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

Joseph Mathews
mathews@alum.mit.edu

David Nafaday
IT&T Industries
Center for High Assurance Computing Systems
NRL Research Laboratories, Code 5454
Washington, DC 20375

Juan Rowell
Rowell@mit.edu

Abstract

Contemporary 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 assure additional security.

While many of these services are required 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 transport SSL-encrypted traffic and activity that would normally not be allowed out of a network. Traffic utilizing SSL is encrypted and cannot be screened by traditional methods of networkdefense for unauthorized activity.

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 encrypted SSL traffic. This solution will provide the ability to detect and prevent SSL misuse.

1. INTRODUCTION

In the early 1990s the Internet was growing while providing a great many applications. At that time, there was a clear need to provide secure HTTP Test 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 Interconnection (OSI) 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 new-application centric design, it has been used to secure many other diverse legacy protocols and services. The wide spread acceptance of SSL has resulted in an abundance of encrypted and secure traffic on Local Area Networks (LANs).

The SSL protocol is comprised 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 encrypts/demultiplexes data from higher level application layer protocols. The encapsulated data is then transmitted over a network as the payload of an SSL packet. Since virtually any network application can be secured over SSL, an opportunity is created for a user to send traffic out of a network that would normally be permitted. This would include connections to untrusted web sites, transfer of sensitive data, and connections to Internet Relay Chat (IRC) servers. To make matters worse, common network security tools such as 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 clearest 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 Man-In-The-Middle (MITM) technique and applying a proxy engine to SSL traffic for analysis.

2. SSL TUNNELING

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 1.2 – SSL Backdoor

As depicted in Figure 2.2, Host A has the ability to exploit vulnerabilities in Host B (and vice versa) without network defenses at either end of the path detecting malicious activity due to the insecure 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 defenses that are deployed. It is also important to note that SSL tunnels can be established to numerous ports on multiple hosts.
3. TRADITIONAL MEASURES OF PREVENTION

SSL traffic is largely unmanageable using the tools of available network security tools and devices. At the most
coarse grain level, payload analysis of every packet on the wire yields gibberish when dealing with encrypted
traffic. Snort 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 defense: firewalls and intrusion detection systems.

3.1 FIREWALLS

Standard packet filters and stateful inspection firewalls have no feature suitable to defend against SSL
reuses. Application layer proxies, often integrated into firewalls, have the ability to examine 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 NIDS

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 impact to the event at hand.

NIDS are equally ineffective at detecting SSL reuses. 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 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 domain.
There are two tasks for such a system: intercepting the SSL traffic, and a policy based analysis of the traffic in its
cleared form.

4.1 FIREWALLS

The interception method for SSL is relatively well known. A form of the Node-in-the-Middle (MitM) technique, the
method intercepts the ongoing SSL handshakes from a client to server, forcing 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 engine or being forwarded to the destination.
Figure 4.1 depicts an SSL connection leaving the client and getting through the proxy as the steps listed below illustrate.

The policy engine’s rule 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 deployed. For example, the rule 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 whitelisted destinations. Certain hosts on the network may want to be
implicitly trusted and not monitored at all. All outgoing SSL traffic may be allowed. Or perhaps only traffic from
particular hosts or to particular destinations may be allowed. It may be desirable to have SSL connections in only
to log the connection instance. A powerful application of this policy based engine would be to archive all
transmissions between a host belong to an ISP server which could then apply its own method of analysis. This
will prevent unauthorized application tunneling and aid the mitigation of attempted exploits.

5 CONCLUSION

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

6 REFERENCES


Figure 4.1 – SSL Interception

- The client will make a request for an SSL connection on an SSL handshake.
- Traffic is routed to the SSL proxy by dynamically routing traffic based on its destination port.
- The proxy examines the end point and completes the SSL handshake with the server.
- The proxy examines the server’s certificate and presents it to the client. Depending on the application service
  requested the client will 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 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.

Figure 4.2 – Policy Engine

Figure 4.2 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 reduced costs of SSL occurring
on their network.