VXWORKS FUZZING 之道

VXWORKS 工控实时操作系统漏洞挖掘揭秘

KCon West 2016

我们

吴少荣　（404安全研究员）

TDG　（404安全研究员）

我们

本文内容

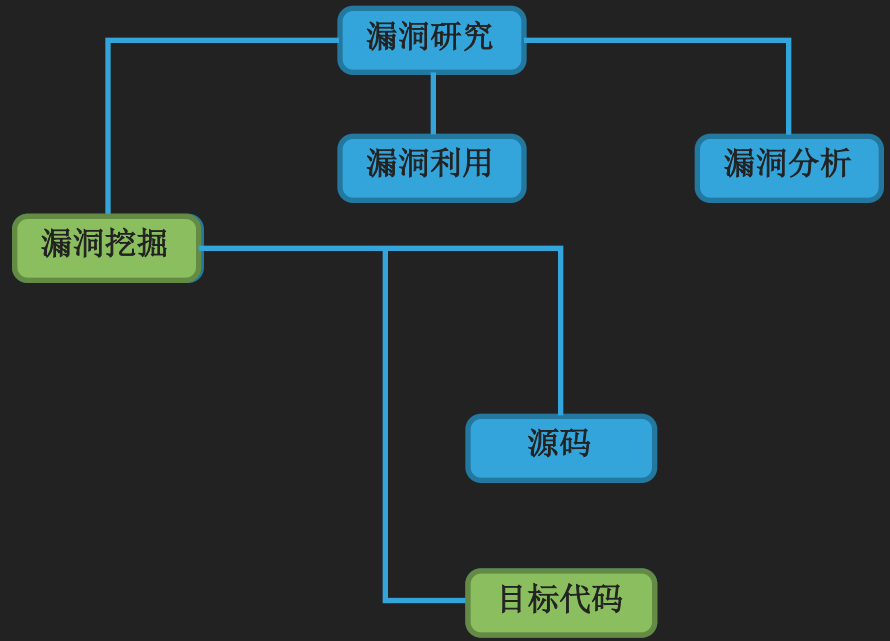
- 漏洞研究简介
- 研究环境准备
- 如何实现自动Fuzz（WDB、崩溃机制）
- 两个Fuzz实例
- Vxworks 系统组件无法调试问题
- 调查互联网上暴露的VxWorks WDB RPC服务
- 总结

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

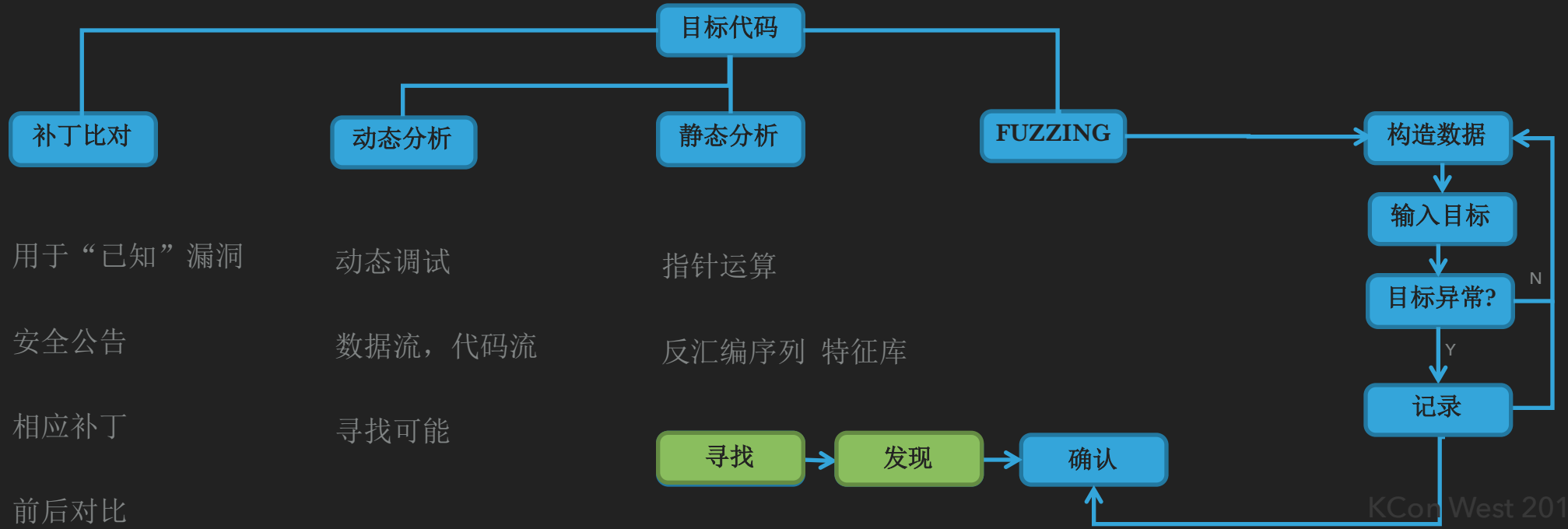


已发现漏洞的细节进行深入分析

漏洞利用

补救措施

漏洞研究-简介


目标代码
补丁比对
动态分析
静态分析
FUZZING
构造数据
输入目标
目标异常?
N
Y
记录
用于“已知”漏洞
安全公告
相应补丁
前后对比
动态调试
数据流，代码流
寻找可能
指针运算
反汇编序列 特征库
寻找
发现
确认
KCon West 2016

实现目标

环境准备

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的安全关键领域

1） 火星好奇心流浪者

2） 波音787梦幻客机

3） 网络路由器

特殊应用场景，高危性质使的VxWorks安全被高度关注。

环境准备

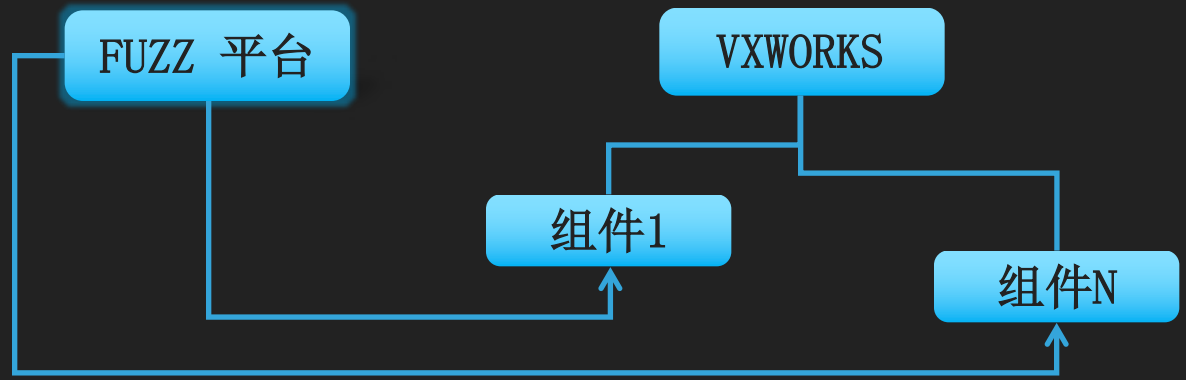
VXWORKS 安装

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
- ✓ 输入目标组件
- ✓ 检测组件状态
- ✓ 获取组件异常信息
- ✓ 记录信息
- ✓ 目标组件环境复原

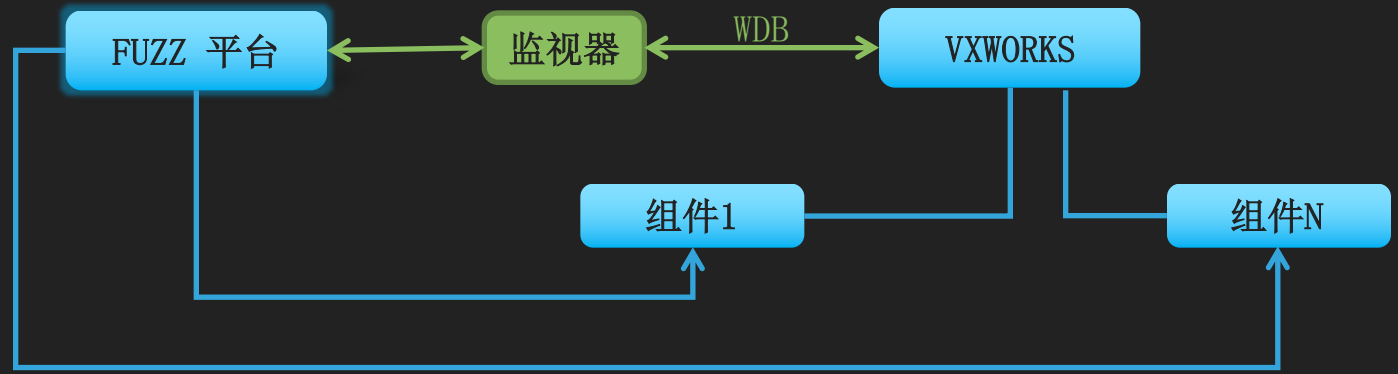
研究线索

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
- ✓ 输入目标组件
- ✓ 检测组件状态
- ✓ 获取组件异常信息
- ✓ 目标组件环境复原
- ✓ 记录信息

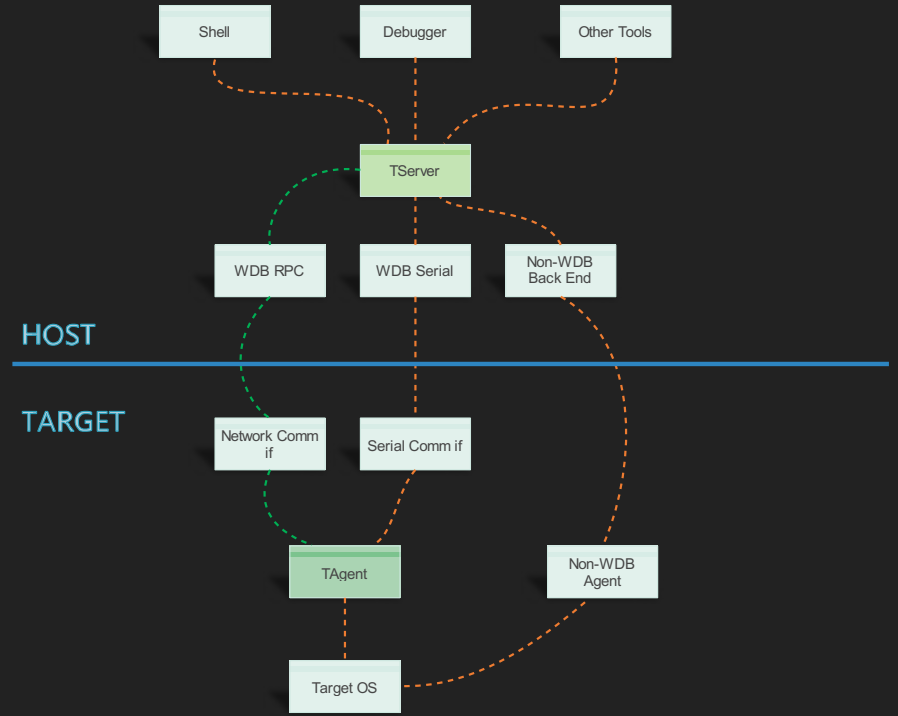
WDB RPC 协议

- 调试接口
- 基于SUN-RPC协议
- 服务运行在UDP协议的17185端口上
- WDB RPC被包含在VxWoks TAgent模块中
- 版本 V1，V2

