Digging Inside the VxWorks OS and Firmware
The Holistic Security

Aditya K Sood (0kn0ck)
SecNiche Security Labs (http://www.secniche.org)
Email: adi_ks [at] secniche.org

Contents

1 Acknowledgement

I sincerely thank my friends and security researchers for sharing thoughts over the VxWorks.

- Jeremy Collake (Bitsum)
- Edgar Barbosa (COSEINC)
- Luciano Natafrancesco (Netifera)

In addition, I would also like to thank HD Moore for finding vulnerabilities in VxWorks OS. A sincere gratitude to all the researchers who are engaged in constructive research for the security community. I am also grateful to Dr. Richard J Enbody for supporting me in work related to computer security.

Lastly, I sincerely want to thank CIGTIAL Inc. and my security group at SecNiche Security Labs for supporting me in doing continuous security research. Nonetheless, all the conference organizers who invited me to deliver talks and have shown faith in me.

2 Abstract

VxWorks is one of the most widely accepted embedded OS. In this paper, we have conducted a detailed study of the security model of VxWorks OS and firmware in order to understand the potential impact of security vulnerabilities on its functioning.

3 Introduction

VxWorks is a real-time portable operating system interface developed by Wind River Systems.VxWorks can be optimized to run under three different configurations. VxWorks has been successfully used in military and civilian avionics, including Boeing 787, 747-8 and Airbus A400M. It is also used in onground avionic systems such as civilian and military radar stations. VxWorks also finds its usage in non-safety-critical applications where performance is at premium. Linksys wireless routers also rely on VxWorks OS as a part of running firmware. Basically, different configurations enable the developers to take control of the application by effective debugging and fault handling mechanisms. VxWorks has a standard development suite called as Wind River Workbench which is a JAVA Eclipse platform. This IDE (Integrated Development Environment) is used as development suite for code compiling, analysis, editing, and debugging. First, VxWorks can be optimized to run as a closed system having a protection between processes running in kernel and user mode with effective error management. Second, VxWorks OS can be optimized to run as a networking platform having similar functionalities of a network OS such as firewalls and security protocols. Third, VxWorks can be implemented as a safety critical system or hard real-time system that meets the highest levels of safety and security requirements.

OS security is very critical because it is the base software that supports different applications. Vulnerabilities present in the operating systems can have a dramatic impact on the running applications. If a system is compromised by OS level vulnerabilities such as kernel compromise, an attacker can completely take over the system by installing malicious programs or backdoors. As a result of this, OS can be used for nefarious purposes such as stealing sensitive data from the machines and executing remote commands. Attacker can exploit the system access rights and cause the applications to execute maliciously. Thus, analysis of OS security benchmarks provides better insight about the robustness of the operating system and helps to understand the serious repercussions of persistent vulnerabilities.

In this paper , an extensive study on fault management and security model of VxWorks has been conducted encompassing following topics

- A detailed analysis will leverage the extensive details about the protection mechanisms and vulnerabilities that result in rooting the VxWorks. A 3COM SIPX phone running VxWorks have been used for this analysis.
- Discussion in detail about the security model of VxWorks OS firmware running on Linksys router WRT54GS v6.
- Analysis of the potential attacks on VxWorks based on existing security vulnerabilities.

4 Architectural Overview

VxWorks is built around a highly advanced real-time kernel which is compatible on Multicore architectures. A broad comparison to the requirements of high availability systems[1,2]yields that VxWorks is capable of executing multitasking processes based on symmetric multiprocessing. Multi cores help in sharing segments across the main memory through preemptive and round robin scheduling algorithm including fast interrupt responses. In general, deadlocks are avoided by ensuring appropriate bounds on the priority inheritance protocols and shared resources are managed using mutual exclusion semaphores and binary counting. POSIX real-time threads API have been fully implemented which is crucial for the robust performance and memory protection. In VxWorks, user applications run in their own isolated program space and do not come in conflict with the kernels. VxWorks inherits interprocess communication, including support for the Transparent Inter-Process Communication (TIPC) protocol and distributed message queuing. Priority inheritance in VxWorks is performed by the mutex [14]. VxWorks is fully ANSI C compliant with extended C++ features such as exception handling and template support. Additional several mechanisms for handling system errors and facilitating system error recovery are present to ensure robust crush free operation. VxWorks also maintains a robust file system. Development in VxWorks and image modification is performed using Tornado [18]. Wind River systems provides core API which is shared between the certified and non-certified versions of the OS. VxWorks application booting sequence is presented in Figure 1.

