A Policy Based Approach to Securing Egress Secure Socket Layer Connections on Local Area Networks

Joseph Mathews, James Rowell, David Nadwodny
US Naval Research Lab, ITT Industries
Center for High Assurance Computing Systems
4555 Overlook Ave SW
Washington, DC 20375
USA

mathews@itd.nrl.navy.mil / rowell@itd.nrl.navy.mil

ABSTRACT

Common network environments allow users a wide variety of protocols and applications to accomplish their job functions as well as day-to-day communications. One such example is the Secure Sockets Layer (SSL) protocol. SSL provides client and server authentication, data confidentiality and data integrity. SSL has been successfully employed in conjunction with a number of legacy protocols in order to ensure additional security. While many of these services are a requirement to complete basic mission-critical tasks, they can be manipulated in order to produce network activities that would normally be prohibited. SSL can be used to tunnel other applications or protocols and can therefore hide traffic and activity that would normally never be allowed out of a network. Traffic utilizing SSL is encrypted and cannot be screened by traditional methods of network defence for unauthorized activities. There is an increasing need to monitor and regulate all traffic in networked environments. Due to the confidentiality provided, SSL traffic offers a unique challenge to these requirements. We explore a policy-based interception solution that allows additional controls to be placed on egress SSL traffic. This solution will provide the ability to detect and prevent SSL misuse.

1.0 INTRODUCTION

In the early 1990s the Internet was growing while preparing to handle commercial applications. There was a clear need to provide secure Hyper Text Transfer Protocol (HTTP) connections. To fill this void, Netscape™ developed the Secure Sockets Layer (SSL) protocol. SSL resides between the transport layer and the higher layers of the Open Systems Interconnect (OSI) protocol stack. It allows for server authentication to a client and vice versa while providing an end-to-end secure channel between two nodes. While SSL was originally designed for HTTP, due to its non-application centric design, it has been used to secure many other cleartext legacy protocols and services. The wide spread acceptance of SSL has resulted in an abundance of egress encrypted traffic on Local Area Networks (LAN).

The SSL protocol is composed of two layers. The SSL Handshake Protocol enables the client and server to authenticate one another and negotiate encryption algorithms and cryptographic keys prior to transmission of application layer data. The SSL Record Protocol resides on top of the Transmission Control Protocol (TCP) layer and cryptographically encapsulates data from higher level application layer protocols. The encapsulated data is then transmitted over a network as the payload of an SSL packet.

Paper presented at the RTO IST Symposium on "Adaptive Defence in Unclassified Networks", held in Toulouse, France, 19 - 20 April 2004, and published in RTO-MP-IST-041.

Since virtually any network application can be tunnelled over SSL, an opportunity is created for a user to send traffic out of a network that would normally not be permitted. This could include connections to unauthorized web sites, transference of sensitive data, and connections to Internet Relay Chat (IRC) servers. To make matters worse, common network security tools like application layer proxies and Network Intrusion Detection Systems (NIDS) lack the ability to effectively prevent or even detect this sort of activity. In order to mitigate this threat, the traffic must be put in its cleartext form allowing for a determination to be made of whether or not it is acceptable for transmission to other networks. This can be accomplished using what is traditionally known as a Man-In-the-Middle (MITM) technique and applying a rules engine to SSL traffic for analysis.

2.0 SSL TUNNELLING

SSL is compatible with any network protocol that runs over the TCP layer of the OSI stack. Therefore, it is possible to set up an SSL tunnel and transmit a wide array of data over that tunnel. This enables any user capable of sending outbound SSL traffic to send any type of data out of the network.

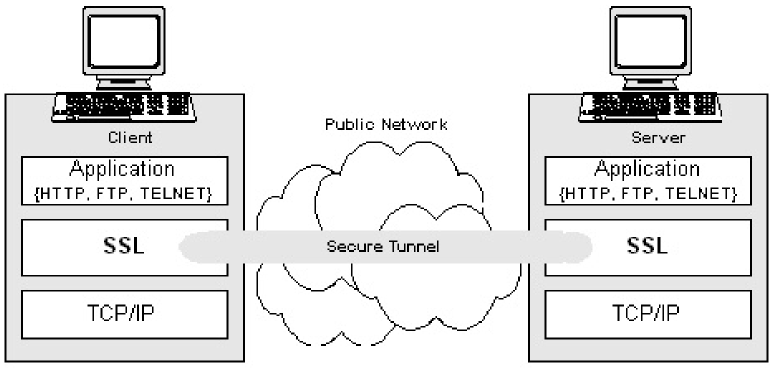

Figure 1: SSL Operation

As depicted in Figure 1, once an SSL tunnel is established between two hosts, any sort of application layer protocol may be encrypted and transmitted over an insecure medium such as a public network. Because the data is encrypted, it is not possible to determine what sort of data is being transmitted.