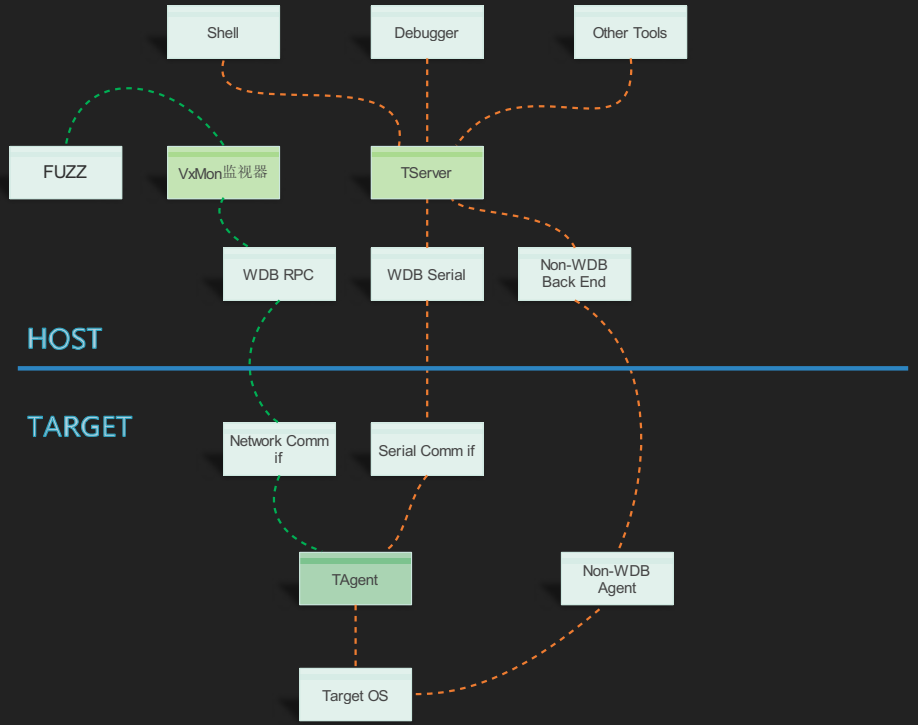
✓ 直接读写系统内存

✓ 感知系统组件状态

VXWORKS 调试通信框架



FUZZ框架思路



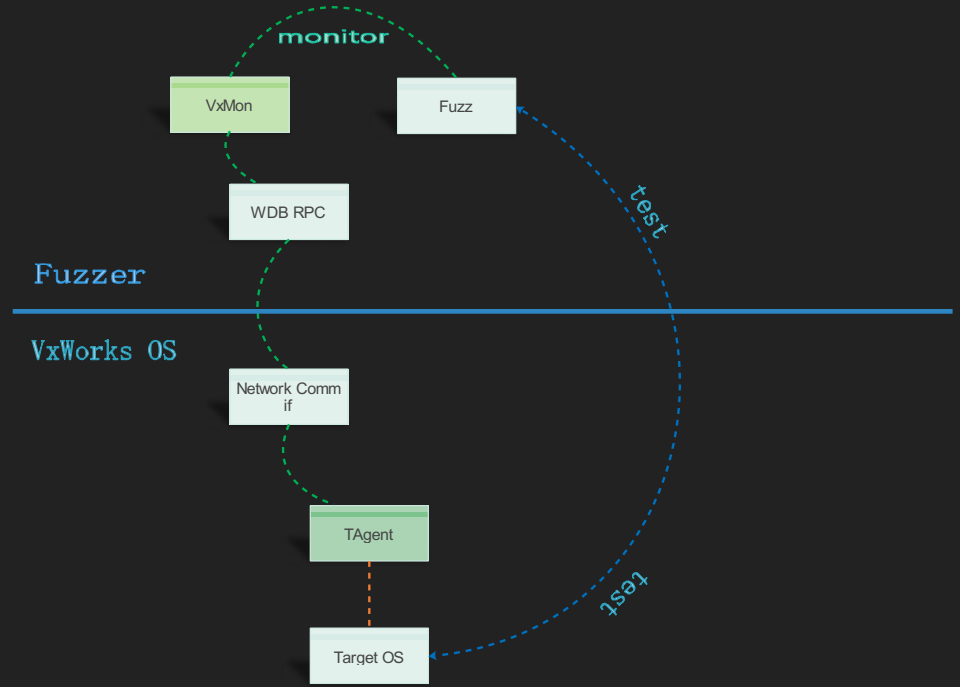
VxMon 充当调试器

模拟Debugger与TAgent通信

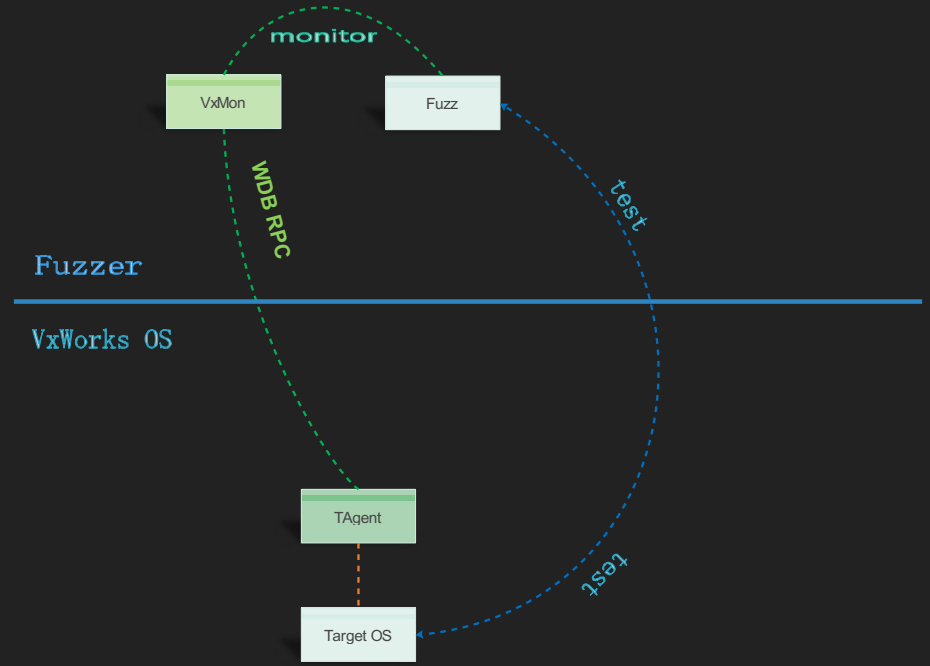
VxMon从TAgent获得异常通知

解决技术难点

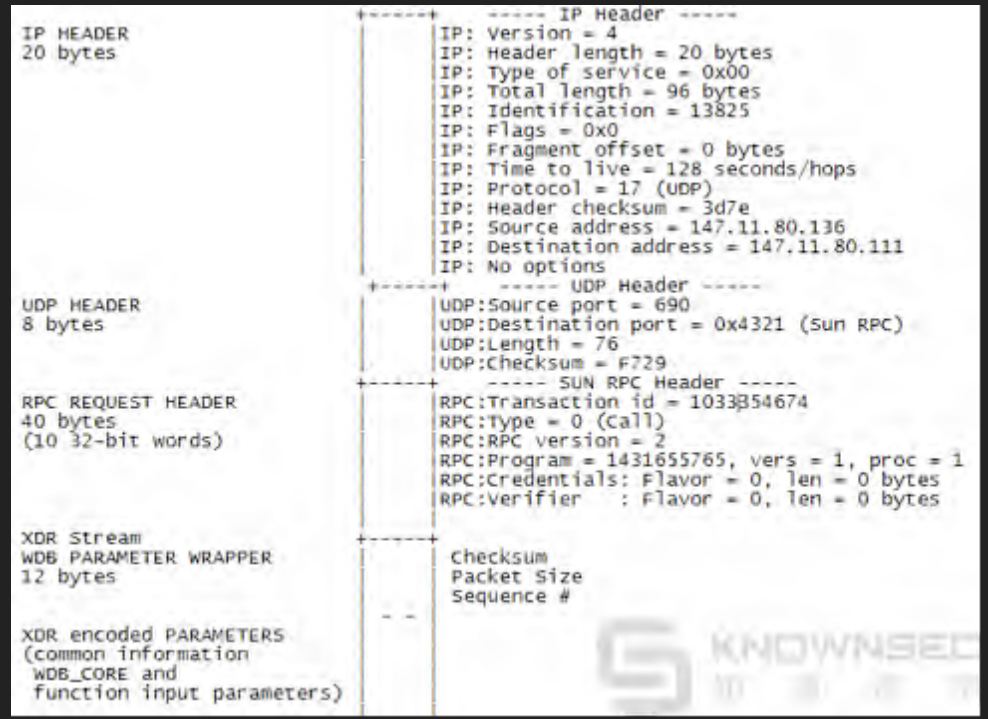
FUZZ框架


monitor
VxMon
Fuzz
WDB RPC
test
Fuzzer
VxWorks OS
Network Comm if
TAgent
Target OS
test

