US Patent & Trademark Office
Patent Public Search | Text View

United States Patent Application Publication 20250267138
Kind Code A1
Publication Date August 21, 2025
Inventor(s) Brady; Justin M. et al.

PROXY-LESS SECURE SOCKETS LAYER (SSL) DATA INSPECTION

Abstract

Some embodiments of proxy-less Secure Sockets Layer (SSL) data inspection have been presented. In one embodiment, a secured connection according to a secured network protocol between a client and a responder is setup via a gateway device, which is coupled between the client and the responder. The gateway device transparently intercepts data transmitted according to the secured network protocol between the client and the responder. Furthermore, the gateway device provides flow-control and retransmission of one or more data packets of the data without self-scheduling the packet retransmissions using timeouts and based on the packet retransmission logic of either the client-side or the responder side of the connection. The gateway device is further operable to perform security screening on the data.

Inventors: **Brady; Justin M. (Livermore, CA), Dubrovsky; Aleksandr (San Mateo, CA), Yanovsky; Boris (Saratoga, CA)**

Applicant: **Sonicwall US Holdings Inc.** (Milpitas, CA)

Family ID: **1000008574859**

Appl. No.: **19/197669**

Filed: **May 02, 2025**

Related U.S. Application Data

parent US continuation 17009606 20200901 PENDING child US 19197669
parent US continuation 15685768 20170824 parent-grant-document US 10764274 child US 17009606
parent US continuation 12497328 20090702 parent-grant-document US 9769149 child US 15685768

Publication Classification

Int. Cl.: **H04L9/40** (20220101); **H04L9/32** (20060101)

U.S. Cl.:

CPC **H04L63/0823** (20130101); **H04L9/321** (20130101); **H04L9/3263** (20130101); **H04L63/0281** (20130101); **H04L63/0884** (20130101); **H04L63/1408** (20130101); **H04L63/166** (20130101);

Background/Summary

CROSS REFERENCE TO RELATED APPLICATIONS [0001] This application is a continuation of U.S. patent application Ser. No. 17/009,606 filed Sep. 1, 2020, which is a continuation of U.S. patent application Ser. No. 15/685,768 filed on Aug. 24, 2017, now U.S. Pat. No. 10,764,274, which is a continuation of U.S. patent application Ser. No. 12/497,328 filed on Jul. 2, 2009, now U.S. Pat. No. 9,769,149, the entire contents of each of which are incorporated herein by reference.

TECHNICAL FIELD
[0002] The present invention relates to intrusion detection and prevention in a networked system, and more particularly, to providing proxy-less data inspection.
BACKGROUND
[0003] FIG. **1** illustrates a current networked system **100**. Conventionally, to make a secure connection between the client **110** and a server **120**, the following operations are performed. A web browser on the client is configured to point to a proxy Internet Protocol (IP) address for Hypertext Transfer Protocol Secured (HTTPS) connections. An initial CONNECT request with full Universal Resource Locator (URL) is sent by the client **110** to a proxy **130** between the client **110** and the server **120**. The proxy **130** connects to the HTTPS server **120** using the full URL provided in the client's **110** request. The HTTPS server **120** sends back a certificate. The proxy **130** strips out relevant information from the certificate (e.g., common name, etc.) and creates a new certificate signed by a certification-authority certificate, which the user of the proxy **130**, i.e., the client **110**, has indicated to trust. Eventually, the newly generated certificate is passed to the client **110** and the client **110** accepts the certificate.
[0004] Data is decrypted on one connection, and clear-text (i.e., decrypted data) is inspected. Then the data is re-encrypted when sent on another connection. As a result, two TCP/SSL connections **115** and **125** are established, namely, a first connection **125** between the proxy **130** and the server **120**, and a second connection **115** between the client **110** and the proxy **130**, where each connection supports full Transmission Control Protocol (TCP) flow-control logic. Packet loss re-transmissions are handled individually for each connection and all retransmission scheduling is done on the proxy **130**.
[0005] One disadvantage of the above scheme is that the client's **110** browser has to be configured with the proxy's IP address. The above scheme is not so scalable due to full TCP based flow control implemented on the inspecting device and due to the fact that sockets do not scale well for large number of connections. Furthermore, it is difficult to configure for non-HTTP protocols.

