-RPC rpcbind服务的自动化Fuzzing，并介绍了在实现VxWorks 6.6自动化Fuzzing过程中必不可少的WDB RPC V2协议，最后对暴露在互联网中的WDB RPC V2协议进行了探测，并给出了相关统计。

可以看到，将WDB RPC服务暴露于互联网中的危险性极大，但它是使用VxWorks系统的硬件设备的系统开发人员不可或缺的工具，在开发过程中需要开启它，但在编译出厂设备的VxWorks系统时一定要将其关闭。

 米汤 
北京 海淀



扫一扫上面的二维码图案，加我微信

Thanks...