

Attacking Next- Generation Firewalls

Breaking PAN-OS

Felix Wilhelm



#whoami



- Security Researcher @ ERNW Research
- Application and Virtualization Security
- Recent Research
 - Hypervisors (Xen)
 - Security Appliances (Fireeye, Palo Alto)
- @_fel1x on Twitter

The Target



- Palo Alto Next-Generation Firewall
- Pan-OS
 - Software stack running on Palo Alto devices
- Analyzed device is a PA-500
 - .. but bugs affect all (unpatched) devices
- Main focus lies on attacks against the device itself
 - ..not detection bypasses

Features



<https://www.paloaltonetworks.com>

- „Next Gen Firewall“
- Management Interfaces
 - Web + SSH
- Signature Matching
 - IPS, Exploit Detection, Wildfire Malware Analysis
- App-ID
- User-ID
- GlobalProtect

Overview

| | Management Interfaces | Signature Matching | App-ID | User-ID | GlobalProtect |
|--------------------------|-----------------------|----------------------|--------------------------------|-----------|----------------------------|
| Availability / Interface | Trusted | Untrusted / External | Untrusted | Untrusted | External |
| Analyzed? | Yes | No | Partially | Yes | Yes (not for User-ID cap.) |
| Impression | ? | - | Seems ok from first impression | ? | ? |

Agenda



- Breaking In
- PAN-OS Architecture
- Attack Surface
 - Management Interface
 - User-ID
 - GlobalProtect
- Conclusion

Breaking In

```
admin@PA-VM>
clear          Clear runtime parameters
configure     Manipulate software configuration information
debug        Debug and diagnose
delete       Remove files from hard disk
diff        Local configuration diffs
exit        Exit this session
find        Find CLI commands with keyword
ftp         Use ftp to export files
grep        Searches file for lines containing a pattern match
less        Examine debug file content
ls          Examine debug file listing
netstat      Print network connections and statistics
ping        Ping hosts and networks
quit        Exit this session
request      Make system-level requests
schedule     schedule test jobs
scp         Use scp to import / export files
set         Set operational parameters
show        Show operational parameters
ssh         Start a secure shell to another host
tail        Print the last 10 lines of debug file content
--more--
```

- Administrative Interfaces: CLI over SSH and Web Interface
 - Do not give full access to the operation system
- „Jailbreak“ is a prerequisite for further research

Breaking In

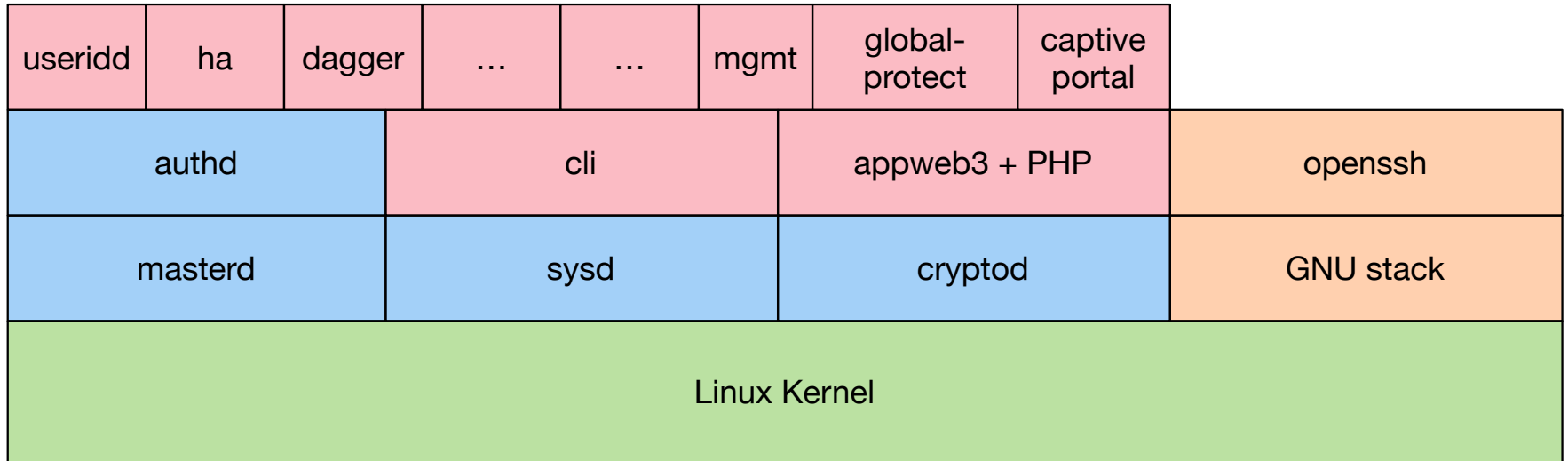
- CLI is restricted interface for configuration, troubleshooting
- Several commands are wrappers around standard Linux utilities
- Command line injection in test scp-server-connection:

```
test scp-server-connection initiate hostname "-  
oProxyCommand = chsh -s /bin/bash ernw" password b  
username c
```


