FortiGate Proxy Splice and Client Comforting Technical Note

1. Introduction

This document explains the motivation behind, and particular behaviours of, the **splice** and **client comforting** features present in the FortiGate proxies. Proxies exist for the following protocols:

- HTTP (client comforting)
- FTP upload (splice)
- FTP download (client comforting)
- SMTP (splice)
- POP (splice)
- IMAP (splice)

Throughout this document **client** designates a machine inside the firewall, and **server** designates a machine outside the firewall. Note that neither class of machine is implicitly trusted; data sent both from the client to the server and from the server to the client is examined by the proxies.

2. Motivation

In the following example, the client makes a request and expects to receive data in response. This is for illustration purposes only; in some protocols the roles are reversed: that is, the server receives the bulk of the data in the transaction.

From the point view of the client, this is what happens:
1. client connects to server and makes request
2. server sends back response.

In fact, the transparent proxy between the client and the server intercepts all connections, requests and responses. The proxy buffers and scans the server's response before flushing it to the client. While buffering and flushing the naive proxy implementation sends no information to the client and server, respectively.

A problem arises if the server response is large, or the proxy to server or proxy to client connection is slow: the buffering or flushing stage can take a relatively long time. This delay can be longer than the minimum timeout dictated by the application protocol. Also, some clients do not follow the standards and will close a connection even before the minimum timeout interval has elapsed. The client therefore closes the connection without receiving the response.

The solution is to send some of the server's response to the client while buffering. This keeps the client from timing out and closing the connection. This is exactly what **splice** mode is in the FortiGate proxies.

If the proxy is in **splice** mode it sends some of the server response to the client while buffering it. The final part is withheld from the client while the proxy scans the completely buffered response. If the response is clean, the final part is sent to the client. If the response is infected, the client and server connections are closed after sending any appropriate error responses or message substitutes. Depending on the details of the application protocol, the client either discards the incomplete response or accepts the substituted infection notification.

3. Current FortiGate Proxy Behaviour

This section describes the current splice strategies/features of the FortiGate proxies.

3.1. FTP

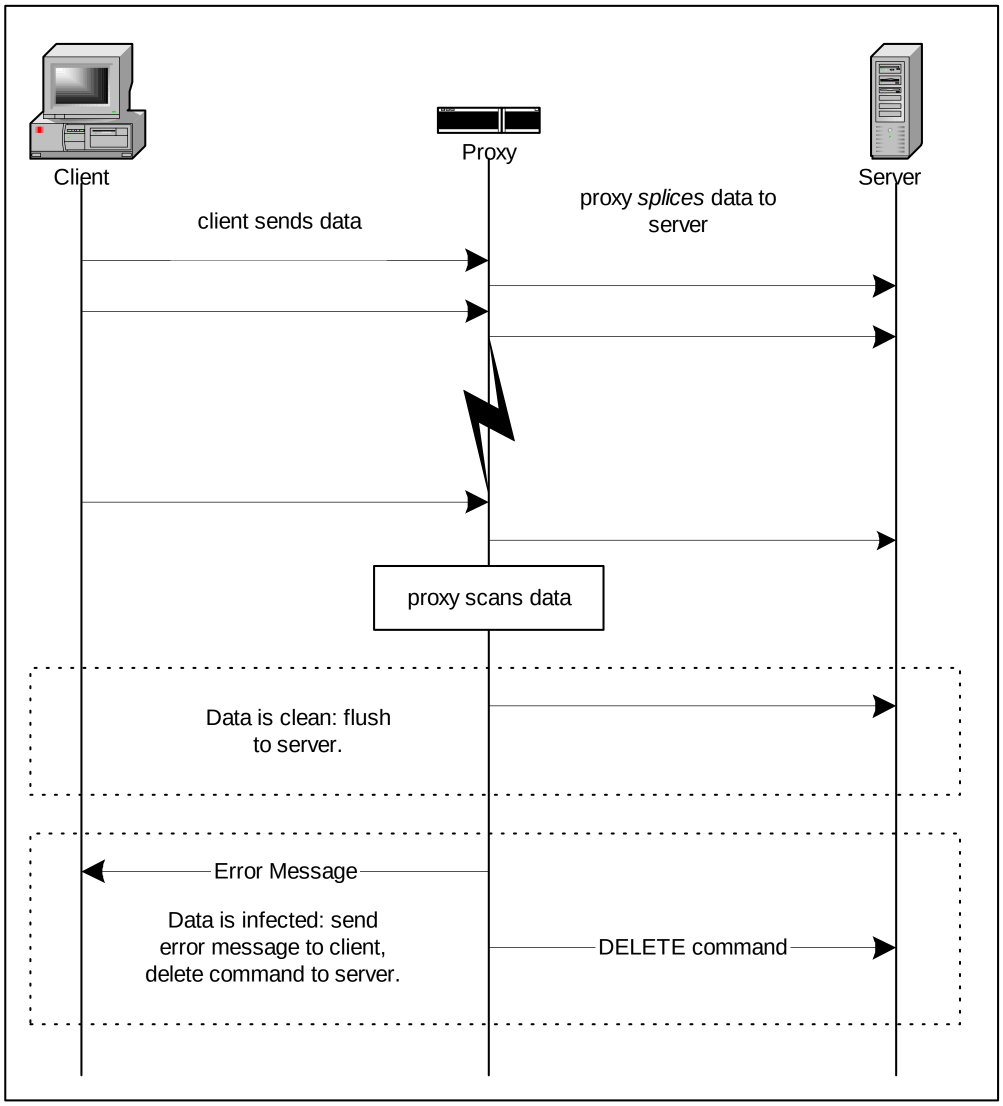
The FTP proxy uses two separate splice strategies. One is for client to server uploads, and one is for downloads from the server to the client.