Description

BRIEF DESCRIPTION OF THE DRAWINGS

[0006] The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which:

[0007] FIG. **1** illustrates a conventional networked system with a proxy.

[0008] FIG. **2** illustrates one embodiment of a proxy-less system.

[0009] FIG. **3** illustrates one embodiment of a method to establish a secured connection between a client and a responder without a proxy.

[0010] FIG. **4** illustrates one embodiment of a method to dynamically generate a certificate.

[0011] FIG. **5** illustrates one embodiment of a method to perform proxy-less data inspection.

[0012] FIG. **6** illustrates a block diagram of an exemplary computer system, in accordance with one embodiment of the present invention.

DETAILED DESCRIPTION

[0013] Described herein are some embodiments of proxy-less Secured Sockets Layer (SSL) data inspection. In one embodiment, a TCP connection is established between a client (a.k.a. the initiator) and a HTTPS server (a.k.a. the responder). The client's web browser (or any network access application) issues a connection request, e.g., SSL Hello, to the server. A proxy-less SSL inspection appliance, such as a gateway device, intercepts the Hello request and sends an identical copy to the server. In response, the server sends a certificate to the proxy-less SSL inspection appliance. The proxy-less SSL inspection appliance strips out relevant information from the certificate (e.g., common name, etc.) and creates a new certificate signed by a certification-authority certificate, which the client has indicated to trust. The newly generated certificate is passed from the proxy-less SSL inspection appliance to the client. The client accepts the newly generated certificate because this certificate is signed by the certification-authority certificate. Packets received by the proxy-less SSL inspection appliance are decrypted and inspected by the proxy-less SSL inspection appliance using various mechanisms, such as deep packet inspection (DPI), content filtering, etc. After inspection, the proxy-less SSL inspection appliance re-encrypts the packets and forwards the packets to the client if there is no security issue with passing the packets. If potential malware or forbidden content is found in the packets, then the proxy-less SSL inspection appliance may block the packets from the client. The proxy-less SSL inspection appliance may further send a message to warn the client of its finding.

[0014] In the above scheme, TCP re-transmission logic is event driven based on retransmissions from server side and client side, rather than being scheduled by a TCP stack on each side of the TCP connection. In other words, the proxy-less SSL inspection appliance provides flow-control and retransmission of data packets without self-scheduling the packet retransmission using timeouts, but rather, based on the packet retransmission logic of either the client-side or server-side of the connection. As a result, security inspection of clear-text can take place at the proxy-less SSL inspection appliance without using a full TCP-based proxy.

[0015] In the following description, numerous details are set forth. It will be apparent, however, to one skilled in the art, that the present invention may be practiced without these specific details. In some instances, well-known structures and devices are shown in block diagram form, rather than in detail, in order to avoid obscuring the present invention.

[0016] Some portions of the detailed descriptions below are presented in terms of algorithms and symbolic representations of operations on data bits within a computer memory. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times,

principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

[0017] It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following discussion, it is appreciated that throughout the description, discussions utilizing terms such as “processing” or “computing” or “calculating” or “determining” or “displaying” or the like, refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

[0018] The present invention also relates to apparatus for performing the operations herein. This apparatus may be specially constructed for the required purposes, or it may comprise a general-purpose computer selectively activated or reconfigured by a computer program stored in the computer. Such a computer program may be stored in a computer-readable storage medium, such as, but is not limited to, any type of disk including floppy disks, optical disks, CD-ROMs, and magnetic-optical disks, read-only memories (ROMs), random access memories (RAMS), EPROMS, EEPROMs, flash memory, magnetic or optical cards, or any type of media suitable for storing electronic instructions, and each coupled to a computer system bus.

[0019] The algorithms and displays presented herein are not inherently related to any particular computer or other apparatus. Various general-purpose systems may be used with programs in accordance with the teachings herein, or it may prove convenient to construct more specialized apparatus to perform the required method steps. The required structure for a variety of these systems will appear from the description below. In addition, the present invention is not described with reference to any particular programming language. It will be appreciated that a variety of programming languages may be used to implement the teachings of the invention as described herein.