PAN-OS Architecture

- Linux system running on MIPS64 processor
 - Cavium Octeon+ processor
 - 2.6.32 Kernel for PanOS 6.X
- Virtual appliances run on x64
- Network processing built on top of standard Linux capabilities
- Advanced features implemented as proprietary Linux daemons

PAN-OS Architecture



PAN-OS Architecture

- Web Interfaces are implemented on top of EmbedThis Appweb 3
 - Functionality is implemented as native PHP extensions called by small PHP wrapper scripts
- Three web server instances
 - Management Interface
 - GlobalProtect / SSL VPN
 - Captive Portal

Attack Surface



- Management Interfaces
 - Hopefully on isolated interfaces
- Content-, App-, User-ID
 - Untrusted network segments
- GlobalProtect / VPN
 - External (as in the Internet)

Management Web Interface



- Web UI for manual management
- REST API for automated access
- Implemented on top of Appweb3 + PHP environment
- Many features => Large attack surface
 - But most features require authentication

REST API

- REST API for automated management
- Can be reached with requests to `/api` URL
- POST requests will trigger call to native `apiWgetFilter` function
 - Unauthenticated 😊
- If request contains `client=wget`, `curl` is invoked to check authentication against internal service.

apiWgetFilter

- curl command escapes and uses following user supplied parameters:
 - “key” request parameter
 - HTTP Authentication Headers
 - Remote IP (from X-Real-Ip header if available)
- **escapeshellarg()** is used to escape values
 - Puts single quote before and after value
 - Escapes single quotes in value

Pseudo Code: apiWgetFilter

```
if key:
    if escapeshellarg(escaped_key, 1024, key) < 0:
        abort_connection
if basic_auth:
    if escapeshellarg(escaped_auth, 1024, basic_auth) < 0:
        abort_connection
if headers['HTTP_X_REAL_IP']:
    escapeshellarg(escaped_ip, 1024, headers['HTTP_X_REAL_IP'])
else:
    escapeshellarg(escaped_ip, 1024, remote_addr)
call_curl(escaped_key, escaped_ip, escaped_auth)
```


Pseudo Code: apiWgetFilter

```
if key:
    if escapeshellarg(escaped_key, 1024, key) < 0:
        abort_connection
if basic_auth:
    if escapeshellarg(escaped_auth, 1024, basic_auth) < 0:
        abort_connection
if headers['HTTP_X_REAL_IP']:
    escapeshellarg(escaped_ip, 1024, headers['HTTP_X_REAL_IP'])
else:
    escapeshellarg(escaped_ip, 1024, remote_addr)
call_curl(escaped_key, escaped_ip, escaped_auth)
```

PreAuth RCE in Management Web Interface

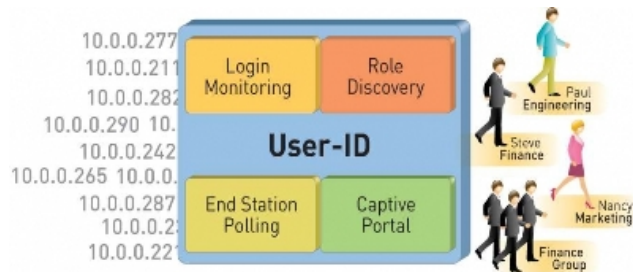
- Return value of `escapeshellarg()` is not checked for X-Real-Ip header
- How can the function fail?
 - Second argument is length of the output buffer → Max amount of bytes that can be written
- Overlong value: Closing single quote won't be written
- Off-by-One in quoting allows simple command injection in other values:
 - `key=; touch /tmp/ernw_poc;`

