

CHAPTER 7: SECURITY SERVERS

.....

NGX Security Servers inherit the folding process from previous versions of VPN-1. The HTTP Security Server provides URL screening and content checking (by incorporating CVP and UFP applications). Although more functionality from Security Servers is being incorporated into the kernel with each revision of VPN-1, troubleshooting specific Security Server processes can still indicate causes of issues.

Objectives

1. Identify different stages in the folding process.
2. Troubleshoot Security Server issues.
3. Debug Security Server



Key Terms

- Folding
- `fwssd`
- `fwauthd.conf`



THE FOLDING PROCESS

Overview

When an NGX kernel matches a connection to a security server rule, the kernel folds the connection to the relevant Security Server. Folding is the process how a Security Server redirects the packets. The Security Server opens a connection to the server to which the client tried to connect. The packet leaving the Security Server has the source IP of the NGX Security Gateway. The outbound kernel translates the source IP to the IP address of the client that originally opened the connection. If the client is configured in the Rule Base for Hide or Static NAT, the source IP is translated, as configured in the Rule Base.

If clients use the HTTP Security Server as a proxy, connections leave the Gateway with the Gateway's IP address as the source IP. No Network Address Translation (NAT) occurs.

TRANSPARENT CONNECTIONS

The default behavior of HTTP, FTP, and Telnet Security Server connections have been changed to transparent in VPN-1 NGX. Only the SMTP Security Server is still non-transparent by default. In other words, if no Hide or Static NAT is involved, and if the client does not set the Gateway as the proxy, packets leave the Gateway with the original client's IP address. The only exception is the SMTP Security Server: The packet leaves the Gateway with the source IP address as the Gateway's IP address, instead of the original client's IP address.

To change this behavior, modify the following properties from **true** to **false** in `$FWDIR/conf/objects_5_0.C`:

```
http_transparent_server_connection
```

```
ftp_transparent_server_connection
```

```
rlogin_transparent_server_connection
```

```
telnet_transparent_server_connection
```




4. The connection table is updated with two new entries, which allows the client following the packets to continue without examination:
<125.32.2.3,1234,180.3.42.3,80,TCP>

<125.32.2.3,1234, 125.32.0.1,8832,TCP >

INBOUND AFTER KERNEL

The packet is <125.32.2.3,1234, 125.32.0.1,8832,TCP>. The Security Server listening on port 8832 accepts and examines the packet. After the examination is done, the Security Server opens a new connection to the destination server. The new connection is recorded in table `PROXIED_CONNS`, with new connection properties (new port) and expiration time of 60 seconds, which means the Security Server must initiate a connection within that period.

The Security Server then sends the packet to its original destination using the `FWX_AUTH` table.

OUTBOUND BEFORE KERNEL

The packet is <125.32.0.1,8832, 180.3.42.3,80,TCP>. The Security Server initiates a connection. The source address is the Security Server and not the original client. The server returns the packet, destination port, and address to the Security server. The Security Server checks the `FWX_AUTH` table and a flag from the `CONN_OXID` table, to re translate the client's address and destination port.

OUTBOUND AFTER KERNEL

The packet is <125.32.2.3,1234,180.3.42.3,80,TCP>, which is the original connection.

Content-Security Rule Order

When setting up content-security rules, a rule allowing a connection from a Gateway to CVP server on port 18181 for the control connection is needed. The rule also must allow TCP high ports between the Gateway and CVP server. This is for the file transfer from the Gateway to the CVP server for file inspection.

Rules that specify CVP inspection do not replace rules that allow FTP, HTTP, or SMTP connections. Since NGX kernel examines the Rule Base sequentially, you must define rules in the appropriate order, to prevent unwanted traffic from entering your network.

Resource rules that accept HTTP, SMTP, and FTP connections must be placed before other rules that accept these services. If you define a rule that allows all HTTP connections before a rule that specifies CVP inspection on a URI resource, you may be allowing unwanted traffic. Similarly, CVP rules must be placed after rules that reject FTP, HTTP, or SMTP resource connections. For example, a rule rejecting large e-mail messages must come before a CVP rule allowing specific SMTP connections.