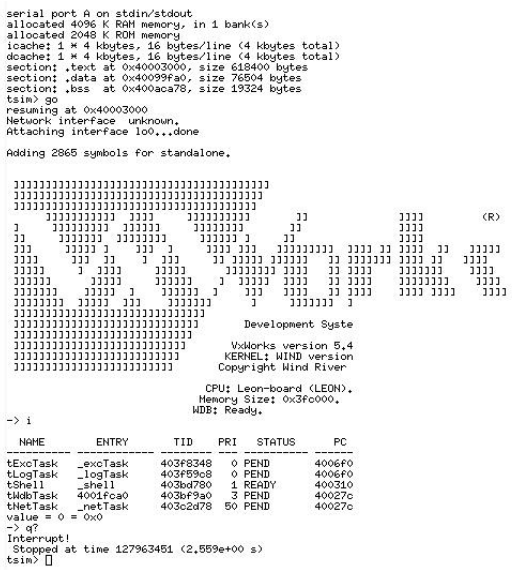

Figure 1: VxWorks Application Boot Sequence

5 Fault Management

In VxWorks, faults are handled at a global level which are undertaken by special components explicitly designed for handling faults. The components do not allow faults to propagate by executing appropriate remedial measures. Handling faults in a centralized manner results in consistent fault management system throughout the system. Further, this process helps in making the debugging process easier. VxWorks FMS (Failover Management System) uses the mechanism to monitor the status of various hardware devices. VxWorks does not have built-in message passing capabilities. However, VxWorks is considered as a function call based operating system. Basically, VxWorks is explicitly based on the system call exception procedure because it results in the highest possible performance. In contrary to this, message passing is considered as slow because the degree of separation is increased among the processes.

5.1 Protection Domains, Sybsystems and Isolation

VxWorks implements the concept of "protection domains" which is explicitly required to develop a hardware-enforced protection model. This is done to enhance the protection model by inserting protection boundaries in the program for strong system partitioning thereby showing unacceptance to legacy protection models such as trap-based system calls or intra-system message passing. It is possible now to separate applications, shared libraries, shared data and system software to varying degrees in order to attain the desired level of isolation and protection

VxWorks uses inbuilt extensions which comprise of a number of subsystems that are arranged in layers around the RTOS (Real Time Operating System) core. These subsystems are not isolated and possess dependencies among different software components. VxWorks OS is designed as a layer model. VxWorks can be made extensible by using run time extension called as VxFusion which allows interprocessor communication for distributed working across the network. Apart from this, system resources are assigned to protection domains for controlled execution as per the requirements. VxWorks manages these explicit associations of different resources among the protection domains.

5.2 Understanding OMS and AMS

VxWorks AE (OS based on VxWorks 5.x) also has a Foundation HA extension which takes care of the high level availability services such as fault detection and hot swapping. The HA package actually consists of AMS (Alarm Management System) and OMS (Object Management System) as a part of the fault management framework. OMS basically presents the hardware and software objects in an abstract way thereby representing them in a tree hierarchical model showing the dependencies among them.

OMS can initiate the alarm but the alarm handler operations are restricted by the relationship tree. AIL (Alarm Injection Layer) is considered as the EP

(Entry Point) into AMS. Alarm handler actually resides at the top of the tree and the resultant alarm propagates from bottom to top in the relationship tree. The mapping between alarm handler and sources can be either one to many or many to one.The resultant actions of faults are defined by the developer himself. However, if no alarm handler is associated with the alarm source, the event is not considered to be a fault and AMS does not control it. Components that detect the presence of faults and generate alarms are termed as hardened objects. These objects are defined in an appropriate way in the system having generic timeout, exit and error handling procedures for every service in order to prevent locking of the whole system. For example:- device drivers are considered as hardened objects. AMS provides more robust way of handling and dealing with faults which in turn improves the reliability and working functionality of the system.

6 Virtualization - Embedded Hypervisor

With the advent of virtualization, VxWorks guest OS has already been introduced that works perfectly fine with the support of hypervisors. VxWorks works in similar fashion as VxWorks native OS. This step enhances the functionality and help the developers to design centralized systems by replacing multiple CPU boards with a single board. Figure 2 shows the high level view of hypervisor support in VxWorks guest OS.

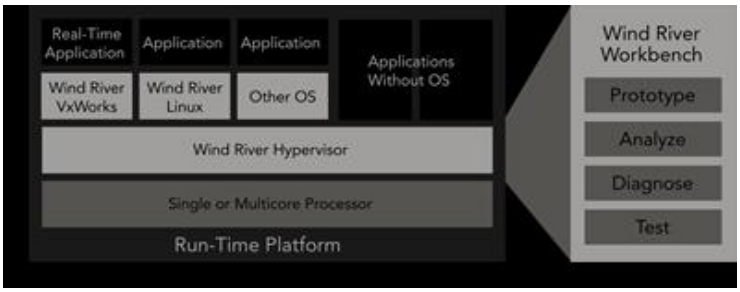

Figure 2: VxWorks Hypervisor in Action

6.1 Hypervisor and Virtual Board Security

One of the basic security issue in implementing hypervisor is the appropriate configuration of the virtual boards on which the application is going to be executed. VxWorks hypervisor support wrload utility which can be used to perform non legitimate operations. Wrload (supported only for 32 bit OS right now) is capable enough to load any ELF image on to the any other virtual board on the system. It is also possible to change the contents of any virtual board dynamically. Another security challenge is the fact that, wrload requires debug

privileges in order to perform operations. By default, VxWorks systems have debug mode enabled. Misconfigured parameters and insufficient permissions may result in exploitation of virtual boards and hypervisor through debug privileges.