Demo

First Result

- Unauthenticated command execution against management web interface
 - 100% stable
- Hardened environments → Management interface won't be accessible for attackers
- Other attack surface is more interesting

User-ID



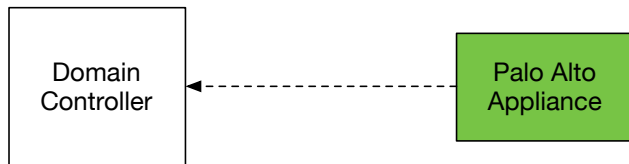
- Core selling point of Palo Alto devices
- Implement firewall policies based on user accounts (not IP addresses)
- Example:
 - User bob@corp can connect to DC on port 3389

<https://www.paloaltonetworks.com>

User-ID

- Firewall needs to have mapping between IP addresses and active user account.
- Five main ways:
 - Server Monitoring (agentless)
 - Server Monitoring (agent)
 - Captive Portal
 - Client Probing
 - Global Protect

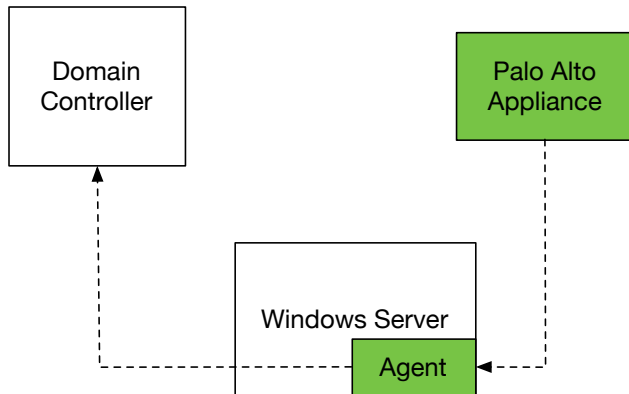
User-ID: Server Monitoring



- Assumption: AD based environment
- Agentless Monitoring
 - Create dedicated user for accessing domain controller (server operator permissions)
 - Store credentials on firewall
 - Firewall connects do DC / Exchange Server and reads event logs

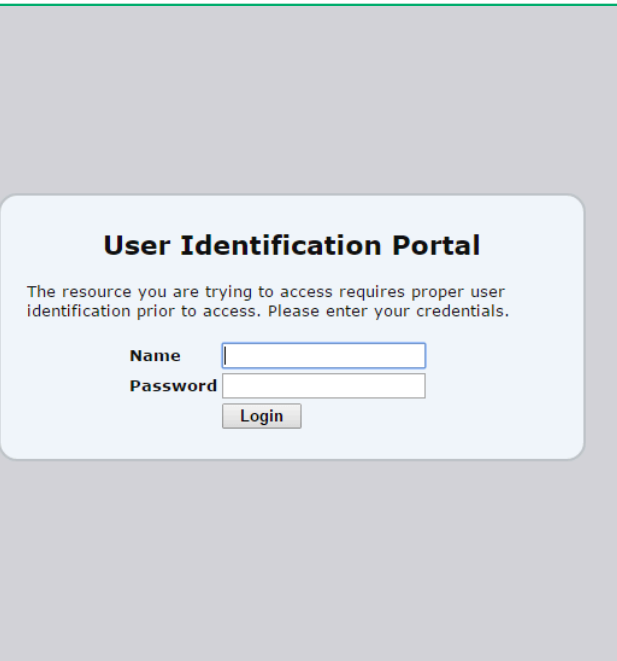
➔ Simple but stores credentials on device

User-ID: Server Monitoring



- Install User-ID Agent on Windows Server
 - Does not need to be the DC
- Configure domain account for agent.
- Agent connects to DC, Firewall connects to agent.
- For accepting connections from firewall User-ID Agent listens on TCP port 5007

User-ID: Captive Portal



User Identification Portal

The resource you are trying to access requires proper user identification prior to access. Please enter your credentials.