3.1.1. FTP Upload

The client sends a put command and then begins to send the file. The following table and diagrams show the behaviour of the FTP proxy during upload both with and without splice enabled. The table also enumerates some advantages and disadvantages of the splice mechanism. This splice behaviour is controlled by a configuration option.

	splice enabled	splice disabled
while buffering	The proxy sends the file to the server while simultaneously buffering it. Withholds the final part while scanning.	The proxy buffers the file and does not send anything to the server.
timeout issues	None.	The server can timeout and close the connection while the proxy is buffering the file from the client.
if clean	The proxy flushes the remainder of the file, and leaves the file on the server.	The proxy flushes the buffered file to the server.
if infected	The proxy removes the file from the server by sending a delete command on behalf of the client. **Note:** The delete command may not succeed with all FTP servers or FTP server configurations.	The proxy discards the buffered file and sends an error message back to the client.

resource issues	This splice method wastes resources if the message is infected. The bandwidth between the proxy and the server and the resources of the server are used to transfer and store, respectively, a file which is immediately deleted.	None.



Client
Proxy
Server
client sends data
proxy *splices* data to server
proxy scans data
Data is clean: flush to server.
Error Message
Data is infected: send error message to client, delete command to server.
DELETE command

Figure 1 FTP Upload with splice enabled

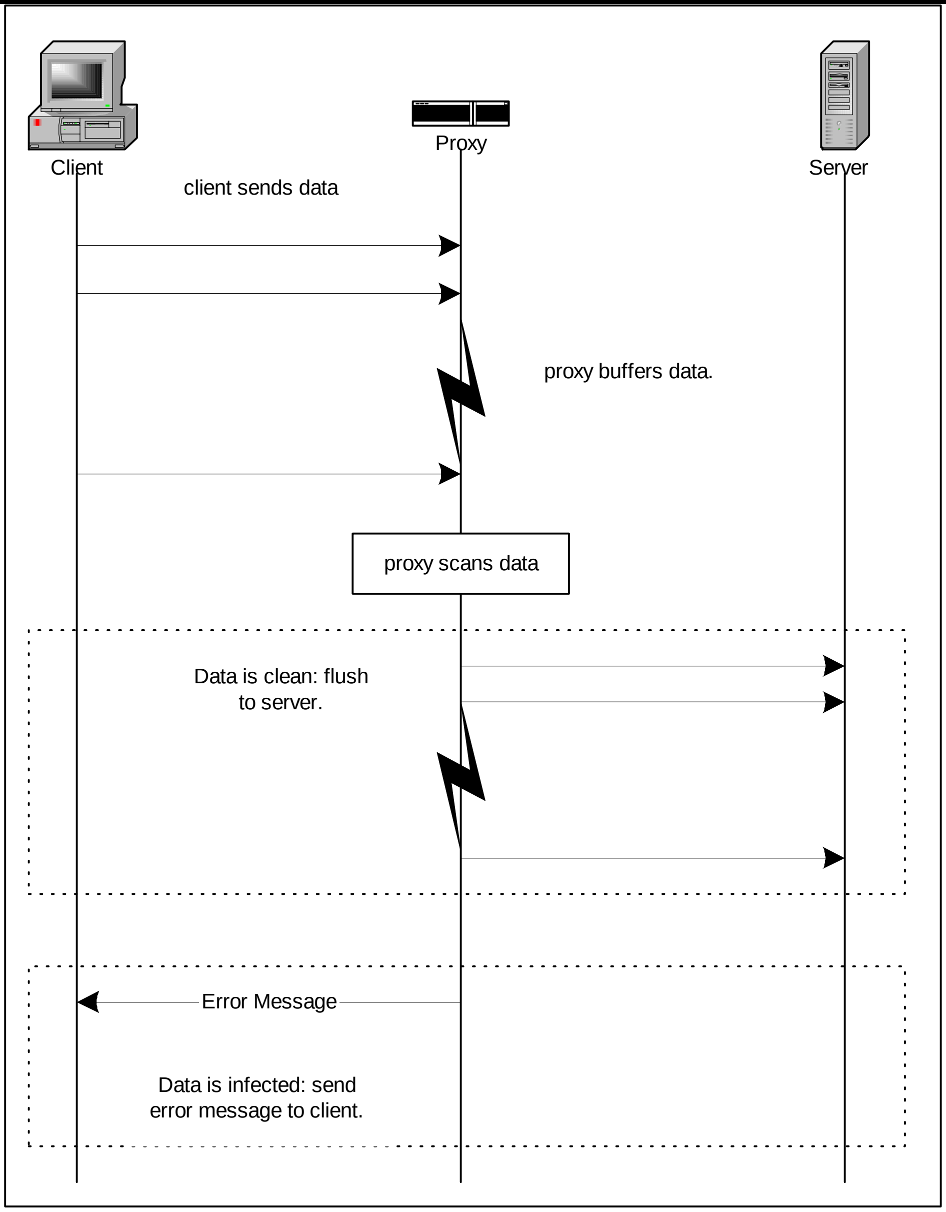


Figure 2 FTP Upload with splice disabled

3.1.2. FTP Download

See FTP Download in the Client Comforting section.

3.2. HTTP

See HTTP in the Client Comforting section.

3.3. SMTP