FUZZ框架


monitor
VxMon
Fuzz
WDB RPC
test
Fuzzer
VxWorks OS
TAgent
Target OS
test

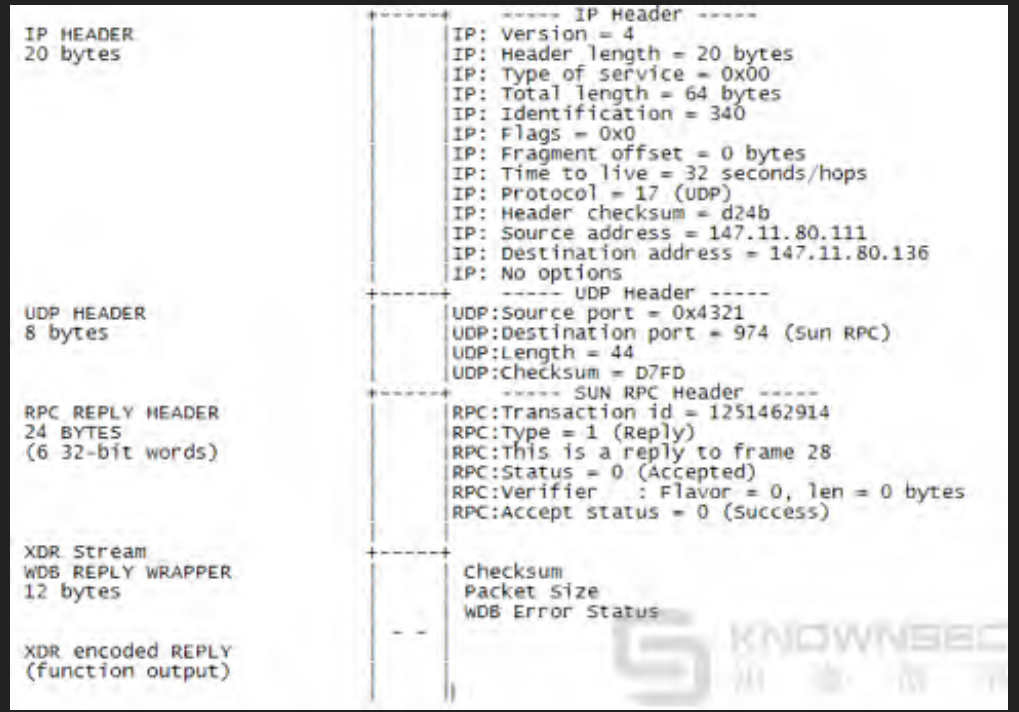
WDB PRC 请求数据包



* 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



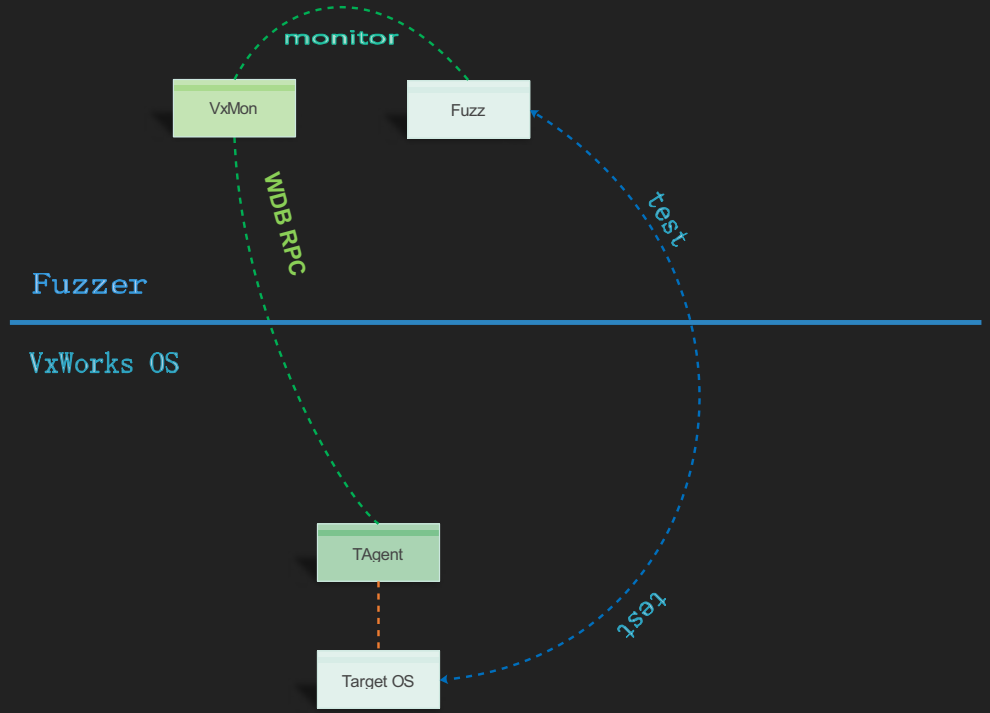
* IP Header
* UDP Header
* RPC Reply Header
* WDB Reply Wrapper
* Function output

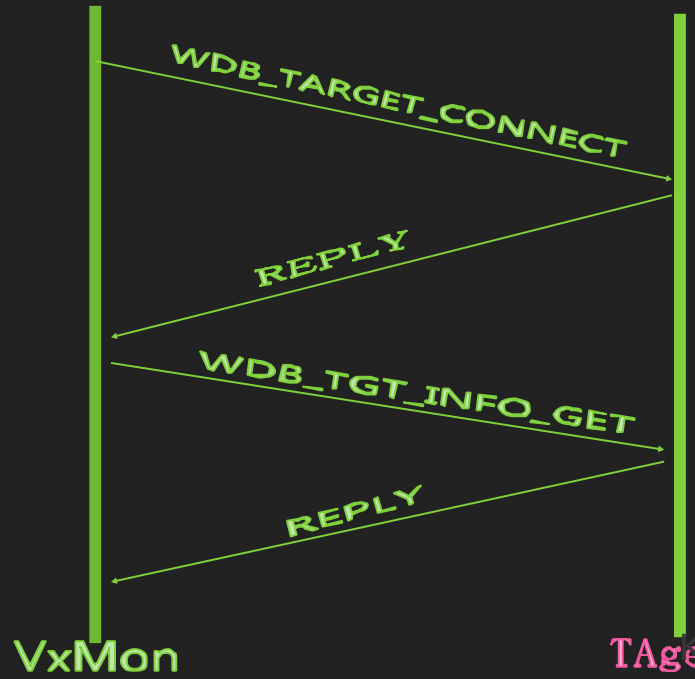
重点内容：WDB Parameter Wrapper内容包含整个请求包的大小、校验和及应答系列号（在每个请求与应答中，应答与请求系列号一致），Function output包含应答的输出信息，为请求功能号的返回信息。

VxMon 与 Vxworks 通信

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。

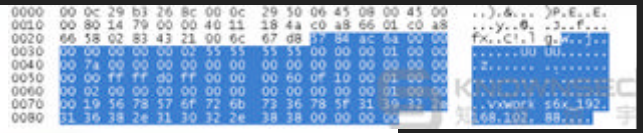
建立通信-举例

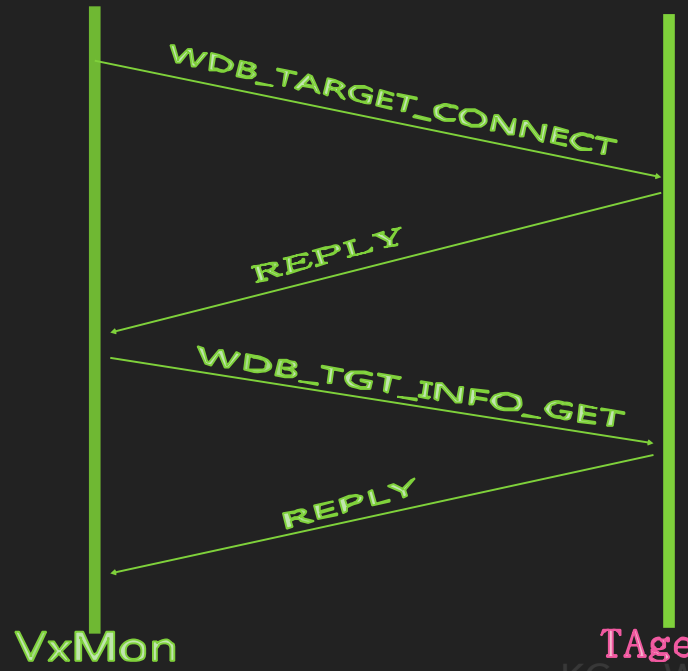

monitor
VxMon
Fuzz
WDB RPC
test
Fuzzer
VxWorks OS
TAgent
Target OS
test


WDB_TARGET_CONNECT
REPLY
WDB_TGT_INFO_GET
REPLY
VxMon
TAgent

建立通信-连接请求

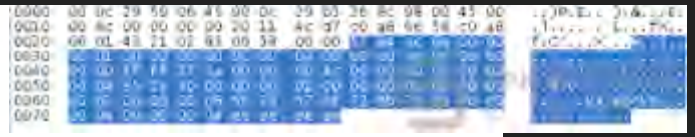
```
//WDB WDB_TARGET_CONNECT 请求包
5784ac6a          //Transaction ID
00000000          //Type is call
00000002          //RPC version
55555555          //Program
00000001          //ver
0000007a          //function id = WDB_TARGET_CONNECT( )

00000000
00000000
00000000
00000000

ffffd0ff          //checksum
00000060          //packet size
0f190001          //sequence

00000002
00000000
00000000

00000001          //Function input parameters
00000019          //length "Vxworks6x_192.168.102.88"
56785576f726b73367838f3139322e3136382e313030
```

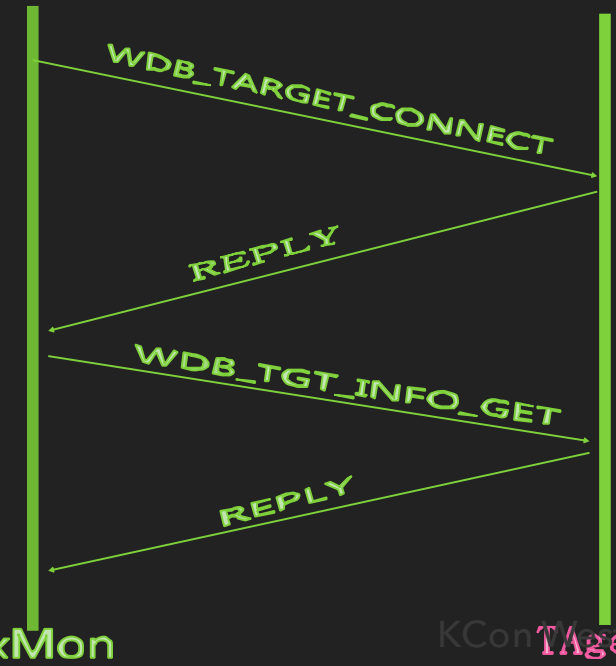




建立通信-连接应答

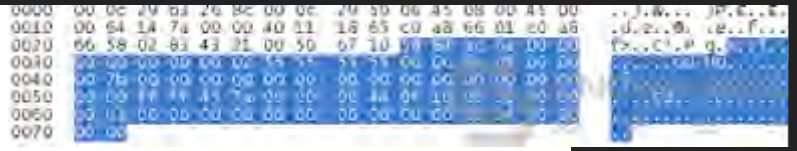
```
////WDB WDB_TARGET_CONNECT 应答包
5784ac6a          //Transaction ID
00000001          //Type is reply
00000000
00000000
00000000
00000000

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

00000004          //WDB_TGT_INFO
352e3000          //"5.0"
00000200
00000003
00000002
00000006          //length
56785761726b7300//"Vxworks"
00000004
00000004
eeeeeeee
```