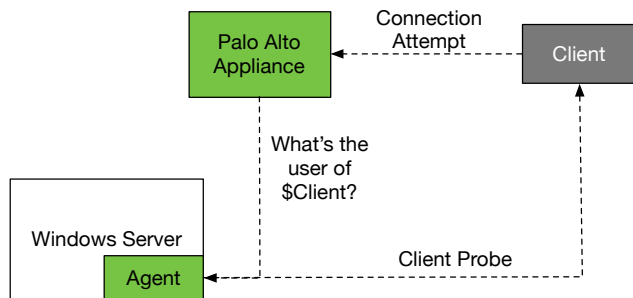
Name

Password

Login



- Addition/Alternative to server monitoring
- Hijack port 80 (+443) connections and force manual login
- Captive Portal is implemented using Appweb3 + PHP Extensions
 - Significant attack surface

User-ID: Client Probing






- Event Logs might be old, captive portal not feasible for non HTTP traffic.
- Idea: Just ask the client what user is logged in!
 - ... I did not say good idea
- Enabled by default
- Netbios and/or WMI


R7-2014-16: Palo Alto Networks User-ID Credential Exposure

 Blog-Eintrag wurde erstellt von [hdmoore](#)  in 14.10.2014

 Gefällt mir • 0

 Kommentar • 0

 [Project Sonar](#) tends to identify unexpected issues, especially with regards to network security products. In July of this year, we began to notice a flood of incoming SMB connections every time we launched the  [VxWorks WDBRPC](#) scan. To diagnose the issue, we ran the Metasploit [SMB Capture](#)  module on one of our scanning nodes and collected the results. After reviewing the data, we realized a common trend in the usernames of the incoming SMB connections.

After some digging, we traced this back to the Palo Alto Networks (PAN) [User-ID](#)  feature, an optional component provided by PAN that **"gives network administrators granular controls over what various users are allowed to do when filtered by a Palo Alto Networks Next-Generation Firewall"**. We contacted PAN and they confirmed that some of their customers must have misconfigured User-ID to enable the feature on external/untrusted zones. In summary, every time we triggered a PAN filter on a misconfigured appliance, our scanning node would receive an inbound authentication attempt by User-ID. This issue is not a vulnerability in the typical sense, but we felt that the impact was significant enough that it required notification and public disclosure.

Demo

GlobalProtect



- VPN solution with support for mobile devices
 - SSL-VPN/IPsec
 - Desktop Clients and Mobile Apps for popular platforms
- Can also be used internally
 - GlobalProtect authentication maps to Client-ID

GlobalProtect

- SSL-VPN and configuration APIs implemented on top of web interface
 - Appweb3 + PHP again 😊
- Very interesting attack surface
 - Remote (from the internet)
 - Some functionality does not require authentication

GlobalProtect: DoS

```
POST /global-protect/login.esp HTTP/1.1  
Host: 192.168.2.1  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 59487
```

```
prot=https%3A&server=192.168.2.1&&ok>Login&inputStr=&action=getsoftware&user=aa&passwd=A  
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA [ . . ]
```


GlobalProtect: DoS

- Password is passed to `unescapeStringForXml` which uses `alloca` to allocate space from stack.
- Stack size is heavily limited 😊 → Invalid memory access
- (Might be exploitable for more than DoS depending on the target system)

GlobalProtect: Static encryption keys

- GlobalProtect cookies are encrypted.
- Uses (shuffled) device master key as AES key
- By default: p1a2l3o4a5l6t7o8
 - No change enforced during installation
- Attack can create arbitrary faked cookies 😊
 - Allows for „interesting“ attacks against VPN authentication
- Not considered a security vulnerability by Palo Alto
- Recommendation: Change Device Master Key!
 - From us and admin guide!