The SMTP proxy uses splice in the client to server direction, since the client is the source of message data. This splice behaviour can be enabled and disabled with a configuration option.

After sending the DATA command and receiving the go-ahead from the server, the client begins submitting the message. The following table and diagrams show the behaviour of the SMTP proxy both with and without splice enabled. The table also enumerates some advantages and disadvantages of the splice mechanism.

	splice enabled	splice disabled
while buffering	The proxy sends message parts to the server while simultaneously buffering them. Withholds the final **.** (dot).	The proxy buffers the message and does not send anything to the server.
timeout issues	None.	The server can timeout and close the connection while the proxy is buffering the message from the client.
if clean	The proxy flushes the final **.** (dot) to the server, completing the transfer.	The proxy flushes the buffered message to the server.
if infected	The proxy closes its connection with the server without sending the final **.** (dot). The server drops the incomplete message submission. The proxy sends a **5xx permanent error** to the client (therefore the client will not try to resubmit the message).	The proxy replaces any infected portions with the appropriate notifications and flushes the message to the server.
resource issues	This splice method wastes resources if the message is infected. The bandwidth between the proxy and the server is used to transfer the contents of a message which will not be sent, and the resources of the server itself are consumed until the message is dropped.	None.
infection notification	The sender of the message (the client) immediately receives a **5xx** error if the message is infected.	The sender of the message is not told that the message was infected. The recipient of the message receives a message with the infected portions removed.