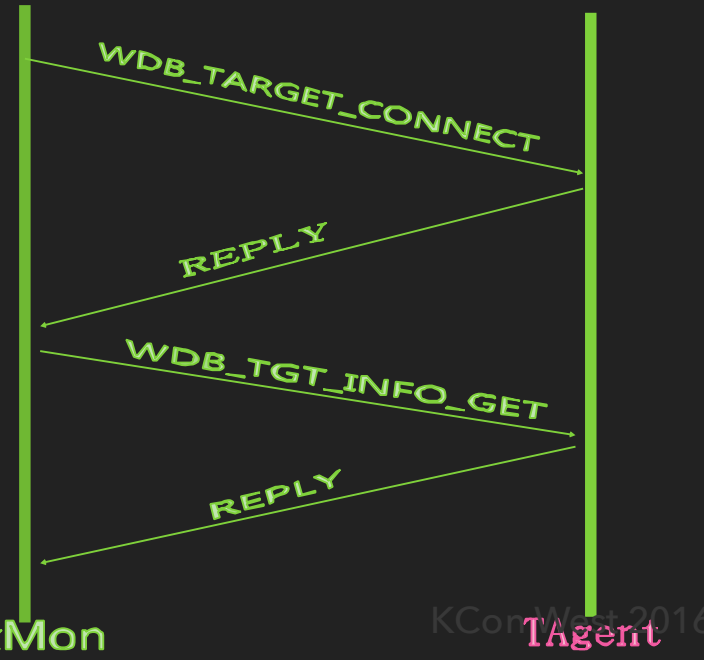




建立通信-获取信息请求

```
//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
```





建立通信-获取信息应答

在应答包中会含有Vxworks目标机很多信息，如：

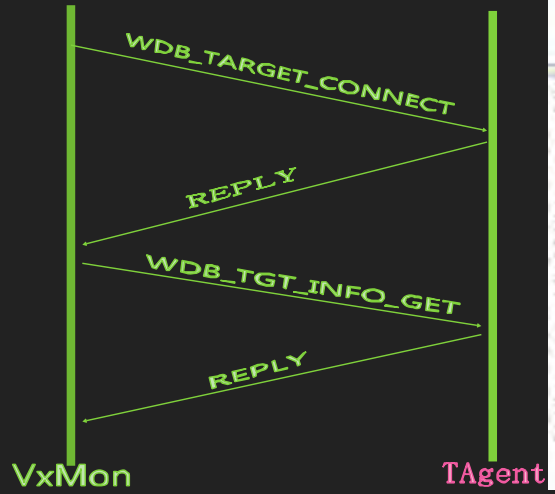
a) 系统版本

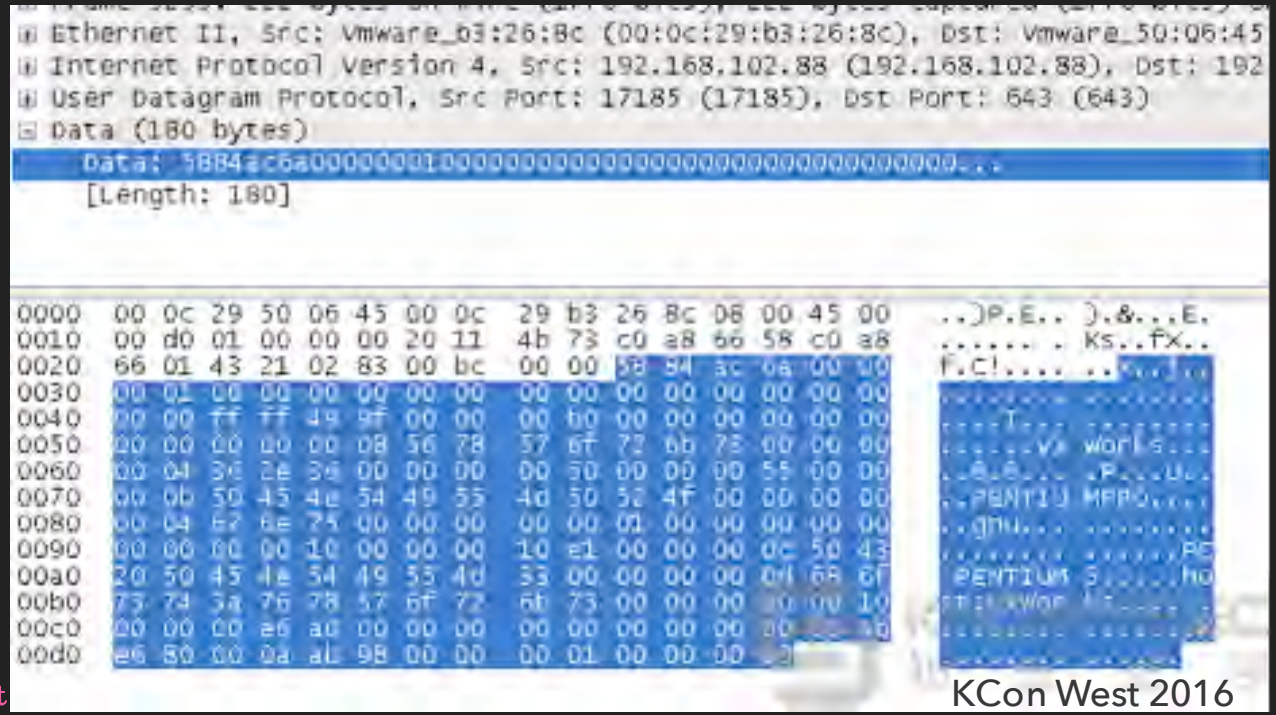
b) 大小端

c) 内存分配

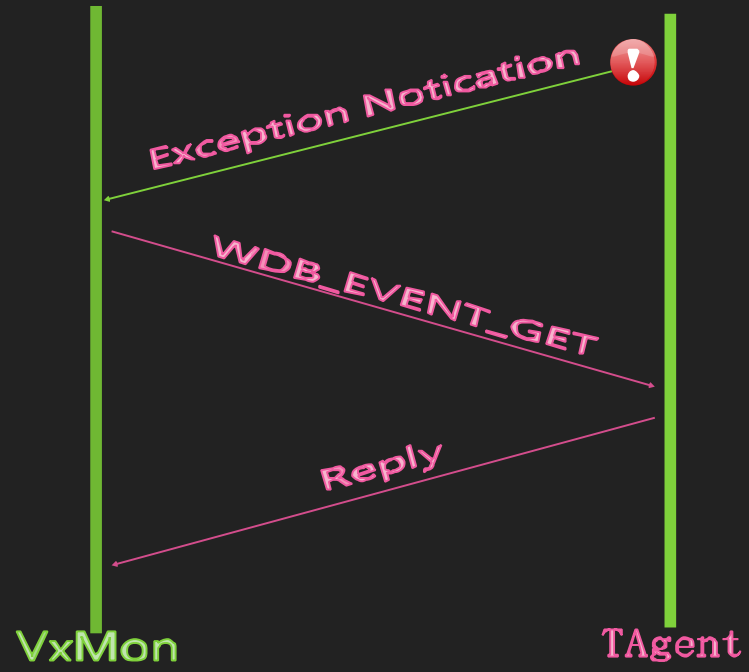
d) 硬件架构

e) 等等





崩溃机制检测

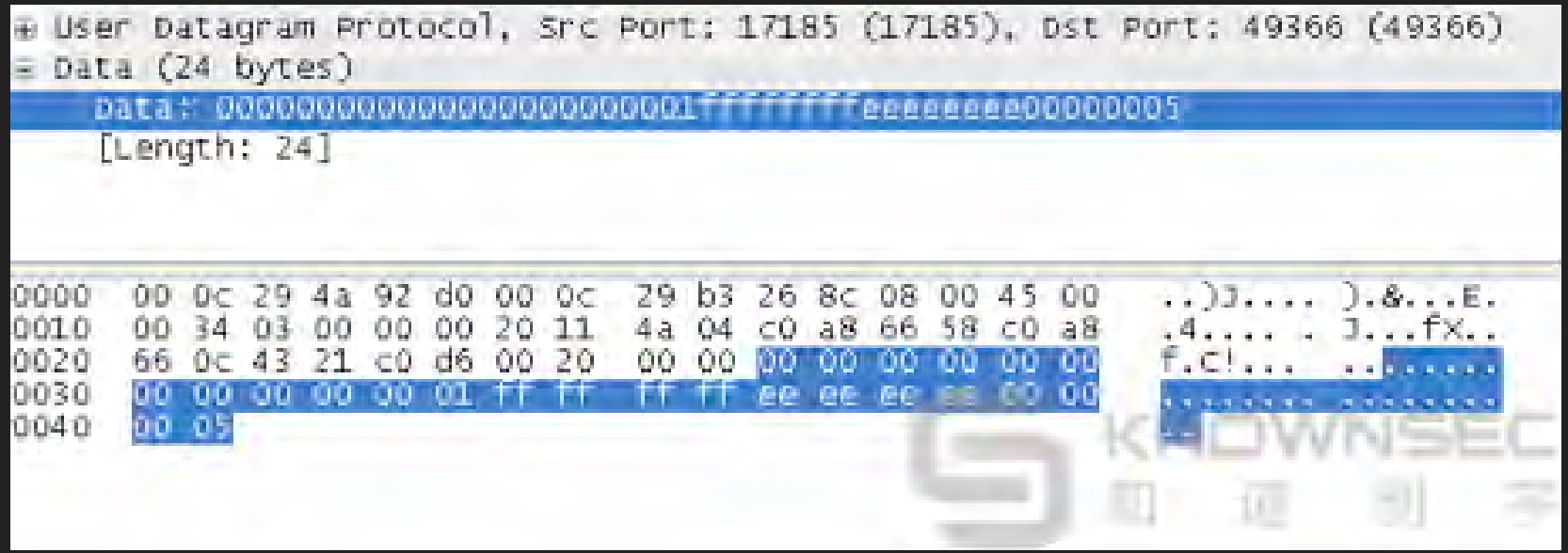


崩溃机制检测

特征：

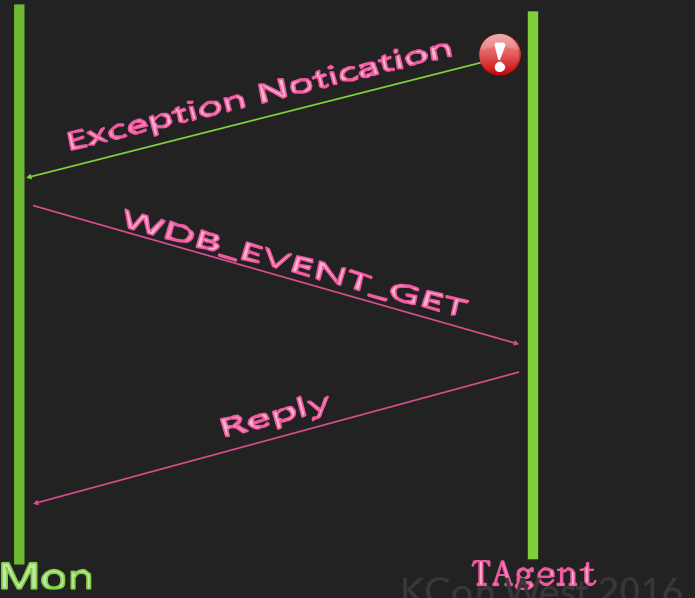
a） 循环通知

b） WDB_EVENT_GET包确认



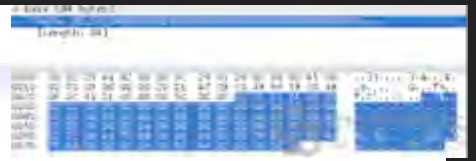
崩溃机制检测-异常信息请求

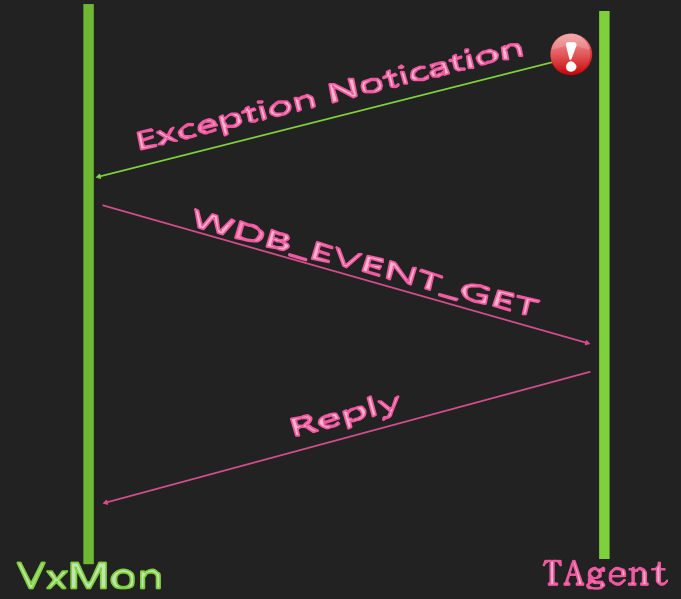
```
//WDB_EVENT_GET 请求包
11112224          //Transaction ID
00000000          //Type is call
00000002          //RPC version
55555555          //Program
00000001          //ver
00000046          //function id = WDB_EVENT_GET( )

00000000
00000000
00000000
00000000

00000000
00000030          //packet size
33339996          //sequence
```