[0020] FIG. **2** illustrates one embodiment of a proxy-less system. The proxy-less system **200** includes a client **210**, a server **220**, and a gateway device **230** coupled between the client **210** and the server **220**. When the client **210** initiates a connection with the server **220**, the client **210** may be referred to as an initiator and the server **220** may be referred to as a responder, and vice versa. The client **210** and the server **220** may be implemented using various computing devices, such as personal computers, laptop computers, personal digital assistants (PDAs), cellular telephones, Smartphones, etc. The gateway device **230** may also be implemented using various computing devices, such as those listed above. In some embodiments, the gateway device **230** is implemented as a set-top box coupled to the client **210** locally. The gateway device **230** acts as a “middleman” device between the client **210** and the server **220**.

[0021] In some embodiments, the gateway device **230** may intercept a client connection request from the initiator, say the client **210**, before it reaches the intended endpoint, say the server **220**, and generate IP TCP packets as replies as if they were originated from that endpoint, and to do the same for communication with the original responder endpoint. Separate TCP state is kept for communication with the initiator and responder endpoints at the gateway device **230**. This state contains data allowing the gateway device **230** to do flow-control and retransmission. For example, the state may include a sequence number of the last packet received, which may be used in determining if the next packet is dropped or lost. In order to increase scalability and to simplify the gateway device **230**, TCP retransmission to a receiver may only be done when a retransmit from the sender is seen in some embodiments. Data from one side is not acknowledged until it is acknowledged by the opposite endpoint.

[0022] During connection setup, the TCP handshake is allowed to complete between the two hosts,

but once the client attempts to send data to negotiate a secured connection (e.g., SSL), the request is passed to an internal secured endpoint (such as the internal secured endpoint **231** or **235** in FIG. **2**) on the gateway device **230**. Before this endpoint continues negotiation, the gateway device **230** may first initiate a secured client connection to the responder endpoint, store the responder certificate details, and complete the key exchange.

[0023] Afterwards, secured connection certificate and/or key exchange and negotiation is completed with the initiator, optionally using a certificate dynamically generated with details from the responder certificate as discussed below. Because the gateway device **230** chooses the public keys and does the negotiation to terminate the SSL connection, it is possible for the gateway device **230** to inspect the clear text data sent by both sides. Once both connections are established, decrypted clear text data is transferred from one connection to the other as follows.

[0024] In some embodiments, the data received by the gateway device **230** from the initiator may be encrypted and sent over the responder secured connection, and vice versa. In this way, it is possible to view and/or modify the clear text data sent from one endpoint to the other. No configuration on either end (i.e., the client **210** and the server **220**) is necessary because the gateway device **230** which sits on the path between the two sides can detect when to attempt secured decrypting and/or re-encrypting by detecting a connection to a known SSL TCP port, or by detecting a presence of a valid SSL Hello packet to any port. As opposed to a conventional explicit third party SSL proxy, where the connecting client must be aware of the forwarding proxy relationship and contact the proxy SSL endpoint directly, both sides' TCP and SSL states appear to be communicating with their original endpoints, so this interception is transparent to both sides.

[0025] As discussed above, the gateway device **230** may dynamically generate a certificate in the process of establishing a secured connection between the client **210** and the server **220**. In some embodiments, the client **210** may use RSA encryption to verify a certificate delivered by the server **220** is “signed” by a third party authority that has previously been trusted by the client **210**. For instance, the client **210** may have previously accepted a certification-authority (CA) certificate from this third party. When the gateway device **230** intercepts the secured connection and responds using its own internal secured endpoint **235**, it is necessary to deliver a certificate containing a public key that the gateway device **230** has the private key for, so that key exchange is possible. The certificate also contains attributes to identify the endpoint to the client **210**. In general, the client **210** may verify these attributes before continuing to negotiate further. If the attributes do not all match what is expected, the client **210** may warn the user before continuing. In order to appear legitimate, the certificate details from the responder certificate from the server **220** are stored by the gateway device **230** and a new certificate is generated that appears substantially identical, except for the public key. The newly generated certificate is then signed by the CA certificate, which the client **210** has previously trusted. In this way, all checks done by the client **210** on the certificate may pass, and the client may complete the connection and begin sending data to the server **220** via the gateway device **230**.