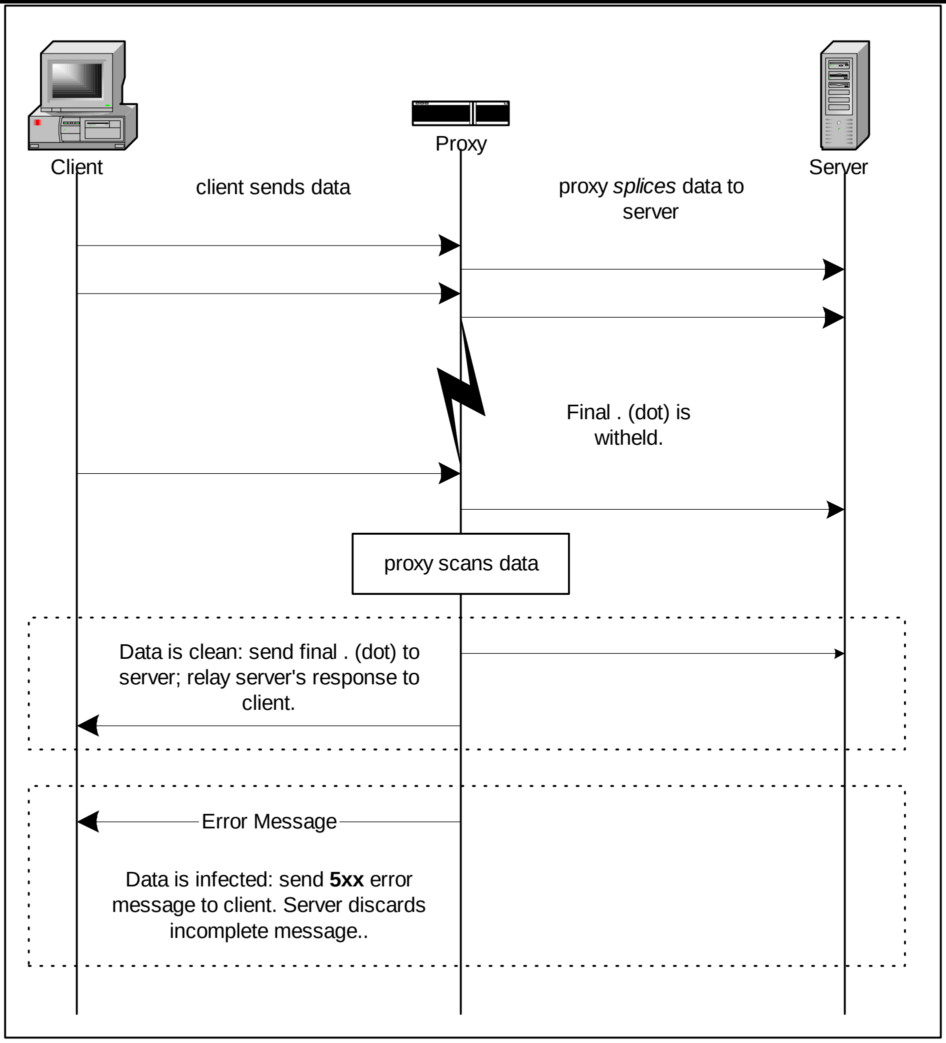

Client
Proxy
Server
client sends data
proxy *splices* data to server
Final . (dot) is witheld.
proxy scans data
Data is clean: send final . (dot) to server; relay server's response to client.
Error Message
Data is infected: send **5xx** error message to client. Server discards incomplete message..

Figure 3 SMTP with splice enabled.