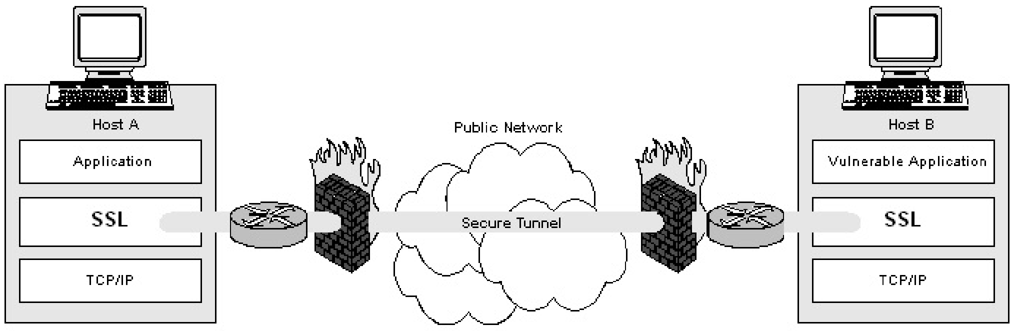

Figure 2: SSL Tunnelling

As depicted in Figure 2, Host A has the ability to exploit vulnerabilities in Host B (and vice versa) without network defences at either end of the path detecting the malicious activity due to the secure tunnel in which the packets travel. At this point either host could be attacked or even compromised without being detected. As shown, SSL has the potential to severely degrade the integrity of network perimeter defences that are deployed. It is also important to note that SSL tunnels can be established to numerous ports on multiple hosts.

3.0 TRADITIONAL MEASURES OF PREVENTION

SSL traffic is largely unmanageable using the suite of available network security tools and devices. At the most exhaustive level, payload analysis of every packet on the wire yields gibberish when dealing with encrypted traffic. Smarter sensors are limited to monitoring the state of the connection itself while implicitly trusting the encrypted payloads once an SSL connection has been successfully negotiated. To illustrate, we examine two of the most popular methods of network defences: firewalls and intrusion detection systems.

3.1 Firewalls

Standard packet filters and stateful inspection firewalls have no feature suitable to defend against SSL misuse. Application layer proxies, often integrated into firewalls, have the ability to compare traffic against the protocol specification for a particular application, but this is ineffective for SSL because traffic is encrypted after the initial handshake is made, preventing the firewall from seeing what application data is actually going out over a port. They are limited to monitoring the details of the connection itself, such as the encryption algorithm utilized, rather than the payload.

3.2 Network Intrusion Detection Systems

Network Intrusion Detection Systems will listen to traffic traversing a network in an effort to identify malicious traffic. These are effective tools for monitoring malicious activity such as network reconnaissance scanning and denial-of-service (DOS) attacks. Sensors may employ multiple methodologies to detect attacks such as signature analysis, protocol decoding, or anomaly detection. Once an attack is detected, the engine will provide a variety of options to notify, alert, or log with respect to the event at hand.

NIDS are equally ineffective at detecting SSL misuse. Despite the multiple detection methods they may utilize, they lack the ability to analyze encrypted payloads for attack signatures or traffic anomalies rendering them useless for SSL management. Other methods of intrusion detection exist, such as monitoring the state of connections between hosts, but are not as effective.

4.0 POLICY BASED INTERCEPTION

By creating a proxy point in a network for all outbound SSL connections and intercepting the SSL handshake, it is possible to decrypt SSL traffic, analyze it, and determine whether it is permitted to leave the network enclave. There are two tasks for such a system: interception of the SSL traffic, and a policy based analysis of the traffic in its cleartext form.

4.1 SSL Interception

The interception method for SSL is relatively well known. A form of the Man-in-the-Middle (MITM) technique, the method involves intercepting the outgoing SSL handshake from a client to server, forging the server's reply back to the client, and then forwarding the traffic along to the actual destination. This enables the SSL traffic leaving the network to be seen in cleartext while passing through a proxy, prior to being forwarded to its destination. Figure 3 depicts an SSL connection leaving the client and getting through the proxy as the steps listed below illustrate.