[0026] FIG. **3** illustrates one embodiment of a method to establish a secured connection between a client and a responder without a proxy. The method may be performed by processing logic that may comprise hardware (e.g., circuitry, dedicated logic, programmable logic, processing cores, etc.), software (such as instructions run on a processing core), firmware, or a combination thereof.

[0027] Initially, processing logic detects a client's attempt to send data to negotiate a secured connection with a responder (processing block **310**). For example, the secured connection may be SSL. Then processing logic intercepts the client's request to responder (processing block **312**). Processing logic initiates a secured client connection to the responder (processing block **314**). In response, the responder may send a certificate to processing logic. Processing logic stores the responder's certificate details (processing block **316**). Then processing logic completes key exchange with the responder (processing block **318**). Finally, processing logic completes secured connection certificate and/or key exchange and negotiation with the client (processing block **319**).

To complete secured connection certificate and/or key exchange and negotiation with the client, processing logic may dynamically generate a new certificate to send to the client.

[0028] FIG. **4** illustrates one embodiment of a method to dynamically generate a certificate. The method may be performed by processing logic that may comprise hardware (e.g., circuitry, dedicated logic, programmable logic, processing cores, etc.), software (such as instructions run on a processing core), firmware, or a combination thereof.

[0029] Initially, processing logic receives a certificate from the responder at a gateway device (processing block **410**). Then processing logic stores details of the certificate, such as common name, on the gateway device (processing block **412**). Processing logic generates a new certificate substantially identical to the certificate from the responder at the gateway device (processing block **414**). Processing logic inserts a public key into the certificate at the gateway device, where the gateway device has the private key for the public key (processing block **416**). In some embodiments, the public key is pre-generated at the gateway device along with its private key pair. Finally, processing logic signs the new certificate with a certificate authority (usually a trusted third party) certificate, which the client has previously agreed to trust as a signing authority (processing block **418**). Note that the same public key may be inserted into all new certificates subsequently generated at the gateway device for the current connection.

[0030] FIG. **5** illustrates one embodiment of a method to perform proxy-less data inspection. The method may be performed by processing logic that may comprise hardware (e.g., circuitry, dedicated logic, programmable logic, processing cores, etc.), software (such as instructions run on a processing core), firmware, or a combination thereof.

[0031] Initially, processing logic uses a gateway device (such as the gateway device **230** shown in FIG. **2**) to set up a secured connection according to a secure network protocol (e.g., SSL) between a client and a responder (processing block **510**). Details of some embodiments of the secured connection setup have been discussed in details above. Then processing logic uses the gateway device to transparently intercept data transmitted between the client and the responder (processing block **512**). Processing logic further uses the gateway device to provide flow control and retransmission of data packets transmitted between the client and the responder (processing block **514**). The flow control and retransmission of data may be provided without self-scheduling the packet retransmission using timeouts at the gateway device, but rather, based on the packet retransmission logic of either the client-side or the responder-side of the connection. Using the gateway device, processing logic performs security screening on data packets transmitted between the client and the responder (processing block **516**). The security screening may include content filtering, deep packet inspection, etc.

[0032] FIG. **6** illustrates a diagrammatic representation of a machine in the exemplary form of a computer system **600** within which a set of instructions, for causing the machine to perform any one or more of the methodologies discussed herein, may be executed. In alternative embodiments, the machine may be connected (e.g., networked) to other machines in a LAN, an intranet, an extranet, and/or the Internet. The machine may operate in the capacity of a server or a client machine in client-server network environment, or as a peer machine in a peer-to-peer (or distributed) network environment. The machine may be a personal computer (PC), a tablet PC, a set-top box (STB), a Personal Digital Assistant (PDA), a cellular telephone, a web appliance, a server, a network router, a switch or bridge, or any machine capable of executing a set of instructions (sequential or otherwise) that specify actions to be taken by that machine. Further, while only a single machine is illustrated, the term “machine” shall also be taken to include any collection of machines that individually or jointly execute a set (or multiple sets) of instructions to perform any one or more of the methodologies discussed herein.