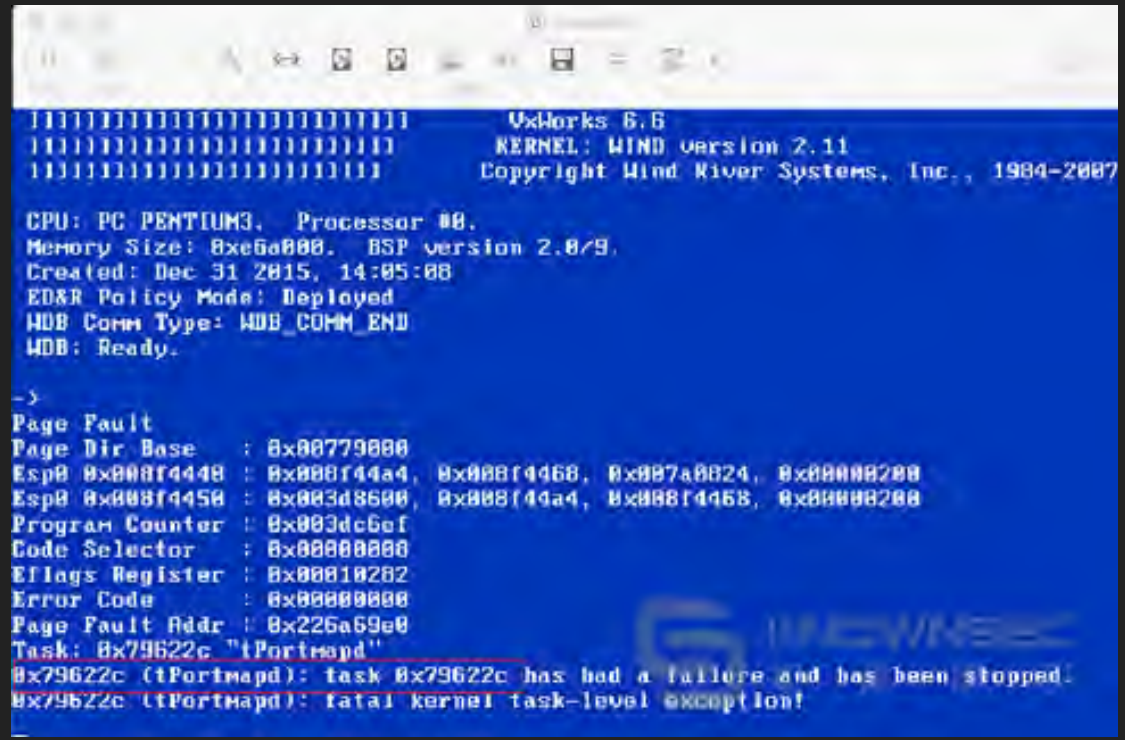
崩溃机制检测-异常信息应答

```
//WDB_EVENT_GET 应答包
11112224          //Transaction ID
00000001          //Type is reply
00000000
00000000
00000000
00000000

00000000
00000000
00000000

00000006          //event tpye = WDB_EVT_EXC
0000000a          //tructure length
00000002          //status of context
00000003          //context stopped by exception
0079622c          //task context
009a79b8
00000003          //context that got exception
0079622c
009a79b8
0000000e
005f4930          //address of exception stack frame
00000000
```





崩溃机制检测


VxWorks 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, 14:05:08
ED&R Policy Mode: Deployed
WDB Comm Type: WDB_COMM_END
WDB: Ready.
->
Page Fault
Page Dir Base : 0x00779000
Esp0 0x008f4448 : 0x008f44a4, 0x008f4468, 0x007a0824, 0x00000200
Esp0 0x008f4458 : 0x003d8600, 0x008f44a4, 0x008f4468, 0x00000200
Program Counter : 0x003dc6ef
Code Selector : 0x00000008
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!

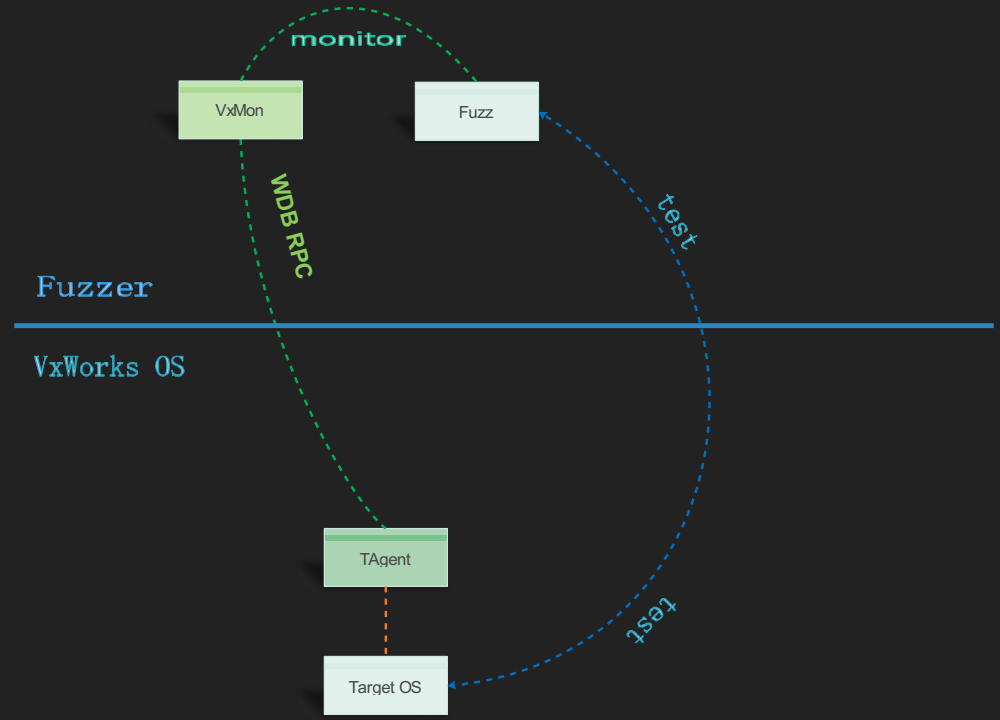
更多...