Security Server Default Messages

Default Security Server messages are in `$FWDIR/conf/spsc/spsc.en_us`, and can be edited manually. For example, the string...

```
CPSC_HTTP_FW_AT_HOST 1024 "FW-1 at #host#:" (host)
```

...Produces the following security-server message...

```
FW-1 at lothar: Access denied
```

...Where `lothar` has been defined as the Gateway. Changing the above string to...

```
CPSC_HTTP_FW_AT_HOST 1024 "Security Server at #host#:" (host)
```

...Produces the following Security Server message:

```
Security Server at lothar: Access denied
```



HTTP 1.0 and 1.1

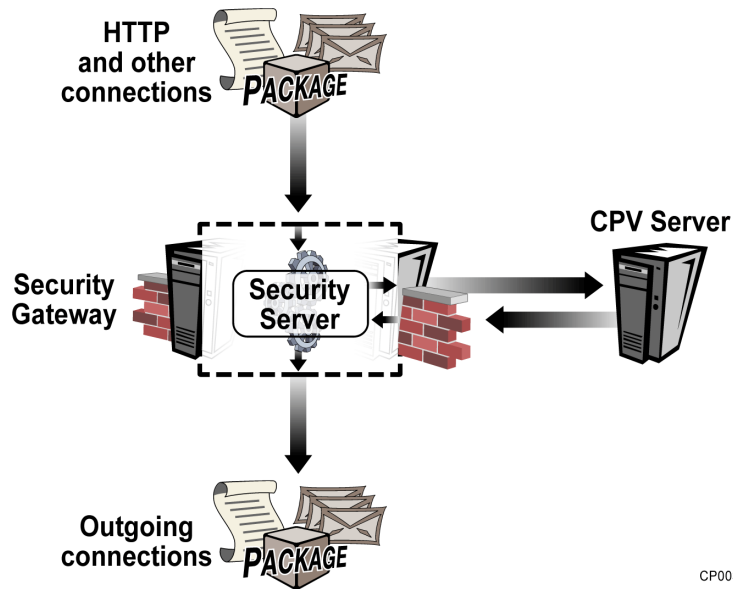
The following table lists differences between HTTP 1.0 and HTTP 1.1. This information can be useful when troubleshooting HTTP Security Server related issues.

Features	HTTP 1.0	HTTP 1.1
Connections	Keep-alive was not used.	Keep-alive is recommended.
Multiple requests per connection	Allowed, but the client cannot send multiple request; it must wait for each response to return before submitting another request.	Allowed; the client can send multiple requests, even before the first response has returned. The server has to return the responses in the same order they were sent.
Data end	Two ways: 1. Use the header-field content length. 2. Close the connection when the response is done.	Content length is obligatory.
Chunks	Not available	Chunking was introduced to allow the server to send responses with variable length without closing the connection. (In HTTP 1.0, this was the only way.)

TROUBLESHOOTING SECURITY SERVER ISSUES

The following steps help troubleshoot performance problems with HTTP Security Servers. The goal is to determine which object is responsible for performance issues (the HTTP Security Server, the CVP, server the machines themselves, and so on), when, and why.

The following is of a scenario where the HTTP Security Server is configured with a CVP server on a loaded network:



CP00332

HTTP Security Server in CVP Environment



Reviewing CPU and Memory

There is not an executable file for each Security Server. Instead, each security Server links to the **fwssd** executable. Under Windows NT, for example, looking at the Task Manager will not show the Security Server to which each process belongs. To find out which process belongs to each Security Server, proceed as follows:

- Look for the relevant Security Server's process identifier (PID) in the `$FWDIR/tmp` directory. For example, the HTTP Security Server PID will be written in the `in.ahhttpd.pid` file.
- Once you know the PID number, look for the number on the Windows Task Manager > Processes tab. On UNIX platforms, such as Solaris and SecurePlatform, the process number is found in `$FWDIR/tmp`. The CPU and memory use can be observed in real time by running the `top` command.