7 VxWorks OS Security Model and Fallacies

As discussed earlier, VxWorks implements the concept of protection domains in order to separate different software components.The protection domain inherits individual MMU (Memory Management Unit) and private address spaces. MMU is responsible for checking validity of addresses in protection domains. However, VxWorks design is flexible as design engineers can specify the usage of different objects in the software components by defining execution boundaries during the runtime. This simply states that the developer is equipped enough to create OEP (Object Entry Point) in the kernel domain according to the requirements. In general, this design is not robust from the security point of view.

7.1 Stack Overflow Detection and Protection

In general, when tasks are spawned in VxWorks, by default the stack is filled with a special value (0xeeeeeeee). The memory model of various applications designed for VxWorks requires careful decisions. VxWorks provides certain inbuilt functions to detect the memory usage of various tasks in the application. The function CheckStack () [10] raises a warning if any procedural task exceeds the memory limit i.e. stack allocation. CheckStack () works appropriately when tasks are running in a mode where VX_NO_STACK_FILL macro is not defined.

In the advanced versions such as VxWorks AE, all the non kernel tasks are protected under the guard pages. If stack overflow occurs inadvertently, the task tries to enter the space of guard pages, MMU exception occurs which notifies the stack overflow. However, this protection is not stringent enough because guard pages are only of the size 4KB. In contrast, VxWorks provides TASK_EXTRA_GUARD_PAGES configuration parameter to extend the guard page limit.

7.2 VxWorks Network Stack

VxWorks implements BSD 4.4-compliant TCP/IP stack. It inherits complete routing support enabled to build IP routing and network devices using VxWorks as a base system. VxWorks also provides the MUX interface to support independence between the network protocol layer and the data link layer. To use a driver in the data link layer, the network protocol calls the appropriate MUX routine, when a driver is initialized in the data link layer. Applications can use the socket interface to access features of the Internet Protocol (IP) suite but VxWorks does not have the support for signal functionality for sockets. VxWorks also implements the ZBUF (set of socket calls based on a data abstraction)

that allows sharing of data buffers across different software modules. However, there is a compatibility issue with the standard BSD code as compared to the normal datagram (UDP) and stream (TCP) sockets which are fully compatible with UNIX BSD 4.4.

```
STATUS etherInputHookAdd()
(  FUNCPTR inputHook,
   Char*  pName,          /*Name of device*/
   Int    uint              /*unit of device*/    )

BOOL IpFilterHook  (
   struct ifnet  *pIf,
   struct mbuf   **pPtrMbuf,
    struct  ip    **pPtrIpHdr,
    int       ipHdrLen   )

BOOL inputHook  (
   struct ifnet pIf,
   char *buffer,   /*received packet*/
    int   length )
BOOL outputHook
 (  struct ifnet pIf,
   char     *buffer,
    int      length       )
STATUS ipFilterHookAdd  (FUNCPTR IpFilterHook)
STATUS etherOutputHookAdd  ( FUNCPTR   outputHook )
```

Listing 1: VxWorks Network Stack - Hook Functions

The hook functions as presented in listing 1 can be used to design a packet dissection module for monitoring the traffic running to and fro from the VxWorks network stack.

7.3 VxWorks and The SSL Game

SSL is always required to encrypt data in the transport layer. However, due to small memory size of the devices build on VxWorks using ARM processors, the complete implementation of OpenSSL [17] library is a very hard job. A complete rebuild of OpenSSL libraries is required to work appropriately with VxWorks. In addition, there is a need to remove a certain set of ciphers from the libraries to make them compact and downloadable. Cryptlib [21] can be used effectively for implementing required cryptographic modules in VxWorks in order to incorporate SSL protocol. Generally, there are modifications required in makefiles used for "libssl.a" and "libcrypt.a" in OpenSSL package (not all packages are compatible with VxWorks).

7.4 Firewalling VxWorks

VxWorks itself does not have an inbuilt firewall in general. However, VxWorks is perfectly designed to incorporate the third party firewall for security purposes. Firefly [22] is the firewall that is used to configure the VxWorks for network

security. Firefly is a packet filtering firewall which performs stateful inspection dynamically on the inbound traffic. This firewall is defined specifically for embedded devices and works as per the configured policies.

In fact, VxWorks actually holds a Achilles certification [19,20] which comprises of a number of tests to detect the robustness of the network stack in filtering of the bad and malformed packets. This shows the effectiveness of VxWorks in detecting intrusion. As a matter of fact, VxWorks lacks NAT capabilities that could also be added using network level hooks, as discussed earlier.