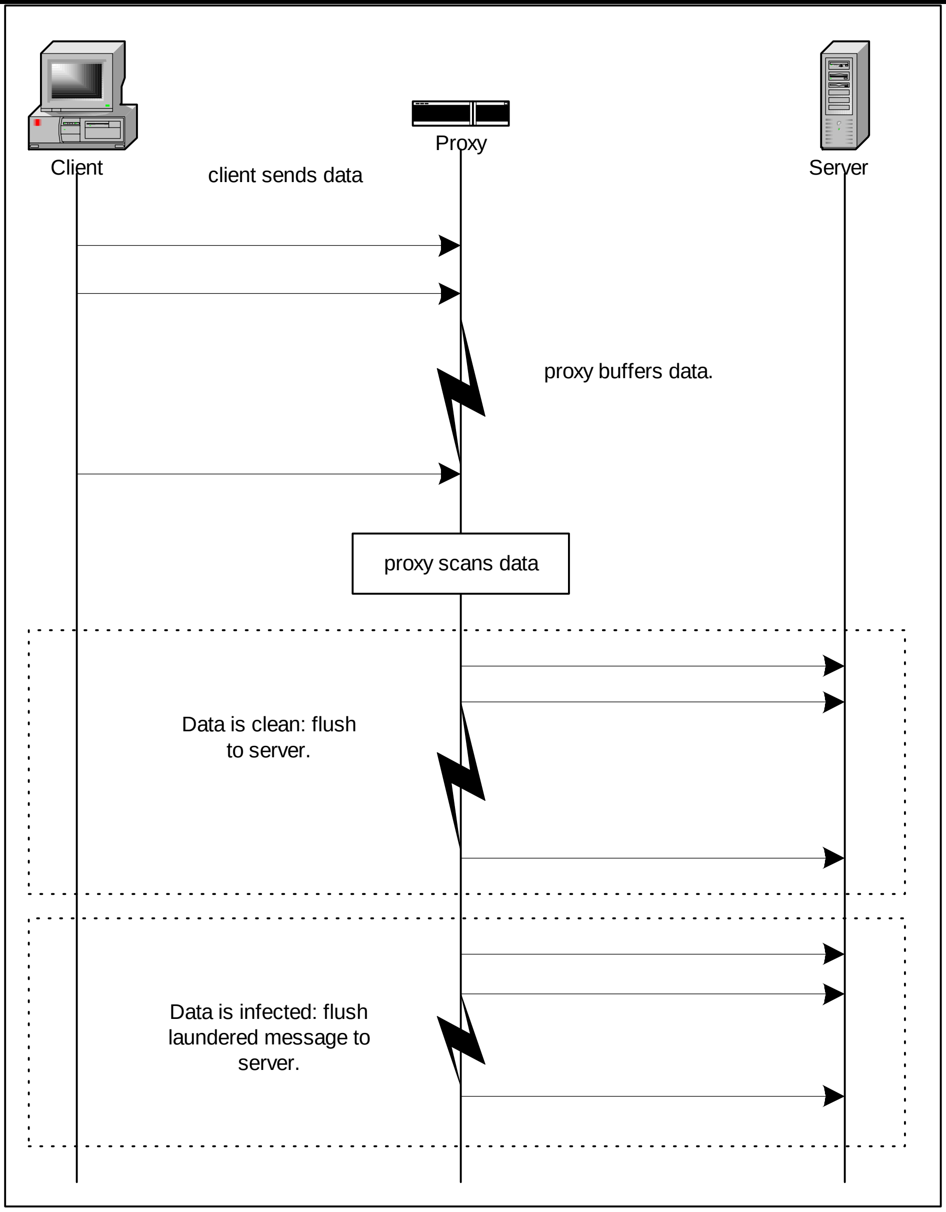

Client
Proxy
Server
client sends data
proxy buffers data.
proxy scans data
Data is clean: flush to server.
Data is infected: flush laundered message to server.

Figure 4 SMTP with splice disabled.

3.4. IMAP

The IMAP proxy uses splice in the client to server direction when the client sends an APPEND command, since the client is the source of message data in this case.

The client starts sending a message after sending an APPEND command. The following table describes the IMAP splice strategy. This behaviour is not configurable.

	Behaviour
while buffering	The proxy buffers the message as it receives it from the client. While buffering, the proxy sends NOOPs to the server. The proxy scans the message once it has been completely buffered.

if clean	If the message is clean, it is flushed to the server unmodified.
if infected	If any parts of the message are infected, an error response is sent to the client. Nothing is sent to the server.

3.5. POP

The POP proxy uses splice in the server to client direction, since the server is the source of message data.

The server starts sending a message in response to a retrieve command from the client. The following table describes the POP splice strategy. This behaviour is not configurable.

	Behaviour
while buffering	The proxy buffers the message as it receives it from the server. While buffering, the proxy sends the message headers to the client one line at a time to keep the client from timing out. If the proxy is still buffering once all of the headers have been sent in this way, it sends an additional *fake header*. This meaningless header, `X-vscanner: inserted by scanner`, is extended with the text `(processing…)` every N seconds while the proxy continues to buffer the message body from the server. The client ultimately ignores this header and under normal circumstances the user does not see it, but it keeps the client from timing out. The proxy scans the message once it has been completely buffered.
if clean	If the message is clean, it is flushed to the client unmodified.
if infected	If any parts of the message are infected, these parts are replaced with the appropriate notifications, and the modified message is flushed to the client.
while flushing	While the proxy is flushing the message to the client, it periodically sends the **NOOP** command to the server to keep it from timing out.