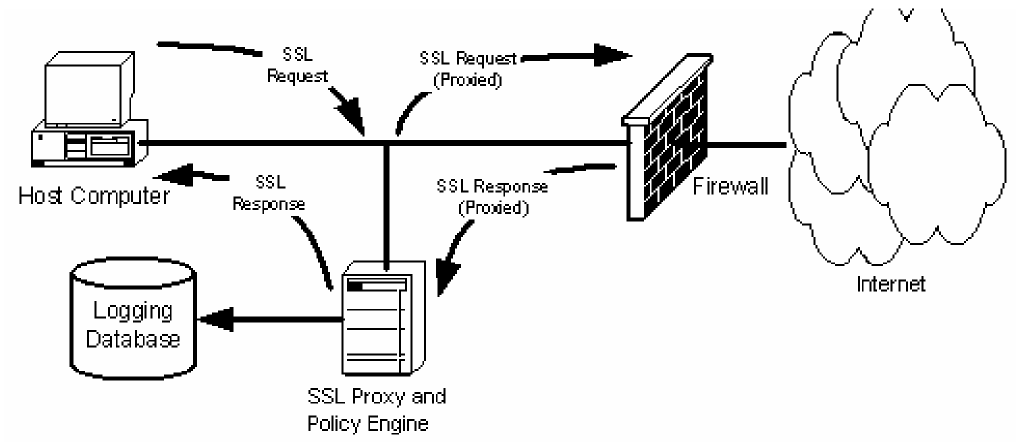

Figure 3: SSL Interception

- The client will make a request for an SSL connection with an SSL handshake.
- Traffic is routed to the SSL proxy by dynamically routing traffic based on its destination port.
- The proxy masquerades as the end point and completes the SSL handshake with the server.
- The proxy masquerades the server's certificate and presents it to the client. Depending on the application service requested the client would receive a warning concerning the server's certificate. This can be eliminated by adding the proxy's certificate to the client's trusted Certificate Authority (CA) list.
- Once the masqueraded certificate is accepted, the proxy completes the SSL handshake with the client's original destination.
- After the proxy and the client's destination complete the handshake process, the proxy is receiving encrypted traffic from the client, which is then passed through the proxy in cleartext and encrypted again before being forwarded to the client's actual destination.

4.2 Policy Engine

Figure 4 illustrates the operation of the SSL proxy. Incoming packets are decrypted and examined in cleartext form by the policy engine using the rules list as a reference for legitimate sites and activities. Legitimate SSL activity is encrypted again and forwarded to the original location while non-compliant data is prevented from leaving the network and an alert is written to an event database. Security administrators may view this database through a console in order to better understand the types of malicious activity and invalid uses of SSL occurring on their network.

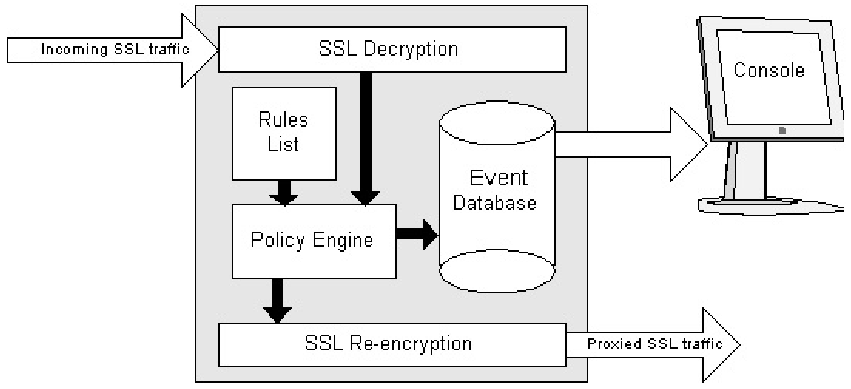

Figure 4: Policy Engine

The policy engine's rules list for a protocol should contain logic similar to that of an application proxy, checking for the correct application layer protocol running over SSL. It may also be tailored to accommodate the particular environment in which the system is employed. For example, the rules list should include a list of acceptable end points for SSL connections which can be left open (outgoing SSL connections are allowed to any host), or in an extreme case statically set to include only a few outbound destinations. Certain hosts on the network may want to be implicitly trusted and not monitored at all. All outgoing SSL traffic may be archived. Or perhaps only traffic from particular hosts or to particular destinations may be archived. It may be desirable in certain circumstances to only log the connection instance. A powerful application of this policy based engine would be to archive all egress SSL traffic while forwarding the traffic in its cleartext form to an IDS sensor which could then apply its own method of analysis. This will prevent unauthorized application tunnelling and aid the mitigation of attempted exploits.

5.0 CONCLUSION

The confidentiality provided by SSL is essential for many network connections, but the nature of encrypted traffic and the inability to effectively control and monitor egress SSL traffic present serious network security risks. It is possible to mitigate many of the risks associated with allowing egress SSL traffic by leveraging a traffic interception technique while applying the operation of a policy engine.

6.0 REFERENCES

[1] McClure, Stuart. Hacking Exposed: Network Security Secrets and Solutions. McGraw-Hill Osborne Media, Feb 2003.

[2] Northcutt, Stephen. Inside Network Perimeter Security. New Riders, Jul 2003.

[3] Transport Layer Security Working Group. The SSL Protocol: Version 3.0. INTERNET-DRAFT, Netscape Communications. Nov 1996. Available: http://wp.netscape.com/eng/ssl3/draft302.txt