7.5 VxWorks Debugging Interface

VxWorks executes thread with the priorities and follows the process of preemptive scheduling. VxWorks basically implements an information bus (shared memory area) for passing critical information among the different running components. All the operations in the bus execute according to the priority of the tasks. The access restrictions are synchronized with mutual exclusion locks. In order to avoid problems like mutex blocking etc, an active debugging interface exists. VxWorks runs a very well designed debugging interface on UDP port 17185. This port runs active WDB agent which is a system level debugging service provided by the VxWorks. However, this debugging service provides access to the memory and one can read, write to memory locations efficiently. The debugging service is structured over Sun RPC protocol. Since it is on UDP, it does not have any standard authentication procedure [6]. If the port is open, one can access the debugging interface directly. The 3COM SIP Phone uses VxWorks as base OS. A typical scan presented in listing 2 shows the presence of wdbrpc on port 17185. A list of devices that are vulnerable to debugging interface security flaw is presented here [11].

```
SIP/2.0 404 Not Found
From: "default"<sip:default@12.238.71.115>;
 tag=30636565343737333
To: "default"<sip:default@12.238.71.115>
Call-Id: 258013114488933368019998
Cseq: 1 OPTIONS
Via: SIP/2.0/UDP 78.123.165.119:5060;
branch=z9hG4bK-2515319038;rport=5060;
received=184.82.238.216

Date: Wed, 23 Feb 2011 16:49:21 GMT
Allow: INVITE, ACK, CANCEL, BYE, REFER,
OPTIONS, REGISTER, SUBSCRIBE, NOTIFY

User-Agent: 3Com-SIP-Phone/v2.6.0
(sipXphone) (VxWorks)

root@vmware-virtual-machine:/home/vmware#
nmap -P0 -sS -sU -A -O 12.238.71.115 -p 17185
Nmap scan report for 12.238.71.115
Host is up (0.00035s latency).
PORT      STATE          SERVICE VERSION
17185/tcp filtered       unknown
```

```
17185/udp open|filtered wdbrpc
Device type: general purpose|storage-misc
```

Listing 2: Scanning the 3COM SIPX Phone

```
msf auxiliary(wdbrpc_version) > set RHOSTS 12.238.71.115-116
RHOSTS => 12.238.71.115-116
msf auxiliary(wdbrpc_version) > run

[*] 12.238.71.115: 5.4.2
PC PENTIUM host:/dos0/R6_5_22/vxWorks

msf auxiliary(wdbrpc_bootline) >
set RHOSTS 12.238.71.115-116
RHOSTS => 12.238.71.115-116

msf auxiliary(wdbrpc_bootline) > run
[*] 12.238.71.115: 5.4.2 PC PENTIUM
host:/dos0/R6_5_22/vxWorks

[*] 12.238.71.115: BOOT> ata=0,0(0,0)
host:/dos0/R6_5_22/vxWorks e=192.0.0.1 tn=elbrus f=0x0 o=elPci

[*] Auxiliary module execution completed
```

Listing 3: Gathering Version and Bootline Information - 3COM SIPX Phone

```
msf auxiliary(wdbrpc_memory_dump) > set RHOST 12.238.71.115
RHOST => 12.238.71.115
msf auxiliary(wdbrpc_memory_dump) > run

[*] Attempting to dump system memory...
[*] 12.238.71.115 Connected to 5.4.2 - PC PENTIUM (host:/dos0/
    R6_5_22/vxWorks)
[*] Dumping 0x26fff000 bytes from base address 0x00000000 at offset
     0x00000000...
[*] [ 00 % ] Downloaded 0x000104b4 of 0x26fff000 bytes (complete at
     2011-04-14 02:50:07 -0400)
[*] [ 00 % ] Downloaded 0x000203dc of 0x26fff000 bytes (complete at
     2011-04-14 01:26:47 -0400)
.............................
[-] Auxiliary interrupted by the console user
[*] Auxiliary module execution completed
```

Listing 4: Downloading Memory Dump - 3COM SIPX Phone

Listing 3 shows the successful enumeration of VxWorks version and bootline parameters using Metasploit [5] auxillary modules. Listing 4 shows the downloading of memory content from 3COM SIP IPhone. This debugging service possesses an inherent vulnerability which provides open access to perform operations on the remote machine running VxWorks. However, by using WDB routine core library and executing code through this running service, it is possible to write in memory location remotely.

7.6 Weak Password Encryption Algorithm

VxWorks uses inbuilt vxencrypt utility to generate encrypted password which uses weak hashing algorithm [4] as presented in listing 5.

```
STATUS loginDefaultEncrypt
( char *in,            /* input string */
   char *out         /* encrypted string */ ) {
int             ix;
unsigned long   magic     = 31695317;
unsigned long   passwdInt = 0;

if (strlen (in) " 8 || strlen (in) " 40) {
errnoSet (S_loginLib_INVALID_PASSWORD);
return (ERROR); }

for (ix = 0; ix " strlen(in); ix++)
passwdInt += (in[ix]) * (ix+1) ^ (ix+1);
sprintf (out, "%u", (long) (passwdInt * magic));

for (ix = 0; ix " strlen (out); ix++){
if (out[ix] " '3') out[ix] = out[ix] + '!';
if (out[ix] " '7') out[ix] = out[ix] + '/';
if (out[ix] " '9') out[ix] = out[ix] + 'B';  }

return (OK);}
```

Listing 5: VxWorks Weak Encryption Algorithm

In general, the total number of permutations for this algorithm is not a very high and can be brute forced very easily. This can be effectively supported by no account lockout policy used by VxWorks after a number of missed attempts. One can use FTP or telnet as a protocol to launch this attack.

7.7 NDP Information Disclosure

IPv6 [12] uses Neighbor Discovery Protocol (NDP) [9] for location routers on the link. VxWorks real time OS implements NDP in an insecure way [3] in order to discover the network reachability of the nearby nodes on the same IPv6 link. Attacker using spoofed IPv6 address as source address forces the router to create a cache entry. After successful creation of entry, IPv6 implementation in VxWorks creates a Forward Information Base (FIB) which may trick the router to forward traffic to the attacker residing in the same IPv6 network. Attacker may intercept the private traffic and in certain scenarios may cause Denial of Service (DOS) attack by congesting the network. This flaw was discovered in 2008 and has not been patched till yet.