Editing fwauthd.conf

In some circumstances, adjusting the number of security servers spawned by **fwssd** may help in troubleshooting performance issues. This is done by editing the `fwauthd.conf` file. The `fwauthd.conf` file contains configuration information for all child processes started by NGX daemons, not only **fwssd**. When working with the `fwauthd.conf` file, ensure that you are only modifying entries relevant to the security servers for FTP, HTTP, HTTPS, or Telnet. Some process configurations (such as those for SMTP or clientless VPN) should not be modified unless under direct instruction by Check Point Technical Support. Take care to only modify the line relevant to the process you are troubleshooting.

`fwauthd.conf` EXAMPLE

A standard entry in `fwauthd.conf` looks like this:

# (port)	Parent Process	Child Process name	Wait	# (to be spawned)
80	fwssd	in.ahhttpd	wait	-5
259	fwssd	in.aclientd	wait	259

In this example, two server processes are configured to spawn when the NGX kernel detects traffic relevant to those servers. The first entry shows that the HTTP Security Server is configured to spawn up to five child processes (`in.ahhttpd`) from parent process `fwssd`. To increase the number of `in.ahhttpd` processes that spawn, modify the number in the final column. The preceding dash “-” is used to indicate that a set amount of processes will spawn. When this number is 0, the parent process will spawn the child processes as needed.



Adjusting the number of processes should be done with great care, as spawning too many processes may increase the number of file descriptors beyond the limits of kernel memory and cause that process to stop functioning.

When adjusting the number of processes being spawned, remember that each process, parent or child, gets 1024 file descriptors per process. Spawning too many processes will result in kernel error messages, indicating too many open files in the log file of the relevant process.

The second entry shows that Client Authentication will spawn only one process (indicated by the fact that the port number is written in the final column, without a preceding dash) with the name `in.aclientd`.

Listing Possible Causes

It is worth defining the possible causes of problem. Assume every one of the involved objects can be a cause of the problem, and the problem may arise from a combination of causes. Possible causes for each object include the following:

MACHINES

- Overloaded CPU
- Memory issue
- Running out of file descriptors

NGX GATEWAY ISSUES

- Limitation of kernel tables
- A loaded kernel blocking Security Servers



SECURITY SERVERS

- A general Security Server issue
- A Security Server with a CVP/UFPP resource issue
- CVP server
- Limitation of hash tables

CVP SERVER

- Overloaded CPU
- Memory issue
- Possible known/unknown issue

Identifying Issue Source

One of the best ways to understand where the issue lies is by eliminating possibilities:

1. Change the rule so the HTTP resource is not used. Replace it with a standard HTTP service. This way HTTP connections are passed through the kernel and not folded to the Security Server. If this solves the problem, the problem is with the HTTP Security Server: Proceed with step 3. If it does not solve the problem, proceed with step 2.
2. Change the rule to use the HTTP resource again, instead of the standard HTTP service. Do not configure the resource with the CVP server. Under this configuration if the problem does not exist, you know the issue is with the interaction with the CVP server.

3. When the problem occurs, run the following:

- `top` (on UNIX) or Task Manager (on Windows)

Notice which process number is in charge for CPU and memory use. Check `$FWDIR/tmp` to find the PID of relevant Security Server process.

- `lsof` (on Solaris)

Run this command to check how many file descriptors are open:

```
lsof | grep <process name> | wc -l
```

- `fw tab -s`

Examine the firewall kernel-tables counts. Look for a table with a high count.

- `snoop` or `fw monitor`

The output should show if connections are being folded to the Security Server, and if the Security Server is passing them to the CVP server.

- Save the log and `ahttpd.log` files

Check the log file for any drops, or information about connections in the Info column. Check the `ahttpd.e1g` file for any error messages.

Analyzing Results

From the results, you should be able to determine:

- The faulty object.
- A measurement of the load (by using accounts, logs, snoops) and the network view (using snoops).
- The state of machine resources.
- Whether the problem is with NGX Gateway or CVP server limitations, or both.