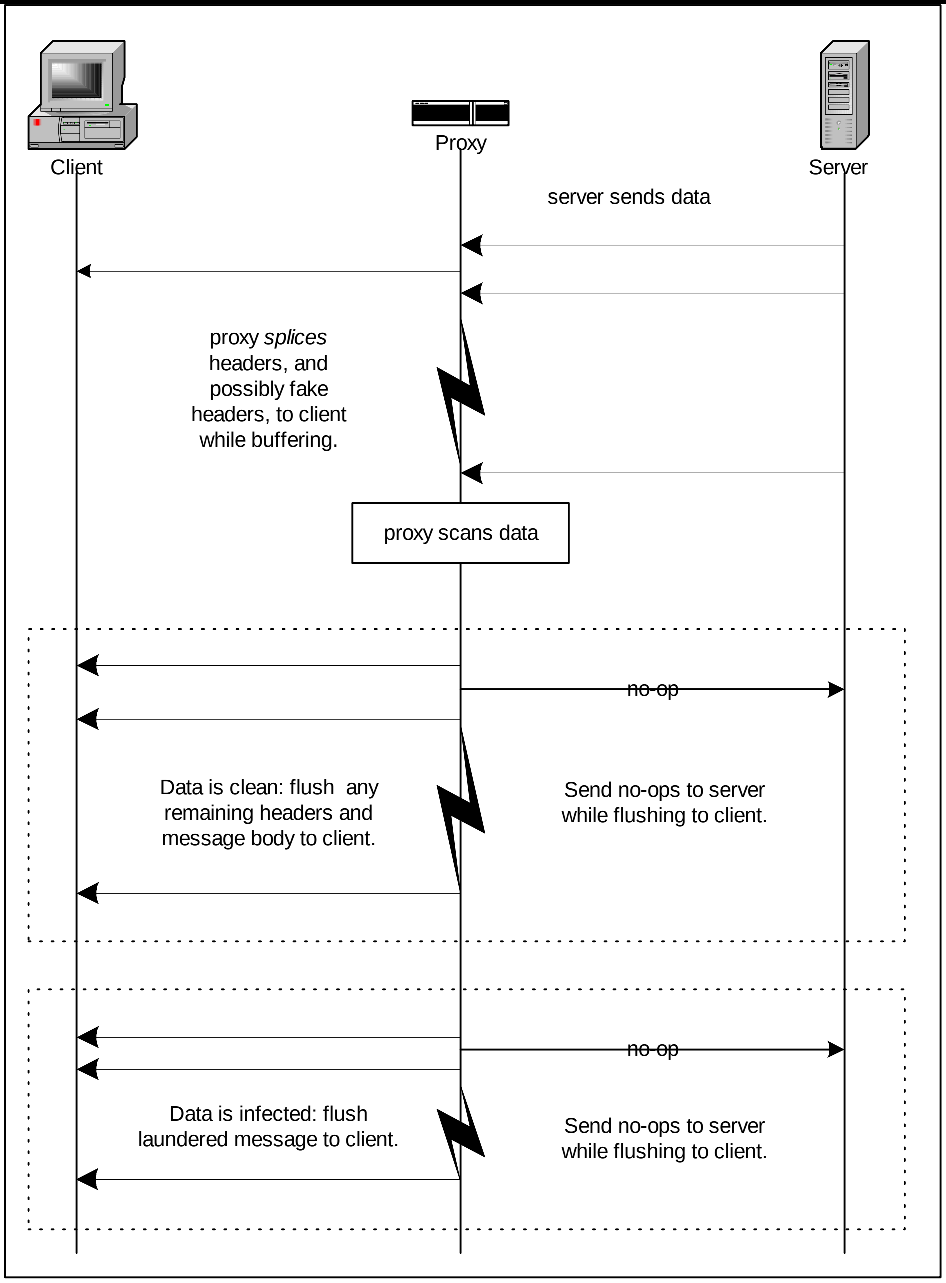

Client
Proxy
Server
server sends data
proxy *splices* headers, and possibly fake headers, to client while buffering.
proxy scans data
no-op
Data is clean: flush any remaining headers and message body to client.
Send no-ops to server while flushing to client.
no-op
Data is infected: flush laundered message to client.
Send no-ops to server while flushing to client.