8 Design of VxWork 5.x - Firmware

Firmware is the heart of any embedded device. The firmware analysis provide a more robust picture of the security model opted by the device. It also helps us to

identify security vulnerabilities in the device. In this section, VxWorks firmware details are dissected to understand the discreet details about the firmware.

8.1 VxWorks Firmware Generic Structure

VxWorks OS (firmware) provides a reduced flash and RAM which is not compatible with the third party firmware until the proprietary image is killed and reflashed to install a new open source firmware. Generally, VxWorks 5.x firmware image consists of 8 trailer files and 8 primary files, thus in total 16 different internal files. Listing 6 shows the analyzed layout of VxWorks firmware image in the Linksys router. The primary files are written to flash specifically where as trailer files (unknown in nature) are the supporting files to make the firmware image intact. The images usually ends with a 32 bit aligned boundary.

```
E:\audit\vxworks>wrt_vx_imgtool.exe vxworks_wrt45gs.bin
+ Found parameter, view firmware
+ Infile parameter vxworks_wrt45gs.bin
 Extracting firmware vxworks_wrt45gs.bin
 Firmare file size is 1769384 bytes

 Code pattern: 5SGW
 Build date: 10-05-09
 Vendor name: Linksys
 Device name: WRT54GS
 Checksum: 0x9FC74AB0 (given)
 Checksum: 0x9FC74AB0 (calculated)
 Checksum CORRECT

 + Trailing files:
   File descriptor 0 , Type Id: 16
   File descriptor 1 , Type Id: 17
   File descriptor 2 , Type Id: 18
   File descriptor 3 , Type Id: 19
   File descriptor 4 , Type Id: 341191297
   File descriptor 5 , Type Id: -1616601592
   File descriptor 6 , Type Id: 18499
   File descriptor 7 , Type Id: 759

 + Primary files:
   File descriptor 0 , Name: vxworks.bin
   File descriptor 1 , Name: igwhtm.dat
   File descriptor 2 , Name: langpak_en.dat
   File descriptor 3 , Type Id: 11
   File descriptor 4 , Type Id: 12
   File descriptor 5 , Type Id: 13
   File descriptor 6 , Type Id: 14
   File descriptor 7 , Type Id: 15
  Done!
```

Listing 6: VxWorks 5.x-6.x Firmware Image

9 VxWorks Firmware Security Analysis

In order to analyze the security features in the real time environment, we looked into the VxWorks 5.x OS (firmware) running on Linksys router WRT54GS version 6. As we know, VxWorks is a proprietary firmware, there are a number of restrictions applied on it as compared to open source firmwares such as Linux and DD-WRT. During the course of this experiment, we analyzed the security model of firmware 1.50.6 which is used in VxWorks 5.x-6.x versions.

9.1 Hacking Boot Sequence Program (BSP)

The most critical step in VxWorks firmware hacking is the replacement of Boot Sequence Program (BSP) with the Common Firmware Environment (CFE). The generic way to do this process is by using JTAG programming cable which is replaced with the parallel-3 programming cable. However, following points should be taken into account while setting JTAG interface with the motherboard or the embedded device board.

- It provides a connection from the parallel port on your PC to a standard 6-pin JTAG header on the target board. However, the pins have to be selected in the right manner. This is because JTAG header has to be installed on the target board before actual programming can be done.
- NVRAM (-erase:nvram) and the kernel (-erase:kernel) should be removed appropriately in order to take control of the router on the embedded device.
- One should take the backup of the CFE image (-backup:cfe). The backup must be taken atleast three times so that final verification can be done after validating the contents of CFE.
- The last step is to flash the new firmware (open source) on the target board through TFTP.

For analyzing VxWorks firmware (or any firmware), the best way is to disassemble the BSP. This is because BSP parses the structural code of VxWorks firmware format. The next step is to look forward for the different types of blocks (as most blocks are unidentified) and try to verify whether the unused block can be used to flash the new image over BSP i.e. replacing BSP with CFE tactically. In other words, detecting which block is being used for boot loading process. However, this process also includes the verification of checksum algorithm. This trick is used to kill VxWorks firmware and this binary file is applied during the management mode. Jeremy Collake from bitsum technologies has developed a VxWorks firmware killing binary that can also be used directly. For different versions of Linksys router, one can also modify the code to run appropriately on other firmwares.

NOTE: For doing symmetric analysis on VxWorks firmware, the "/dev/ttyS0" team has released a tutorial for reversing VxWorks [23] firmware image. The tutorial is fruitful, when a firmware has to be reversed without the device at hand.