DEBUGGING SECURITY SERVERS

To debug a Security Server the relevant process must be running. Before starting the debug verify that the process you wish to debug has a current PID in the `$FWDIR/tmp` directory. If the process has no PID, the following error will appear: “Cannot find process id for (in.aclientd)”

Check Point recommends debugging all processes by on the active process. In circumstances where the process is not starting correctly, stop VPN-1 NGX, set the environment variables for debugging, and then restart VPN-1 NGX.

TD_ERROR_ALL_ALL Flag

When configuring a debugging session, whether for a running process or setting an environment variable for a restarted session, it is important to remember to set the environment variables for that debugging session. While each Security Server will have specific flags relevant to their functionality, all debugging will require a `TD_ERROR_*_*` flag to be set.

The `TD_ERROR_ALL_ALL` flag (most often seen when configuring debugging as `set TD_ERROR_ALL_ALL=3`) essentially tells the process being debugged the level of information to write to the output file (typically `processname.elg`).

The numeric value is a “verbosity level” between 1 and 5, where 1 is the minimum amount of information to be written, with 5 being maximum verbosity. Check Point recommends setting the verbosity level to 3 or 4, as this will often provide enough information for troubleshooting an issue.

`TDERROR_*_*` is also used to configure specific debugging sequences, as shown in the following sections. Each of the following sections are the standard commands for enabling debugging on a running process, sorted according to the specific Security Server.

FTP Security Server

To enable debugging all platforms, run:

```
fw debug in.aftpd on | off FWAFTPD_DEBUG 3
```

Output is automatically redirected to `$FWDIR/log/aftpd.elg`.



Some OPSEC applications may not be able to be debugged during operation as in these previous examples. In these situations, define the environment variables to monitor OPSEC application information:

1. Stop the FireWall-1: `cpstop`
2. Run the command: `set OPSEC_DEBUG_LEVEL 3`
3. Run the FireWall-1 command: `fw`
4. Start the FireWall-1: `cpstart`

HTTP Security Server (Client Authentication on Port 259 Telnet)

To enable debugging on all platforms, run:

```
fw debug in.aclientd on | off FWACLIENTD_DEBUG 3
```

Output is automatically redirected to `$FWDIR/log/aclientd.elg`.

HTTP Security Server (Client Authentication on Port 900 HTTP)

To enable debugging on all platforms run:

```
fw debug in.ahclientd on | off FWACLIENTD_DEBUG=3
```

Output is automatically redirected to `$FWDIR/log/ahclientd.elg`.

SMTP Security Server

There is *no need* to stop the FireWall. Check that files `$FWDIR/temp/mdq.pid` and `$FWDIR/tmp/in.asmtpd.pid` exist.

If they do not, you will not be able to run this debug command:

```
fw debug mdq on TDERROR_ALL_MDQ_APP_DBG=1
```

```
fw debug in.asmtpd on
```

To stop debugging, run:

```
fw debug mdq off TDERROR_ALL_MDQ_APP_DBG=1
```

```
fw debug in.asmtpd off
```

Multiple Security Server Troubleshooting

As mentioned previously, some configurations may require multiple processes to be spawned for Security Servers. In circumstances when debugging is enabled using `fw debug <process> on`, only the Security Server process with the PID defined in `$FWDIR/tmp/<process>.pid` will write to the relevant `*.elg` file. It is best to stop VPN-1 NGX and configure the environment variables so that on restart *all* desired processes will write to their `*.elg` files:

1. Stop the FireWall-1: `cpstop`
2. Run the command: `set <process_name>_<debug>_LEVEL n`
3. Run the command: `set TDERROR_ALL_<process_messages> n`
`n` is the level of detail desired.
4. Run the FireWall-1 command: `fwd`
5. Start the FireWall-1: `cpstart`

ALTERNATE METHOD

Or use the following method:

1. Enable debugging on fwd with the following command:

```
fw debug fwd on [options]
```

2. Kill all `ahhttpd` processes. When the new processes start, they will start in debug mode, and all daemons will print to the `ahhttpd.elg` file.