[0033] The exemplary computer system **600** includes a processing device **602**, a main memory **604** (e.g., read-only memory (ROM), flash memory, dynamic random access memory (DRAM) such as synchronous DRAM (SDRAM) or Rambus DRAM (RDRAM), etc.), a static memory **606** (e.g.,

flash memory, static random access memory (SRAM), etc.), and a data storage device **618**, which communicate with each other via a bus **632**.
[0034] Processing device **602** represents one or more general-purpose processing devices such as a microprocessor, a central processing unit, or the like. More particularly, the processing device may be complex instruction set computing (CISC) microprocessor, reduced instruction set computing (RISC) microprocessor, very long instruction word (VLIW) microprocessor, or processor implementing other instruction sets, or processors implementing a combination of instruction sets. Processing device **602** may also be one or more special-purpose processing devices such as an application specific integrated circuit (ASIC), a field programmable gate array (FPGA), a digital signal processor (DSP), network processor, or the like. The processing device **602** is configured to execute the processing logic **626** for performing the operations and steps discussed herein.
[0035] The computer system **600** may further include a network interface device **608**. The computer system **600** also may include a video display unit **610** (e.g., a liquid crystal display (LCD) or a cathode ray tube (CRT)), an alphanumeric input device **612** (e.g., a keyboard), a cursor control device **614** (e.g., a mouse), and a signal generation device **616** (e.g., a speaker).
[0036] The data storage device **518** may include a machine-accessible storage medium **630** (also known as a machine-readable storage medium or a computer-readable medium) on which is stored one or more sets of instructions (e.g., software **622**) embodying any one or more of the methodologies or functions described herein. The software **622** may also reside, completely or at least partially, within the main memory **604** and/or within the processing device **602** during execution thereof by the computer system **600**, the main memory **604** and the processing device **602** also constituting machine-accessible storage media. The software **622** may further be transmitted or received over a network **620** via the network interface device **608**.
[0037] While the machine-accessible storage medium **630** is shown in an exemplary embodiment to be a single medium, the term “machine-accessible storage medium” should be taken to include a single medium or multiple media (e.g., a centralized or distributed database, and/or associated caches and servers) that store the one or more sets of instructions. The term “machine-accessible storage medium” shall also be taken to include any medium that is capable of storing, encoding or carrying a set of instructions for execution by the machine and that cause the machine to perform any one or more of the methodologies of the present invention. The term “machine-accessible storage medium” shall accordingly be taken to include, but not be limited to, solid-state memories, optical and magnetic media, etc. In some embodiments, machine-accessible storage medium may also be referred to as computer-readable storage medium.
[0038] Thus, some embodiments of cloud-based gateway anti-virus scanning have been described. It is to be understood that the above description is intended to be illustrative, and not restrictive. Many other embodiments will be apparent to those of skill in the art upon reading and understanding the above description. The scope of the invention should, therefore, be determined with reference to the appended claims, along with the full scope of equivalents to which such claims are entitled.

Claims

1. A method for performing proxy-less data inspection, the method comprising: intercepting at a gateway device, a connection request from a client for a secured connection between the client and a responder device; generating by the gateway device an identical copy of the connection request to send to the responder device; receiving from the responder device a first certificate including a first public key, wherein the first certificate uses the first public key to complete a first public key exchange between the gateway device and the responder device; creating at the gateway device a second certificate that uses a second public key to complete a second public key exchange between the gateway device and the client; decrypting data packets transmitted between the client and the

responder device and intercepted by the gateway device, wherein the gateway device executes flow-control of one or more data packets by managing at the gateway device a first Transmission Control Protocol (TCP) state for the client and a second TCP state for the responder device, wherein the first TCP state includes a sequence number of a last data packet from the client, and wherein the second TCP state includes a sequence number of a last data packet from the responder device; and performing security screening on the decrypted data packets.

2. The method of claim 1, wherein the second public key of the second certificate is different from the first public key of the first certificate.

3. The method of claim 1, further comprising storing certificate details of the first certificate at the gateway device after receiving the first certificate from the responder device.

4. The method of claim 1, wherein the second certificate is identical to the first certificate except for the second public key.