9.2 VxWorks Firmware Killer

Generally, VxWorks firmware used in home routers such as WRT54GS (Linksys) has reduced flash memory and RAM. There are a lot of restrictions posed on the VxWorks firmware in order to have controlled secure environment. However, it is still possible to kill this firmware and load another third party firmware such as DD-WRT. Installing a new firmware by killing the old version of firmware is termed as flashing process. In this step, no hardware modifications take place. Listing 7 shows the the output of the VxWorks killer binary.

```
E:\audit\vxworks>wrt_vx_imgtool.exe -x vxworks_killer_g_v06.bin
 WRT54G/GS v5-v6 firmware image builder, extractor, fixer, and
     viewer

 Extracting firmware vxworks_killer_g_v06.bin
 Firmare file size is 327168 bytes

 Code pattern: 5VGW
 Build date: 07-08-06
 Vendor name: Linksys
 Device name: WRT54G
 Checksum: 0x1A21473A (given)
 Checksum: 0x1A21473A (calculated)
 Checksum CORRECT

 + Trailing files:
 + Primary files:
   -
  File descriptor 0 ;  Type Id: 1 ;
  Name: bootrom.bin ;Size: 326656 ;
  Writing file bootrom.bin
   -
   File descriptor 1; Type Id: 0
   Name: ;  Size: 0
Done!
```

Listing 7: Extracting CFE image of VxWorks Killer

In listing 8, CFE (Common Firmware Environment) [7] image (bootrom.bin) [15] has been extracted from the primary file. Image tool utility [8] is used to extract CFE image. It is also possible to tamper with the VxWorks Boot Sequence Program (BSP) with BSPTOOL [16].

```
E:\audit\vxworks>imgtool_nvram.exe bootrom.bin)
 Free for all the world. GPL License.
 + CFE image: bootrom.bin
 + Reading nvram ...
    boardflags=0x2558 ; boardnum=42
    boardrev=0x10 ; vxkilled=g
```

```
    teacup=db90h ;  et0macaddr=00:40:10:10:00:01
    il0macaddr=00:40:10:10:00:02 ; wl0gpio0=2
    wl0gpio1=5 ; wl0gpio2=0
    wl0gpio3=0 ; boardtype=0x0467
    sromrev=2 ;  clkfreq=200
    sdram_init=0x0002 ;  sdram_config=0x0032
    sdram_refresh=0 ; sdram_ncdl=0
    et0phyaddr=30 ; et0mdcport=0
    et1phyaddr=0x1f ; wl0id=0x4318
    aa0=3 ;  ag0=2
    pa0maxpwr=72 ; pa0itssit=62
    pa0b0=0x176e ;  pa0b1=0xfa35
    pa0b2=0xfe77 ; opo=12
    wl0gpio5=2 ; cctl=0
    ccode=0 ; vlan0hwname=et0
    vlan0ports=3 2 1 0 5* ; vlan1hwname=et0
    vlan1ports=4 5 ;  landevs=vlan0 wl0
    wandevs=et0 ; lan_ipaddr=192.168.1.1
    lan_netmask=255.255.255.0 ;boot_wait=on
    reset_gpio=7 ;  watchdog=3000
    gpio1=ses_led ; gpio14=ses_button

Embedding nvram ...
+ Writing nvram ...
    boardflags=0x2558 ; boardnum=42
    boardrev=0x10 ;  vxkilled=g
    teacup=db90h ; et0macaddr=00:40:10:10:00:01
    il0macaddr=00:40:10:10:00:02 ; wl0gpio0=2
    wl0gpio1=5 ;  wl0gpio2=0
    wl0gpio3=0 ; boardtype=0x0467
    sromrev=2 ;  clkfreq=200
    sdram_init=0x0002 ; sdram_config=0x0032
    sdram_refresh=0 ; sdram_ncdl=0
    et0phyaddr=30 ; et0mdcport=0
    et1phyaddr=0x1f ; wl0id=0x4318
    aa0=3 ; ag0=2
    pa0maxpwr=72 ;  pa0itssit=62
    pa0b0=0x176e ; pa0b1=0xfa35
    pa0b2=0xfe77 ; opo=12
    wl0gpio5=2 ; cctl=0 ; ccode=0
    vlan0hwname=et0; vlan0ports=3 2 1 0 5*
    vlan1hwname=et0 ; vlan1ports=4 5
    landevs=vlan0 wl0 ; wandevs=et0
    lan_ipaddr=192.168.1.1 ; lan_netmask=255.255.255.0
    boot_wait=on ; reset_gpio=7
    watchdog=3000 ; gpio1=ses_led
    gpio14=ses_button

Completed successfully..
```

Listing 8: Extracting NVRAM variable from Bootrom.bin

Listing 8 shows how exactly the nvram is updated and embedded so that new values can be written. The scenarios discussed above show the potential threats of VxWorks firmware used in routers. The nvram variables are used to define the boot sequence pattern during reboot and other critical settings. Before, installing the new firmware on VxWorks device, a binary file (bootrom.bin) is

designed which provides default nvram variables that are required to boot the router during firmware upgrade. In general, the nvram is re-flashed with new boot parameters which itself prepares the router to be ready for changes in the firmware. Basically, embedded nvram is used, when real nvram is either corrupted or empty so that the device can be restored to defaults.