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，可以获取异常寄处器内容。

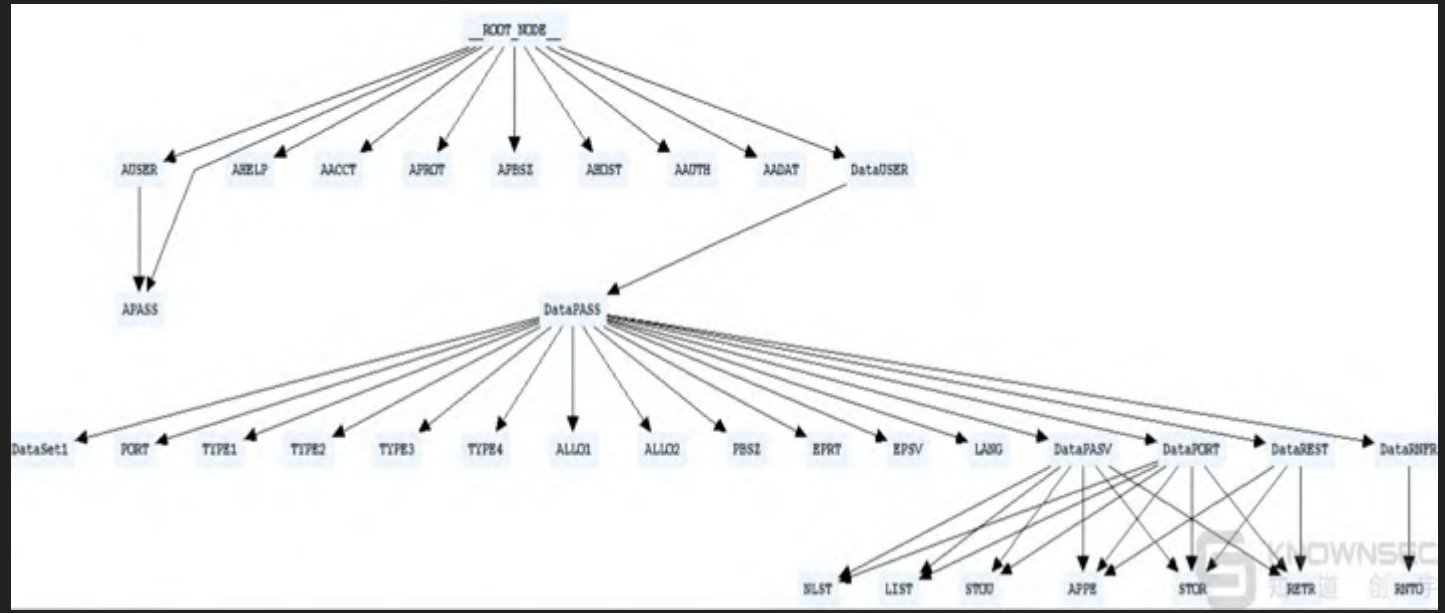
通过VxMon发送WDB_MEM_READ请求，可以获取异常地址的执行代码。

FUZZ框架


monitor
VxMon
Fuzz
WDB RPC
test
Fuzzer
VxWorks OS
TAgent
Target OS
test

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

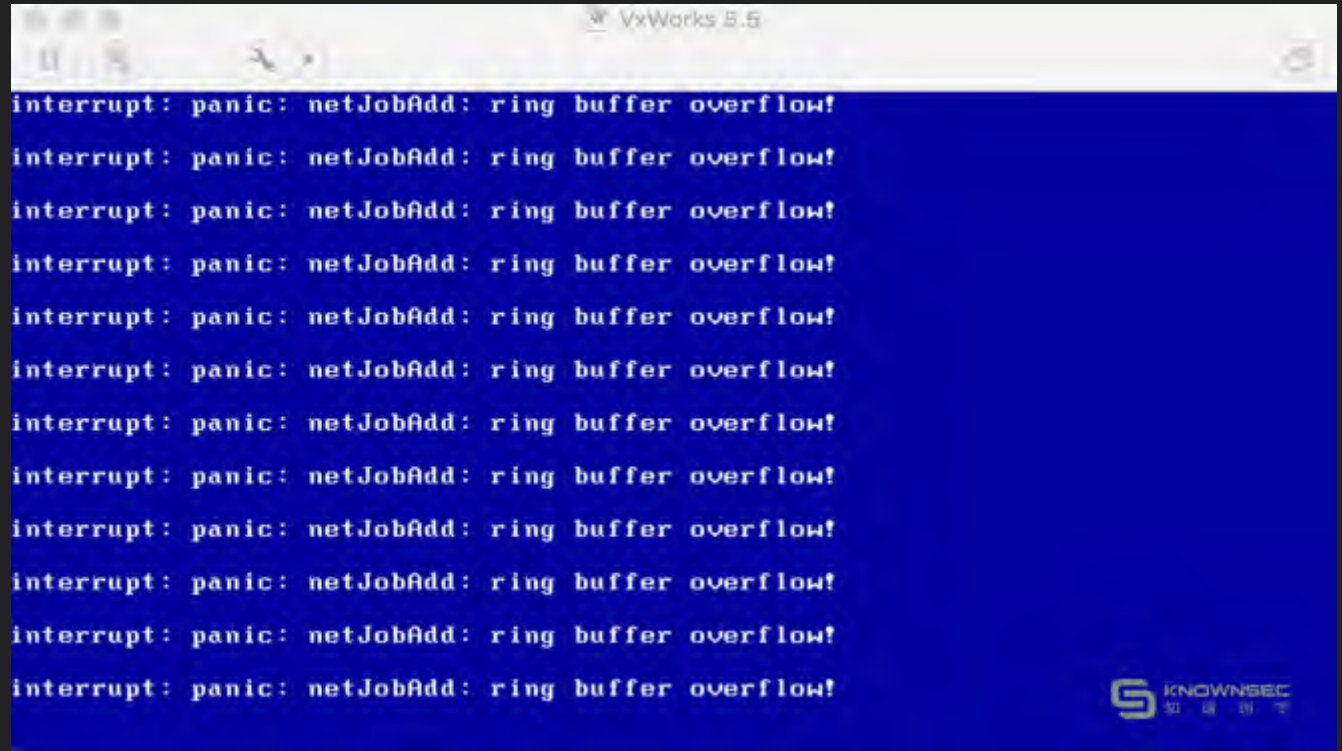
FTP协议中很多命令需要在登录后才能执行，我们主要关注未登录的情况。

fuzz结果：

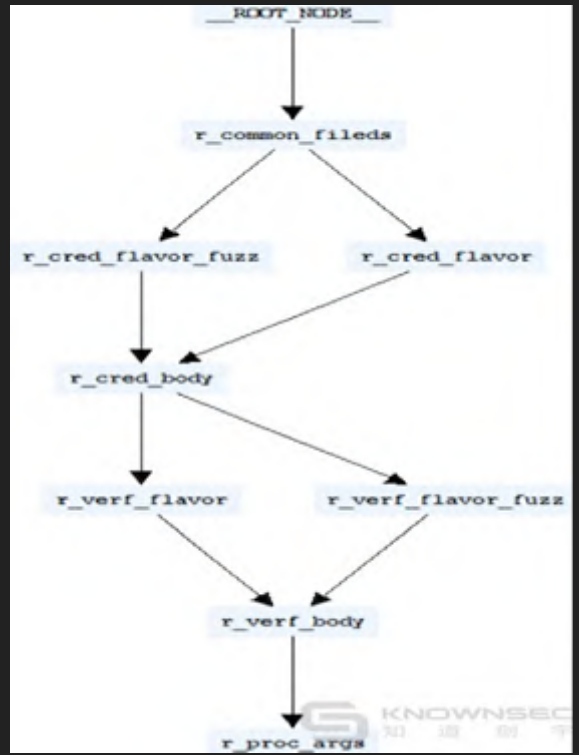
* 6.6版本无影响。

* 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


VxWorks 5.5
interrupt: panic: netJobAdd: ring buffer overflow!
interrupt: panic: netJobAdd: ring buffer overflow!
interrupt: panic: netJobAdd: ring buffer overflow!
interrupt: panic: netJobAdd: ring buffer overflow!
interrupt: panic: netJobAdd: ring buffer overflow!
interrupt: panic: netJobAdd: ring buffer overflow!
interrupt: panic: netJobAdd: ring buffer overflow!
interrupt: panic: netJobAdd: ring buffer overflow!
interrupt: panic: netJobAdd: ring buffer overflow!
interrupt: panic: netJobAdd: ring buffer overflow!
interrupt: panic: netJobAdd: ring buffer overflow!
interrupt: panic: netJobAdd: ring buffer overflow!
interrupt: panic: netJobAdd: ring buffer overflow!
KNOWNSEC

SUNRPC协议 RPCBIND服务 FUZZING



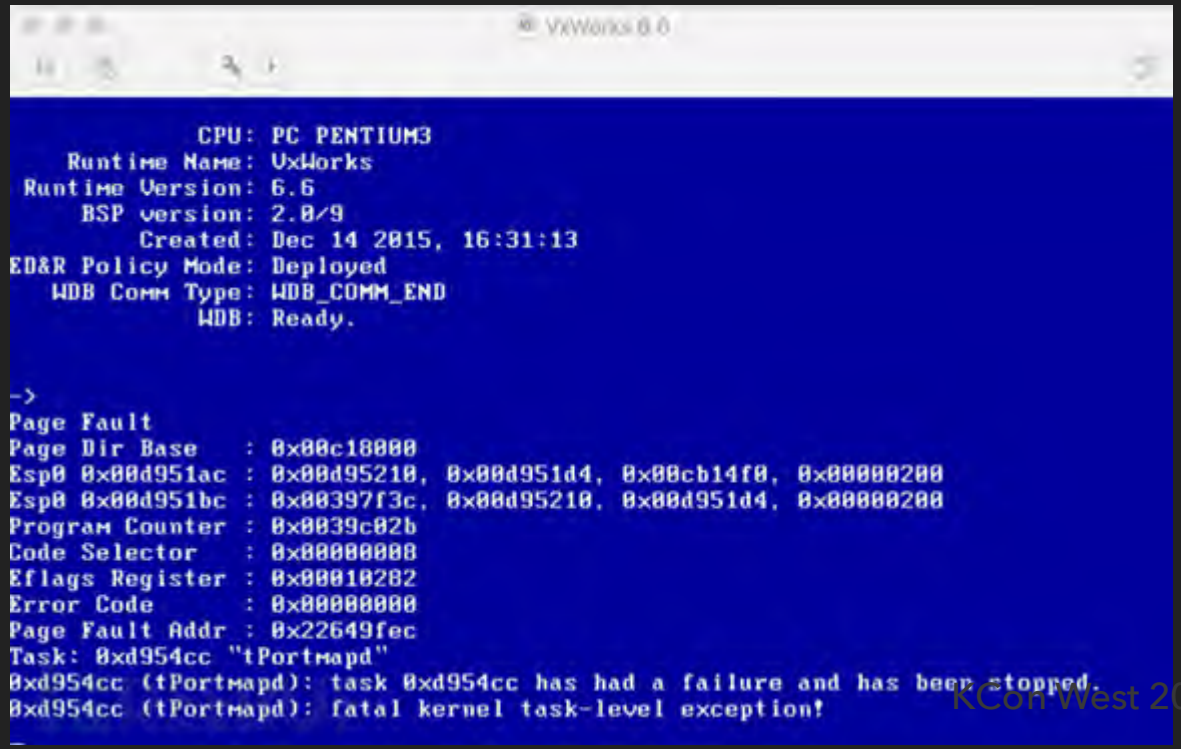
SUNRPC协议 RPCBIND服务 FUZZING

Fuzzing结果:

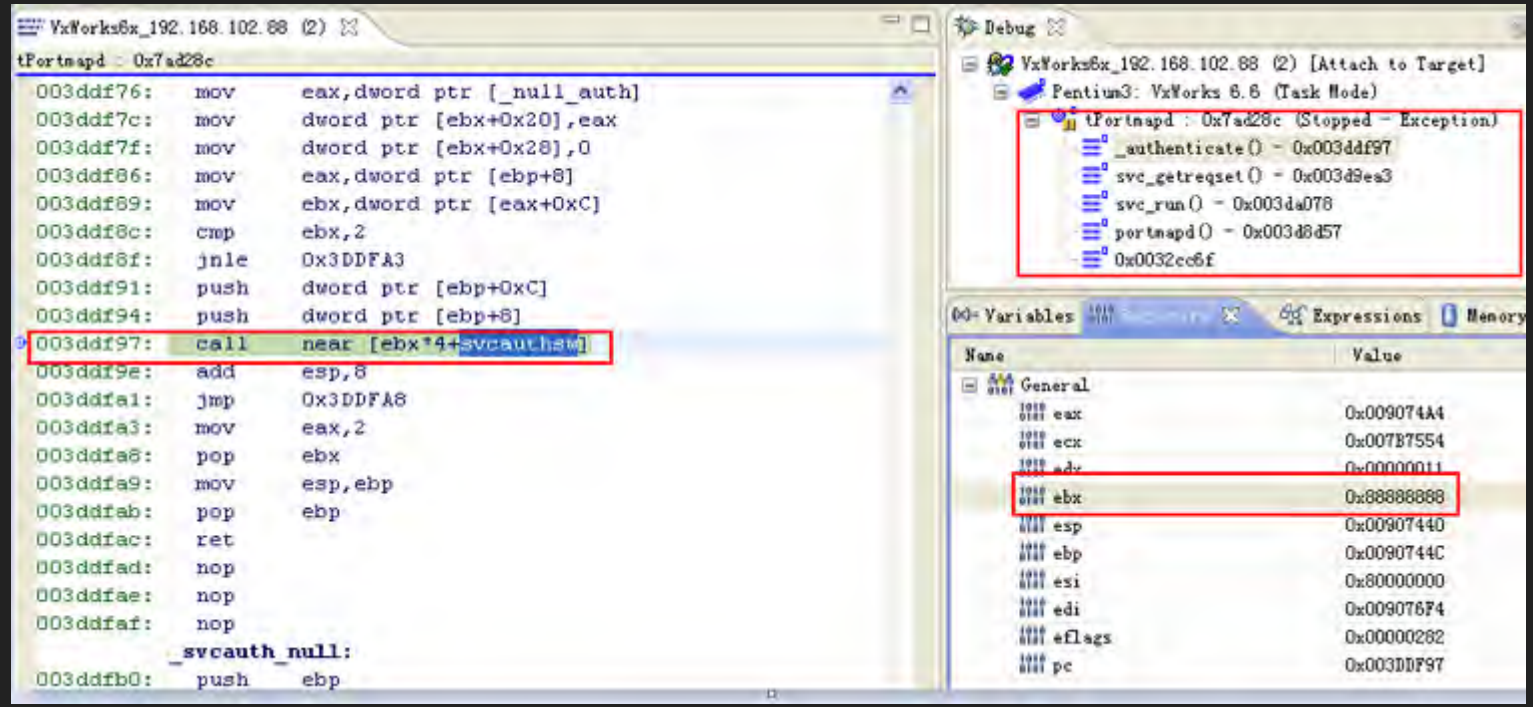
5.5及6.6版本均测试出18处崩溃点，（Payload 存在Github）通过观察结果中的寄存器状态，都属于一类，该漏洞仅造成tPortmapd服务崩溃，对其他服务没有影响。