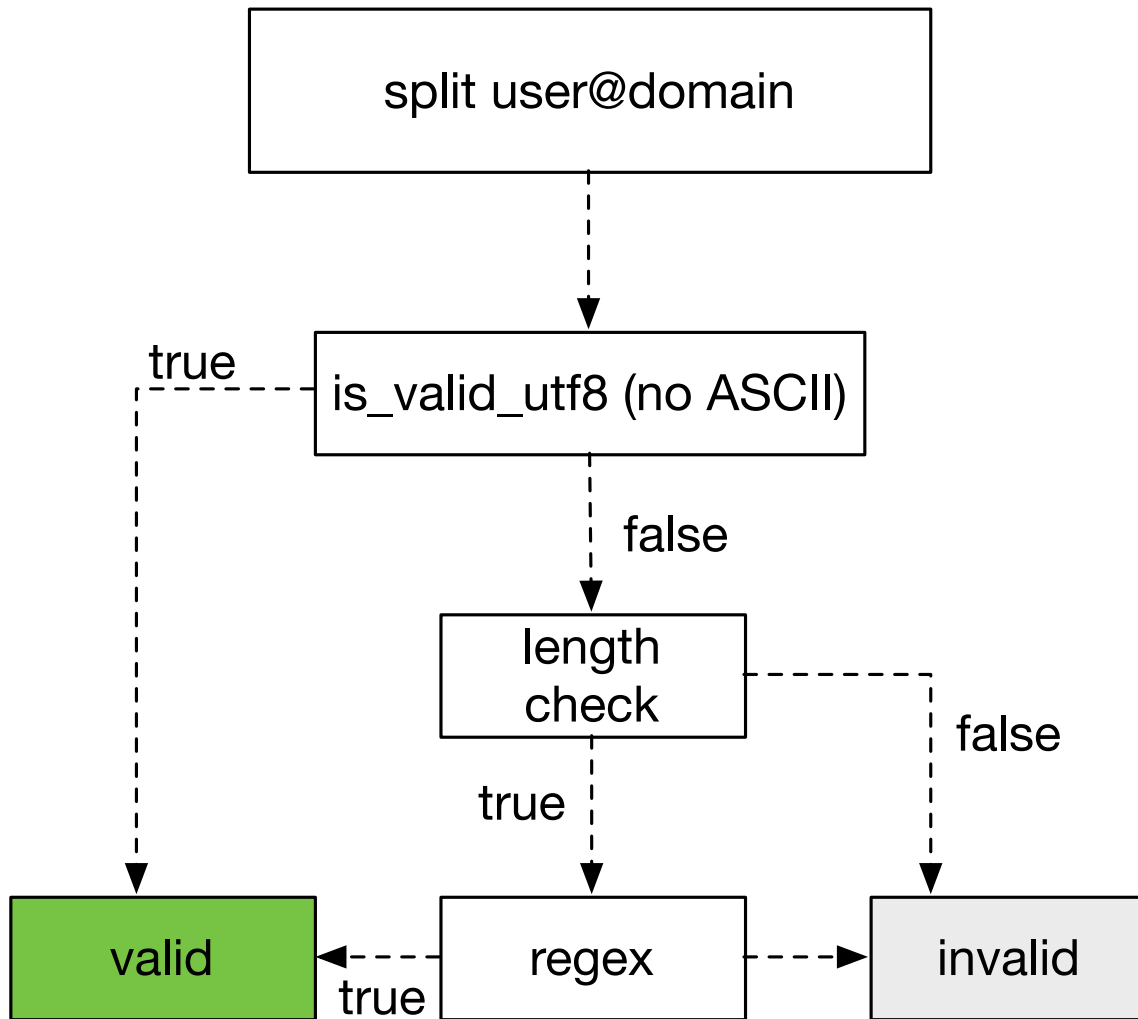
GlobalProtect: Getting Code Execution

- Goal: Remote unauthenticated compromise of the device
- Unauthenticated attack surface is limited
 - Most code directly calls into login functions
- Code uses `escapeStringForXml` function to escape username before sending XML encoded IPC message to authentication daemon.