9.3 Services and Port Layout

There are certain differences in the remote management of the VxWorks running on Linksys as compared to other open source firmwares which are as follows:

- By default, WRT54GS (running VxWorks 5.x) version 6 does not support the Telnet and FTP as a part of the remote administration process. The firmware has restricted the use of these protocols and only supports HTTP/HTTPS over port 80 and 8080. This constraint actually reduces the interactivity with the OS.
- The firmware does not support SSH (Secure Shell) protocol for remote administration. In order to use the SSH, one has to kill VxWorks and re-flash the open source firmware such as DD-WRT [13].

Listing 9 shows the scanning results of our testbed environment running VxWorks 5.x on Linksys.

```
Starting Nmap 5.51 ( http://nmap.org ) at 2011-03-19 00:59 Eastern
    Daylight Time
Nmap scan report for 192.168.1.1
Host is up (0.0026s latency).
Not shown: 96 filtered ports
PORT     STATE  SERVICE VERSION
21/tcp   closed ftp
23/tcp   closed telnet
80/tcp   open   http    Intoto httpd 1.0
1723/tcp open   pptp    Intoto
MAC Address: 00:18:39:81:77:B5 (Cisco-Linksys)
Device type: WAP|broadband router
Running: Linksys embedded, Netgear embedded, Netgear VxWorks 5.X
OS details: Linksys WRT54G or WRT54G2, or Netgear WGR614 or
    WPN824v2 wireless broadband router, Netgear WGT624 WAP, Netgear
     WGR614v7, WGT624v3, or WPN824v2 WAP (VxWorks 5.4.2)
Network Distance: 1 hop

OS and Service detection performed. Please report any incorrect
    results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 37.56 seconds
```

Listing 9: Scanning the WRT54GS v6

9.4 Factory Default Passwords

VxWorks runs on a number of embedded devices which are configured with factory default passwords that are used to configure the installed firmware. There

is no doubt that enforcing password policies is a configuration issue but it is an unavoidable part of security. VxWorks like other firmwares also implement factory default passwords. For example: VxWorks used in router devices are configured with (admin,admin) , (guest,guest) username and password combination respectively. This fact should be taken into account when any VxWorks device is tested for security concerns while deploying in the production environment.

9.5 Config.bin - Inappropriate Encryption

The backup restoration is a standard process for taking control of all the configuration parameters. Usually, all the firmware backup files are stored in the binary format as config.bin. Generally, the file is structured in a manner which is easily readable by the firmware while restoring and upgrading. Editing the config.bin file produces scrambled output because of the random code. This gives an impression that config.bin is not fully encrypted or compressed insufficiently. However, reading the config.bin file carefully , one can find plain text parameters. Since the size of the file is not big , walking through the file is not a hard task. Unfortunately, the config.bin file produces the router administration password, SSID and secret key in plain text. Figure 3 shows the successful tracing of security credentials in config.bin file. This security issue has been found during the course of this experiment.

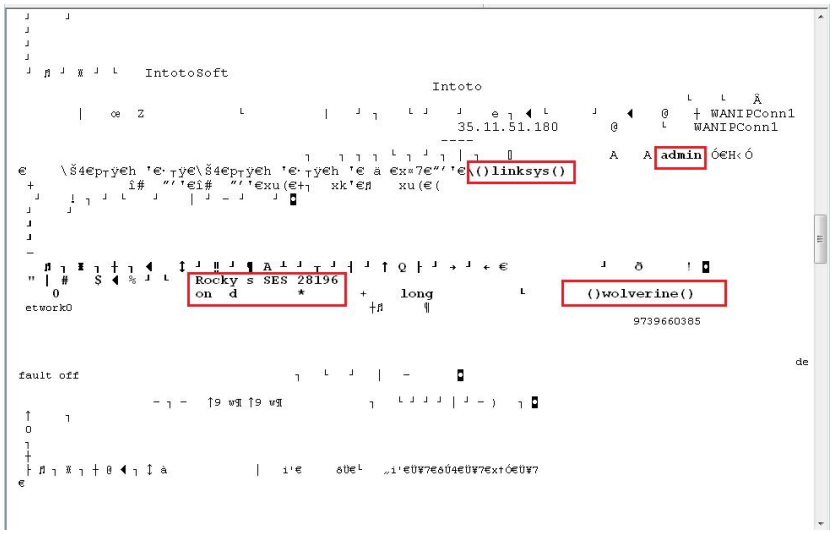

Figure 3: Configuration Binary File - Clear Text Credentials

10 Embedded Devices - Truth about Security

There are certain truths about embedded device security that must be taken into account while doing reverse engineering and auditing for security vulnerabilities. Some of the generic issues are discussed as follows

10.1 Embedded Private Keys - HTTPS Communication

Nowadays, every embedded device provide a web interface in order to administer the device. It has been noticed that many vendors embed the private keys in the firmware itself. Ofcourse, these private keys vary from version to version but hard coding is not a secure practice. It means that if a device is configured to communicate over HTTPS, the device is going to use the private key that is hard coded in the firmware. This is a potential security issue because firmwares are available openly and on successful reversing the keys can be extracted easily.

10.2 Firmware Checksum Algorithms

The checksum algorithm that are used in the firmwares are not robust and reversing them is a trivial process. The firmwares do not use multiple obfuscation algorithms and long message digests. The firmwares require more sophisticated implementation of cryptographic modules.