RPCBIND 服务问题

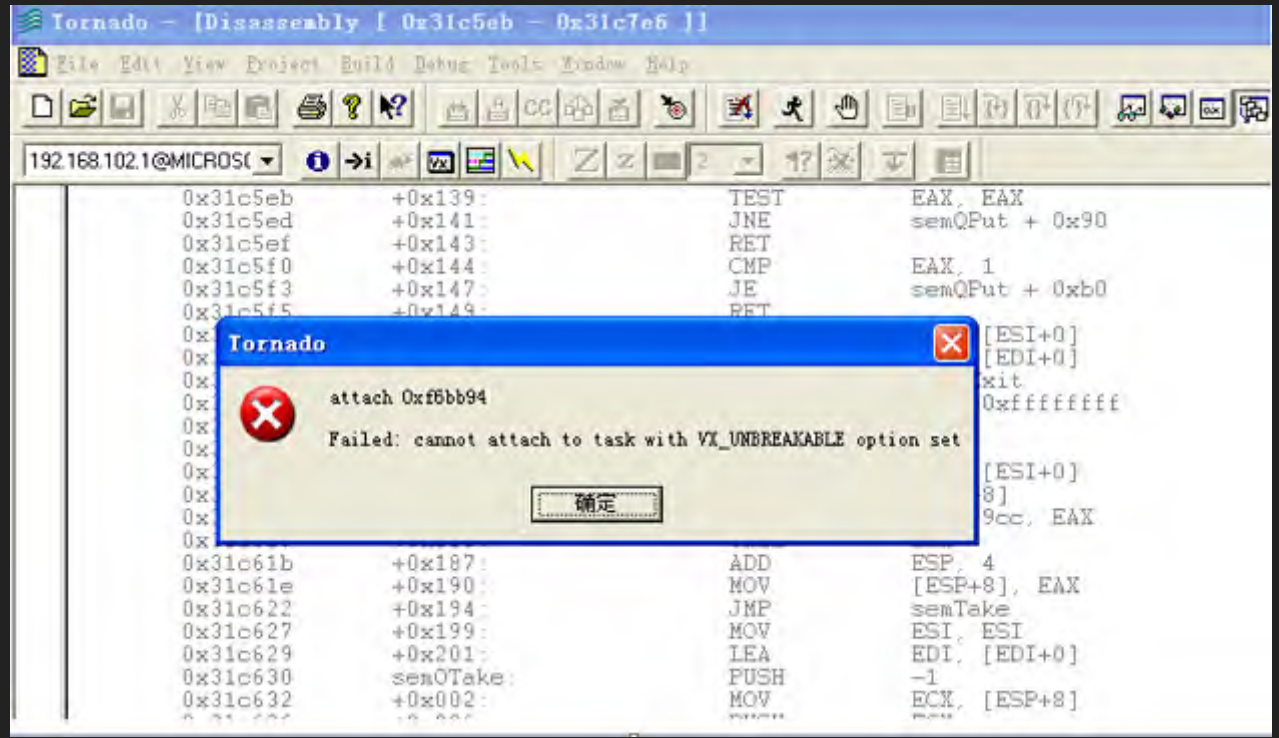
rpcbind服务是SUN-RPC的一部分，在VxWorks系统中该服务监听在tcp/111及udp/111端口，攻击者向该端口发送经过特殊构造的数据包，可使rpcbind服务崩溃，精心构造的请求可能可以造成任意代码执行。终端会给出错误信息，报错信息如下图：



VXWORKS 6.6-调试



VXWORKS 5.5-调试问题

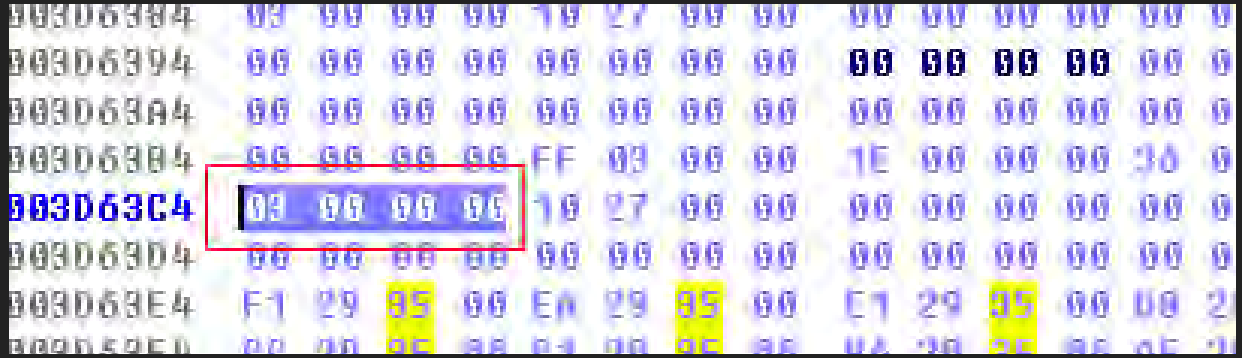


VXWORKS 5.5-调试问题

```
.text:003BE350 taskSpawn       proc near               ; CODE XREF: wdbSp+36↑p
.text:003BE350                                         ; sub_3A2D30+95↑p
.text:003BE350                                         ; ftpdInit+108↑p
.text:003BE350                                         ; muxPollStart+F5↑p
.text:003BE350                                         ; netLibInit+91↑p
.text:003BE350                                         ; rpcInit:loc_351DE2↑p
.text:003BE350                                         ; telnetdIoTasksCreate+74↑p
.text:003BE350                                         ; telnetdIoTasksCreate+174↑p
.text:003BE350                                         ; telnetdStart+163↑p
.text:003BE350                                         ; tFtpChildTaskSpawn+36↑p
.text:003BE350                                         ; tFtpdInit+A9↑p
.text:003BE350                                         ; tFtpdTask:loc_3566E2↑p
.text:003BE350                                         ; dcacheDevCreate+1FF↑p
.text:003BE350                                         ; excInit:loc_382B03↑p
.text:003BE350                                         ; lugInit:loc_387C73↑p
.text:003BE350                                         ; shellInit+70↑p
.text:003BE350                                         ; sub_39A950+61↑p
.text:003BE350                                         ; taskRestart+A5↑p
.text:003BE350                                         ; DATA XREF: .data:standTbl↑o
.text:003BE350
.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
000863B0 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:00351DF2                 jz      short loc_351E23
```