Figure 5 POP Download.

4. Client Comforting

This section describes the **client comforting** feature of the FortiGate FTP and HTTP proxies. The feature is most easily described by explaining the configuration options that control it:

Configuration Option	**Meaning**
comfort_interval (seconds)	The number of seconds the proxy waits before client comforting begins. Eg if set to

	20, the proxy will start sending *comfort* bytes to the client only after it has been buffering for 20 seconds. *Comfort* bytes are sent every 20 seconds.
comfort_amount (bytes)	The number of bytes that the proxy will send to the client as *comfort*. Eg if set to 512, the proxy will send 512 bytes to the client at each comfort interval.

Let the configuration be **comfort_interval=20, comfort_amount=512** in the following examples.

4.1. FTP Download

1. client requests 10MB file
2. proxy buffers file from server. The connection is slow, so after 20 seconds about one half of the file has been buffered.
3. proxy continues buffering file from server, but sends 512 bytes to the client.
4. after 20 more seconds, the proxy sends the next 512 bytes of the buffered file to the client.
5. when the file has been completely buffered, the client has received **ca * (T/ci)** bytes == 512 * (40/20) == 512 * 2 == 1024 bytes, where **ca** is the comfort_amount, **T** is the buffering time and **ci** is the comfort_interval.
6. if the file is clean, the remainder is flushed to the client. If the file is infected, the data connection is closed an an error message is sent to the client.

4.2. HTTP

1. client requests 10MB file
2. proxy buffers from server. The connection is slow, so after 20 seconds about one half of the file has been buffered.
3. proxy continues buffering file from server, but sends 512 bytes to the client.
4. after 20 more seconds, the proxy sends the next 512 bytes of the buffered file to the client.
5. when the file has been completely buffered, the client has received **ca * (T/ci)** bytes == 512 * (40/20) == 512 * 2 == 1024 bytes, where **ca** is the comfort_amount, **T** is the buffering time and **ci** is the comfort_interval.
6. if the file is clean, the remainder is flushed to the client. If the file is infected, the connection is closed. The client receives no error message.

4.3. HTTP Complications

HTTP is a simple stateless protocol with no out of band communication channel to send error messages to the user. When client comforting occurs with an infected file, the only action the FortiGate can take is to close the connection: no error message can be sent to the client because some of the actual file has already been sent. The user gets no feedback other than a prematurely completed download and a truncated file.

If the user tries to download the file again, the browser may attempt to resume the download by sending an HTTP Range request . The response will likely be allowed through because only complete responses can be scanned. In FortiOS, HTTP Range requests can be blocked, forcing the client to download the entire file from the beginning.

To avoid the frustrating user experience of repeated truncated downloads with no explanation, the URLs of infected files are stored in a fixed size cache (older entries will expire). If a client makes an HTTP request for a URL which is in the cache, an informative error response similar to the existing Infection Replacement message is sent to the client.

5. Risks

By design, this feature sends unscanned and therefore potentially infected content to the client. The configuration options must be adjusted for your particular environment (link speeds, etc) and risk tolerance.

6. Configuration

6.1. Configuring FTP and SMTP splice

Use the following CLI commands to configure FTP splice:

```
config firewall profile
 edit <profile_name>
      set ftp splice
 end
```

Use the following CLI commands to configure SMTP splice:

```
config firewall profile
 edit <profile_name>
      set smtp splice
 end
```

6.2. Configuring FTP and HTTP client comforting

In the Web-based Manager, go to **Firewall** > **Protection Profile**. When creating a new profile or editing an existing one, configure the client comfort settings under **Anti-Virus**.

Or use the following CLI commands:

```
config firewall profile
 edit <profile_name>
      set ftpcomfortinterval 20
      set ftpcomfortamount 512
      set ftp clientcomfort
 end


config firewall profile
 edit <profile_name>
      set httpcomfortinterval 20
      set httpcomfortamount 512
      set http clientcomfort
 end
```