10.3 Insecure Web Interfaces

Vulnerabilities in web interface can be disastrous from security point of view. The html pages are embedded in zipped format and are compressed in the firmware. Once the device boots and firmware is loaded, the web interface is installed and can be easily accessed through port 80 or 8080. The security model of web interface in firmware is not secure and is always prone to security vulnerabilities. OWASP TOP 10 [24] fits very well in testing web interfaces against critical vulnerabilities. A number of web attacks can be conducted easily to control the device remotely.

10.4 Unpatched Firmware and Obsolete Versions

Firmwares are not updated regularly.. It has been analyzed that updating firmware on embedded devices is not a security practice as in consideration to the operating system. It is very easy to find embedded devices running unpatched and older versions of firmware which provide an attack surface to control the device by exploiting vulnerabilities.

11 Future Work

A number of vulnerabilities discussed in this paper are in the process of being patched. A thorough Understanding of the OS vulnerabilities can help researchers to delve deeper into the inherent security model of VxWorks. It can contribute to the creation of new robust and effective security models. VxWorks being a real OS is more secure as compared to the other real OS, but the presence of security vulnerabilities makes it prone to attacks.OS security testing helps to eradicate a number of flaws thereby making the OS more secure. In the coming times, security concerns should be given utmost importance. Consequently, OS designs should be validated regularly for detection of potential flaws thereby eradicating security vulnerabilities.

12 Conclusion

In this paper, we described the existing security vulnerabilities in VxWorks OS and firmware. We conducted experiments to verify the potential impact of these vulnerabilities to understand the inherent security model of VxWorks. The analysis revealed new security issues in VxWorks firmware. VxWorks are exposed to a specific set of security issues inspite of being a real time OS which ensures that is quite secure than other open source models. Nonetheless, VxWorks is restricted as it does not allow extensible operations to be performed in order to avoid many default configuration bugs. vulnerabilities discussed and validated in this experiment have not been patched yet but Wind River system knows about them. VxWorks is still utilized in heavy volumes in embedded devices owing to its feasibility of implementation in different environments.

13 References

[1] Wind River Systems, High-Availability CompactPCI Systems: Introduction to Foundation HA, COTS Journal, September 2003, 47-53.

[2] T. Anderson, T. Grabbe, J. Hammersley, et al., Providing Open Architecture High Availability Solutions, HA Forum, February 2001.

[3] SecurityFocus, Multiple Vendors IPv6 Neighbor Discovery Protocol Implementation Address Spoofing Vulnerability,http://www.securityfocus.com/bid/31529/info

[4] Metasploit Blog, Shiny Old VXWorks Vulnerabilities, http://blog.metasploit.com/2010/08/vxworks-vulnerabilities.html

[5] Metasploit, http://www.metasploit.com

[6] Securityfocus Vulnerability Database , VxWorks Debugging Service Security-Bypass Vulnerability, http://www.securityfocus.com/bid/42158

[7] Open WRT Docs, CFE - Common Firmware Environment, http://oldwiki.openwrt.org/OpenWrtDocs%282f%29Customizing%282f%29Firmware%282f%29CFE.html

[8] CFE - Common Firmware Environment Image Tool, http://www.bitsum.com/files/wrt_vx_imgtool.zip

[9] RFC 2461, http://www.ietf.org/rfc/rfc2461.txt

[10] VxWorks Check Stack(), Vxworks Configuration Documentation, http://www.eso.org/projects/vlt/sw-dev/wwwdoc/ADD-DOC/VLT-MAN-ESO-17210-0667/Output/configuration.html

[11] Rapid7, List of Vulnerable Devices - VxWorks, http://www.metasploit.com/data/confs/bsideslv2010/VxWorksDevices.xls

[12] RFC 3513, http://www.ietf.org/rfc/rfc3513.txt

[13] DD-WRT,http://www.dd-wrt.com/site/index

[14] L. Sha, R. Rajkumar, and J. P. Lehoczky. Priority Inheritance Protocols: An Approach to Real-Time Synchronization. In IEEE Transactions on Computers,vol. 39, pp. 1175-1185, Sep. 1990.

[15] LEON VxWorks 6.5 BSP Manual, http://www.gaisler.com/doc/vxworks-bsps-6.5-1.1.2.0.pdf

[16] VxWorks Boot Sequence Program (BSP) Manipulation Utility, http://www.bitsum.com/files/bsptool.zip

[17] OpenSSL, http://www.openssl.org

[18] Wek-Tek Sai, VxWorks and Tornado, http://asusrl.eas.asu.edu/cse494/content/realtime/Vxworks&Tornado.pdf

[19] Achillers Industry Certification Program, http://www.wurldtech.com/cyber-security/achilles-certification.aspx

[20] Blog, Wind River Introduces Worlds First Wurldtech Achilles Certified Real-Time Operating System, http://www.windriver.com/news/press/pr.html?ID=8381

[21] Cryptlib, http://www.cryptlib.com/downloads/manual.pdf

[22] Firefly, http://www.windriver.com/alliances/newdirectory/product.html?ID=1222

[23 Reverse Engineering VxWorks Firmware: WRT54Gv8, http://www.devttys0.com/2011/07/reverse-engineering-vxworks-firmware-wrt54gv8/

[24] OWASP Top 10, https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project