VXWORKS 5.5-调试问题



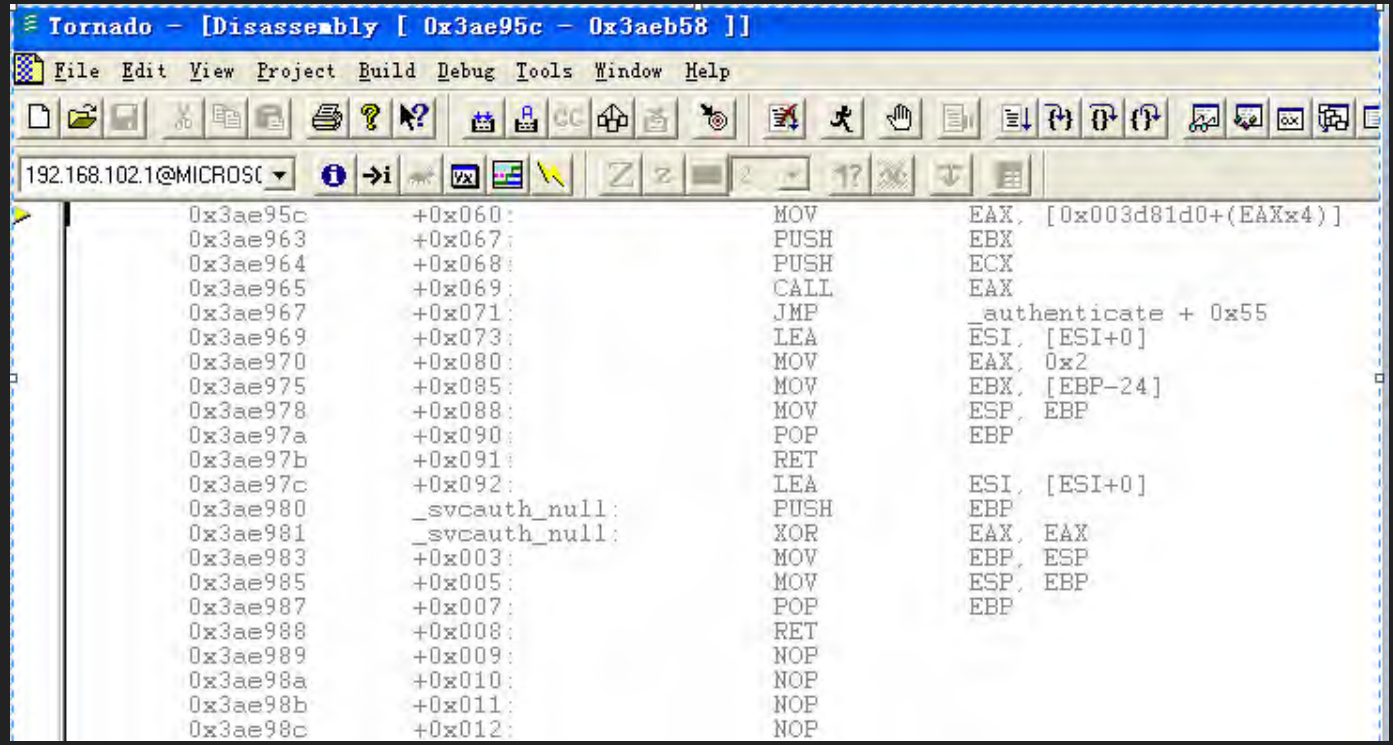
VXWORKS 5.5-调试问题

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
```

VXWORKS 5.5-调试问题



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

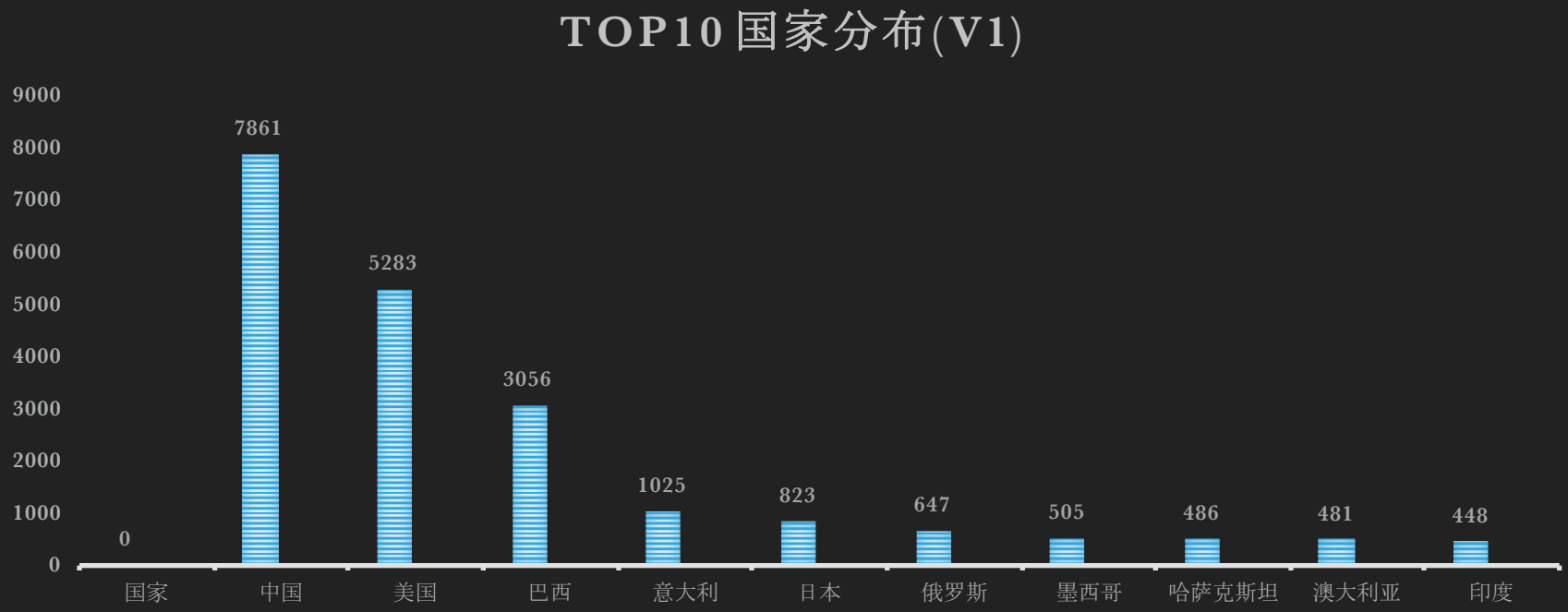
* 登陆绕过

* ...

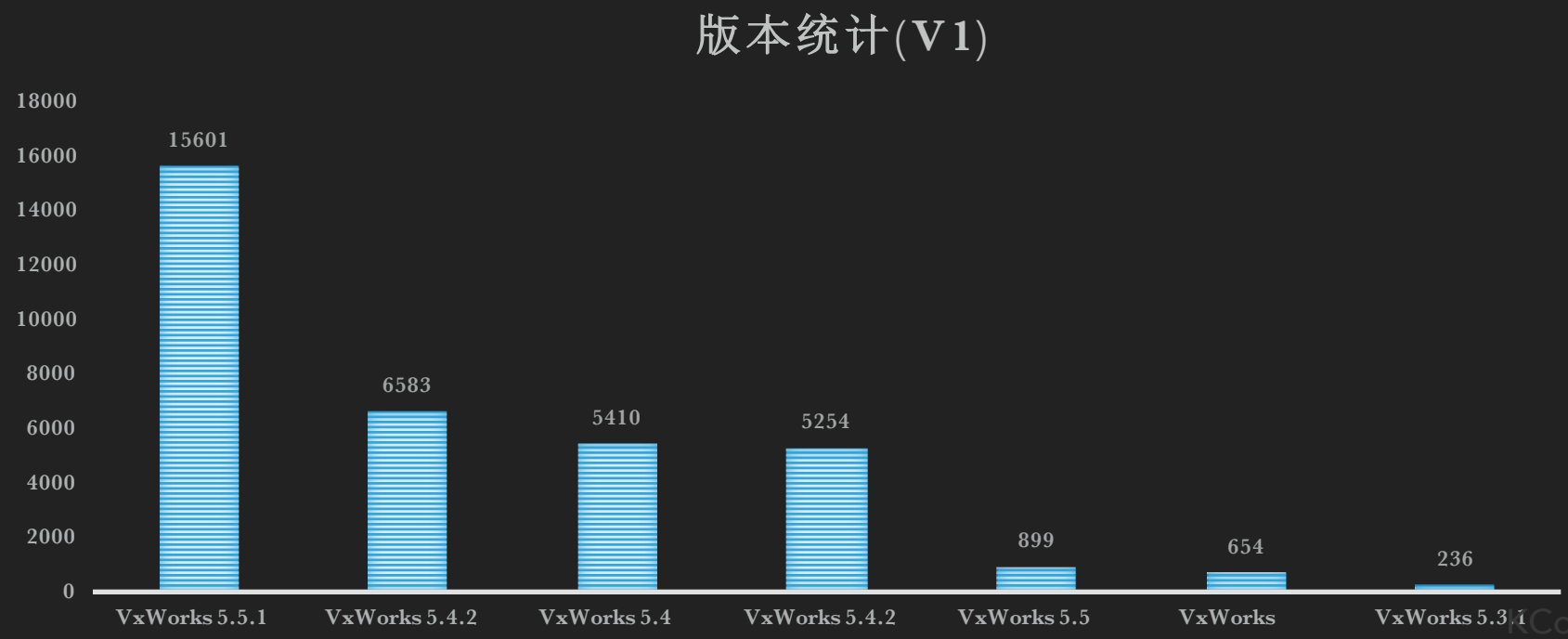
暴露在互联网中的VXWORKS WDB RPC V2服务!!!

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-One 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!!!



暴露在互联网中的VXWORKS WDB RPC V2服务!!!



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：

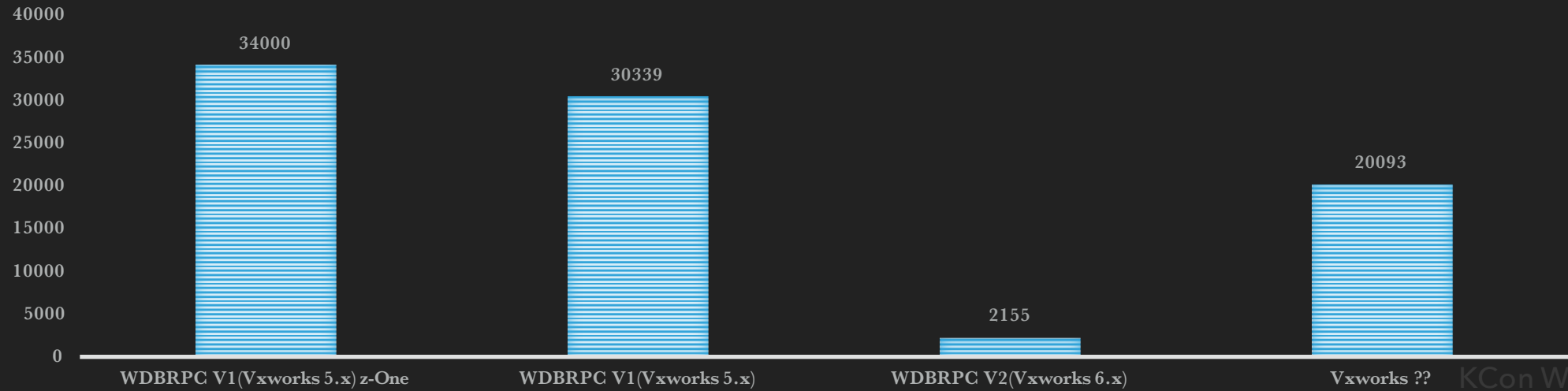
a）罗克韦尔Rockwell Automation 1756-ENBT固件版本为3.2.6、3.6.1及其他

b）西门子Siemens CP 1604、Siemens CP 1616

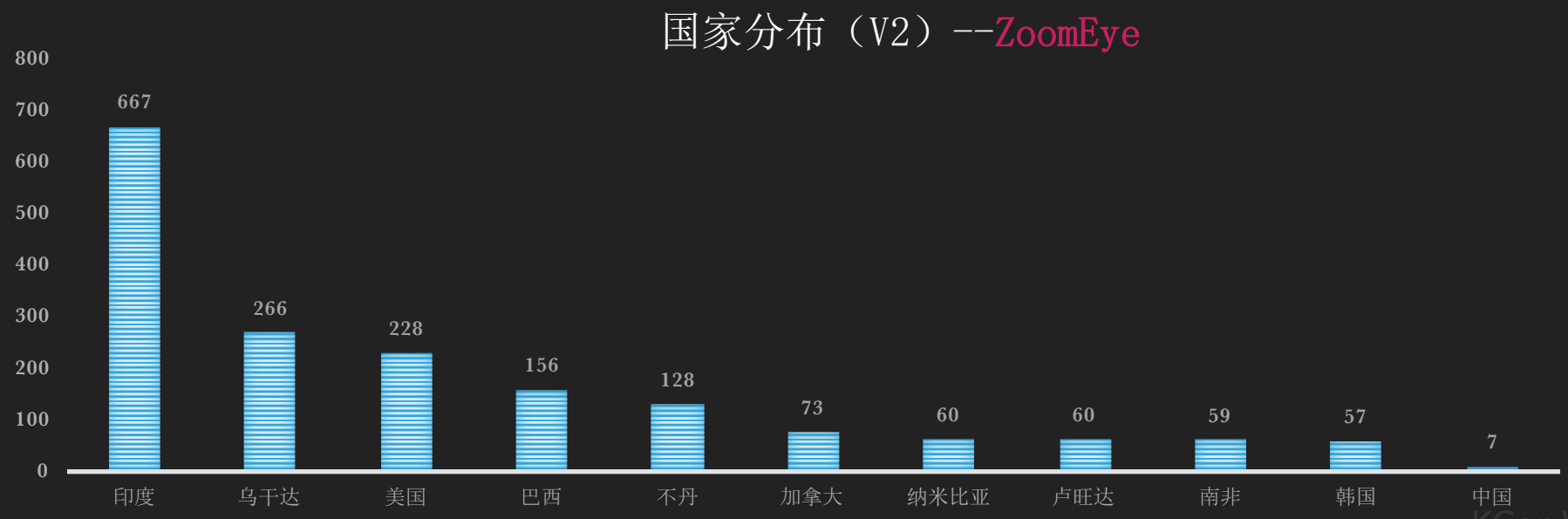
c）施耐德Schneider Electric 昆腾部分以太网模块

暴露在互联网中的VXWORKS WDB RPC V2服务!!!

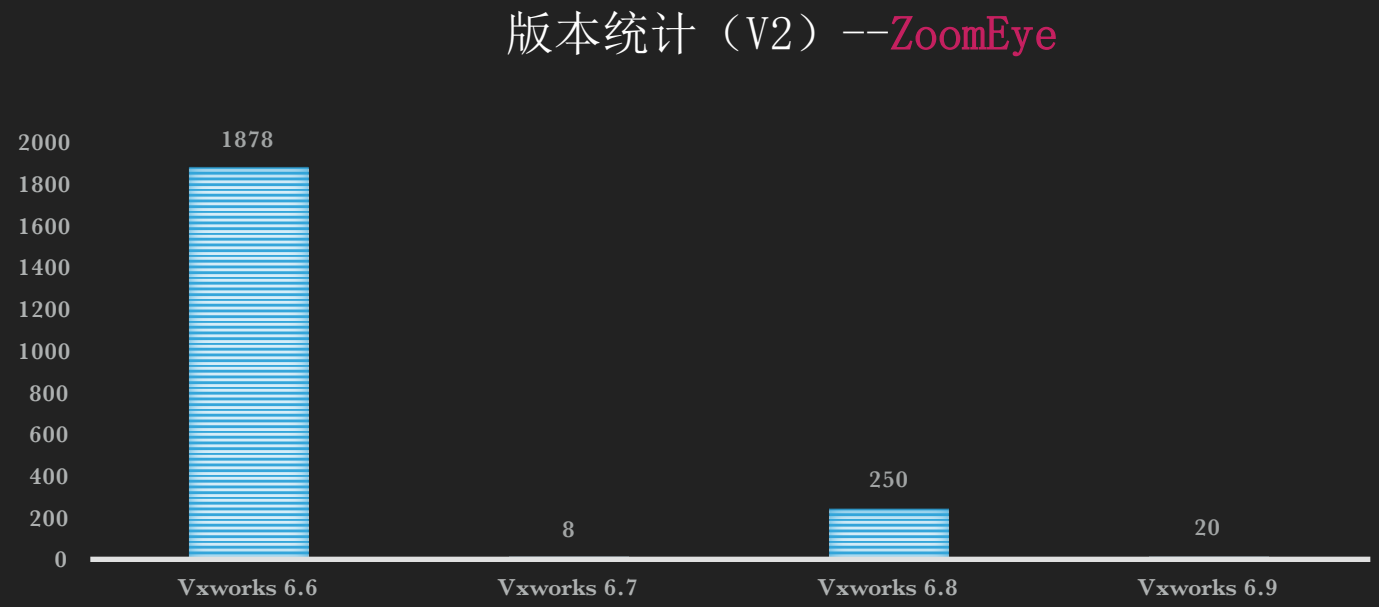
ZoomEye团队探测全球IPv4网络空间结果：52586台Vxworks主机



暴露在互联网中VXWORKS WDB RPC V2服务!!!



暴露在互联网中VXWORKS WDB RPC V2服务!!!



芯片/电路板 统计

Freescale P2020E - Security Engine	6	联网、电信、军事、工业
Freescale E600C3	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 HV_A ColdFire Board (MCF5485)	1	工业和嵌入式联网
RDL3000-SS - ARM11MPCore (ARM)	1	远载、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。

Thanks...