GlobalProtect: escapeStringForXml

<xml />

- Function does not perform any length checks
- Destination is stack allocated buffer of size 1024
- To ensure that no overlong usernames are passed to function, `sslvpn_field_filter_check_user` is used.



sslvpn_field_filter_
check_user

sslvpn_field_filter_check_user

- If username/domain consists only of UTF-8 characters (and no ASCII) length check is skipped.
- Trivial DoS: Login with a username consisting of 10000 Ä
- RCE possible?



The Way To Code Execution

- Destination buffer is fixed size stack buffer
 - No stack canaries
- Executable without PIE
 - Very small helper binary that calls into main appweb3 library
 - Libraries use ASLR
- MIPS64
 - Big Endian (no partial overwrites)
 - eXecute Inhibit
 - pointers and address Space are 32bit
 - \$ra register (return address) is 64bit wide!

The Way To Code Execution

- First problem: Username can not contain any ASCII characters
- Can be partially bypassed by splitting username into `user@domain`
 - user is utf-8 string of arbitrary length
 - domain is alphanumeric ASCII string < 250
- Return Address overwrite?
 - \$ra is 64bit, upper half needs to be zero
 - Big Endian Overwrite + Alphanumeric ASCII == :(

The Way To Code Execution

- Pointer to PHP context is stored on stack
 - Used before function return for call to `php_body_write`
- Context has pointers to pointers to function pointer (double indirection)
- Problem: New value for context pointer needs to be alphanumeric
- Solution: Heap Spray

HeapSpray

– Appweb3 Heap Spray:

- Stores up to 1MB of arbitrary content until it finds „\r\n\r\n“
- Open many connections and send payload. Keep connections alive by repeatedly sending additional single-bytes

– Reliable allocates payload at:

- 0x31633130 or 1c10 in ASCII

PC Control: php_body_write

```
php_body_write:
lw      $a3, 0($a2)
lui     $a4, 0x61
addu    $a4, $t9
addiu   $a4, (unk_9A7)
lw      $v1, (output_)
lw      $v0, (output_)
addiu   $v0, -1
sll     $v0, 2
addu    $a3, $v0
lw      $v1, 0($a3)
lw      $t9, 0($v1)
jr      $t9
nop
```

\$a2 == 0x31633130

\$a3 == 0x31633134

\$v1 == 0x3163313C

\$t9 == ROP GADGET



\$pc to Code Execution

- Problem: Cavium Octeon+ support non executable memory → Heap spray is not executable
- ROP needed!
- MIPS64 Rop:
 - Aligned 4byte instructions → No accidental gadgets
- Only object at constant address is appweb3
 - Contains only 10 functions mostly wrapper that directly call into (randomized) shared libraries
- → No suitable ROP chain to get arbitrary execution of MIPS instructions discovered ☹️

\$pc to Code Execution

- But: Creation of arbitrary files possible:

```
lw      $t9, 0($s1)
addiu   $s0, 1
move    $a0, $s5
move    $a1, $s4
move    $a2, $s3
jalr    $t9
```

```
maStartLogging:
lui     $t7, 0x1010
lw      $t9, maStartLogging_ptr
addiu   $t8, $t7, (maStartLoggin
jr      $t9
```

- Control over \$s1 and \$s4 → Call to maStartLogging with arbitrary second argument
- maStartLogging creates a file at the path stored in the second argument

File Creation to Code Execution

- Needs another (local) bug 😊
- Includes a local privilege escalation to root.

Final Demo

Recommendations

- Isolate management interface
 - Very feature rich, hard to secure completely
- Think critically about relying on User-ID for security critical filtering
 - OK for business related policies or in combination with strong authentication (802.1X e.g.)
 - Not recommended for isolation of management interfaces
- Disable Client Probing
- Isolate User-ID Agent
- Change Master Password
- Keep System updated

Summary

- More features → Bigger attack surface → More vulnerabilities
- Very professional handling and response by Palo Alto
- Vulnerabilities are not great but response show right mindset → Positive about future progress

Thanks for your attention!

Q&A



@_fel1x



fwilhelm@ernw.de



Also visit our blog: <https://insinator.net>