5. The method of claim 1, wherein the security screening includes at least one of content filtering or deep packet inspection (DPI).

6. The method of claim 1, wherein decrypting the data packets is based on a private key stored at the gateway device.

7. The method of claim 1, further comprising re-encrypting the data packets before the data packets are forwarded and after determining that the data packets do not include potential malware or forbidden content.

8. The method of claim 1, further comprising: blocking the data packets from the client after identifying that the data packets include at least one of a potential malware or forbidden content; and sending a warning message to the client from the gateway device.

9. The method of claim 1, wherein the flow-control of the data packets includes use of the sequence number of the last data packet from the client to determine a status of transmission of a subsequent data packet from the client.

10. The method of claim 1, wherein the flow-control of the data packets includes use of the sequence number of the last data packet from the responder device to determine a status of transmission of a subsequent data packet from the responder device.

11. A non-transitory computer-readable storage medium having embodied thereon a program executable by a processor for implementing a method for performing proxy-less data inspection, the method comprising: intercepting at a gateway device, a connection request from a client for a secured connection between the client and a responder device; generating by the gateway device an identical copy of the connection request to send to the responder device; receiving from the responder device a first certificate including a first public key, wherein the first certificate uses the first public key to complete a first public key exchange between the gateway device and the responder device; creating at the gateway device a second certificate that uses a second public key to complete a second public key exchange between the gateway device and the client; decrypting data packets transmitted between the client and the responder device and intercepted by the gateway device, wherein the gateway device executes flow-control of one or more data packets by managing at the gateway device a first Transmission Control Protocol (TCP) state for the client and a second TCP state for the responder device, wherein the first TCP state includes a sequence number of a last data packet from the client, and wherein the second TCP state includes a sequence number of a last data packet from the responder device; and performing security screening on the decrypted data packets.

12. The non-transitory computer-readable storage medium of claim 11, wherein the second public key of the second certificate is different from the first public key of the first certificate.

13. The non-transitory computer-readable storage medium of claim 11, further comprising instructions executable to store certificate details of the first certificate at the gateway device after receiving the first certificate from the responder device.

14. The non-transitory computer-readable storage medium of claim 11, wherein the second

certificate is identical to the first certificate except for the second public key.

15. The non-transitory computer-readable storage medium of claim 11, wherein the security screening includes at least one of content filtering or deep packet inspection (DPI).

16. The non-transitory computer-readable storage medium of claim 11, wherein decrypting the data packets is based on a private key stored at the gateway device.

17. The non-transitory computer-readable storage medium of claim 11, further comprising instructions executable to re-encrypt the data packets before the data packets are forwarded and after determining that the data packets do not include potential malware or forbidden content.

18. The non-transitory computer-readable storage medium of claim 11, further comprising instructions executable to: block the data packets from the client after identifying that the data packets include at least one of a potential malware or forbidden content; and send a warning message to the client from the gateway device.

19. The non-transitory computer-readable storage medium of claim 11, wherein the flow-control of the data packets includes use of the sequence number of the last data packet from the client to determine a status of transmission of a subsequent data packet from the client.

20. A gateway apparatus for performing proxy-less data inspection, the gateway apparatus comprising: a memory; a communication interface that communicates over a communication network wherein the communication interface: intercepts a connection request from a client for a secured connection between the client and a responder device, sends an identical copy of the connection request to the responder device, receives from the responder device a first certificate including a first public key, wherein the first certificate uses the first public key to complete a first public key exchange between the gateway apparatus and the responder device, and intercepts data packets transmitted between the client and the responder device; and a processor that executes instructions stored in the memory, wherein execution of the instructions by the processor: creates a second certificate, wherein the second certificate uses a second public key to complete a second public key exchange between the gateway apparatus and the client, decrypts the data packets transmitted between the client and the responder device, wherein flow-control is executed on one or more of the data packets by managing a first Transmission Control Protocol (TCP) state for the client and a second TCP state for the responder device, wherein the first TCP state includes a sequence number of a last data packet from the client, and wherein the second TCP state includes a sequence number of a last data packet from the responder device, and performs security screening on the decrypted data packets.