Secure Web Access in an Environment of Mutual Distrust

Jeff Morgan, Howard Morris, Venky Krishnan
Anca-Andreea Ivan[1]
Internet and Mobile Systems Laboratory
HP Laboratories Palo Alto

HPL-2001-60
March 22nd , 2001*

E-mail: morgan@hpl.hp.com, venky@hpl.hp.com, ivan@cs.nyu.edu

Today, the Internet is used for a variety of services that require a high degree of security (ex: e-commerce, banking and stock exchange transactions, and credit card account management). More than ever, companies rely on the web for conducting their businesses: telecommuters working from home use computers to remotely access resources, partner companies doing collaborative projects use networks to interact, retail companies sell products over the web, etc. Wireless networks further enhance user connectivity in an increasingly nomadic environment. Users demand transparent access to secure resources in their "home" systems from insecure remote network infrastructures like Internet Cafe, airports, shopping centers, other companies. Further, network providers want to prohibit these users from accessing resources other than the ones for which they have access rights. This paper focuses on a solution that provides secure remote access to Web resources, when the connection traverses several trusted and nontrusted, wireless or wired based networks.

* Internal Accession Date Only Approved for External Publication

[1] Courant Institute of Mathematical Sciences, New York University, New York, NY 10003

Contents

1 INTRODUCTION **3**

2 TYPES OF SECURITY ISSUES **4**

3 SWT (SECURE WEB TUNNELING) **5**

3.1 SWT CONNECTION FLOW 6
3.2 MULTIPLE TARGET DOMAINS 7
3.3 CONTROLLED ACCESS RIGHTS TO USERS 8
3.4 WIRELESS INTEGRATION 9
3.4.1 Network & SWT Auto-Configuration 9
3.5 PRIOR SYSTEMS 10
3.6 TEST-BED 10
3.6.1 Scenario 1 11
3.6.2 Scenario 2 11

4 CONCLUSIONS **11**

5 APPENDIX A **12**

5.1 INTERNET PROTOCOL SECURITY (IPSEC) 12
5.2 SECURE SOCKETS LAYER / TRANSPORT LAYER SECURITY (SSL/TLS) 13
5.3 SECURE SHELL PROTOCOL (SSH) 13
5.4 SECURITY PROTOCOLS ANALYSIS 15

6 APPENDIX B - THE WIRELESS NETWORKS **17**

6.1 RADIO FREQUENCY TECHNOLOGIES 17
6.1.1 Security in 802.11b 17
6.1.2 Wireless Configuration Issues 17
6.1.3 802.11b Security Solutions' Problem Domain 18
6.2 INFRARED TECHNOLOGIES 18

7 APPENDIX C **19**

8 APPENDIX D **20**

9 APPENDIX E - BIBLIOGRAPHY **21**

Secure Web Access in an Environment of Mutual Distrust

Abstract

Today, the Internet is used for a variety of services that require a high degree of security (ex: e-commerce, banking and stock exchange transactions, and credit card account management). More than ever, companies rely on the web for conducting their businesses: telecommuters working from home use computers to remotely access resources, partner companies doing collaborative projects use networks to interact, retail companies sell products over the web, etc. Wireless networks further enhance user connectivity in an increasingly nomadic environment. Users demand transparent access to secure resources in their "home" systems from insecure remote network infrastructures like Internet Café, airports, shopping centers, other companies. Further, network providers want to prohibit these users from accessing resources other than the ones for which they have access rights. This paper focuses on a solution that provides secure remote access to Web resources, when the connection traverses several trusted and non-trusted, wireless or wired based networks.

1 Introduction

The widespread use of the Internet in our every day lives significantly increases the need for secure communication. From this perspective, there are two very distinctive trends: one part of the Internet world, formed by important organizations and communities is focusing its work on designing security protocols; on the other hand, the second community, formed by hackers and web pirates, spends a huge amount of effort to steal secrets and violate users' privacy.

In the second section of this paper, "**Types of Security Issues**", a few of the very well known and vicious attacks against users and servers are discussed: packet sniffing, identity and packet spoofing, and denial of service attacks.

The 3rd section, "**SWT (Secure Web Tunneling)**", gives a detailed presentation of the HP – SWT architecture. The SWT provides the underlying mechanism to facilitate secure Web access in an environment of mutual distrust. The section explains the SWT protocol and the configuration settings of the two components of this system – the SWT Client Proxy and the SWT Server Proxy. This Section also describes a configuration process that sets up network connectivity for the nomadic user. Next, it provides an **Auto Configuration** solution for an *802.11b*-based environment that simplifies the usage.

The SWT architecture and protocol were tested on a simulation of an enterprise environment. The tests are described at the end of the SWT Section, "**Test-bed**". The test-bed consisted of three private networks, both wireless and wired-based, representing two companies separated by firewalls, and the Internet.

The last section, "**Conclusions**", highlights open issues and the proposed next steps.

The Appendix includes a discussion of the different protocols for security and an analysis of the choice of SWT, a combination of SSL & proxy for the nomadic environment.

Appendix A, "**Security Protocols**" describes some of the protocols used today against security threats. There is no single protocol solution that could successfully protect against all the different attacks. Typically, Virtual Private Networks (VPN) refers to a security solution that uses a combination of protocols to provide a secure environment in a non-nomadic environment. VPN includes IPSEC, SSL/TLS and SSH. These protocols are analyzed in the context of this paper's problem space. It also identifies SSL in combination with Web Proxies as the building blocks of the Secure Web Tunnel. Appendixes C and D include two simple tables that highlight the similarities and the differences between IPSEC, SSL/TLS and SSH.

Appendix B, "**The Wireless Networks**", defines wireless networking and enumerates the different technologies used to build such networks. The different technologies have varied characteristics & this section identifies the areas where these technologies can be best applied. Wireless technologies, due to its nature, stipulate a need for additional configuration & encryption measures to ensure secure access. This section discusses these issues & their impact on nomadic connectivity.

2 Types of Security Issues

The more complex operating systems and protocols become, the easier it is to find a safety breach that can be exploited for malevolent purposes. No matter how many security bugs are fixed and how much the systems are improved, hackers keep finding new open doors for breaking in. This section describes the various types of security issues:

Packet Sniffing: Even though it can be considered the least invasive attack, packet sniffing is probably the most feared by common Internet users (people who connect from home and use WWW for e-commerce, banking or stock exchange operations). Users tend to be reluctant to send passwords, credit card numbers or social security numbers over the Web, considering that the best way to protect their secrets from being stolen by hackers is not to make them available. Unfortunately, this drastic solution impedes the users from fully take advantage of the WWW and causes important significant losses to Internet-based companies.

Packet & Identity Spoofing: The next level in security attacks is packet and identity spoofing. This attack is much more serious than the previous one, because not only does it get the information sent but it also modifies it. The effects can be disastrous in a financial company that buys/sells stock over the Web. It is enough to change a simple packet in a stock exchange transaction, from "buy" to "sell", and the meaning of the entire operation will be completely different with possible irreparable results. Identity spoofing is a serious threat in today's world, where most authentication and authorization systems are based on <user login, password> pairs. Stealing the identity of another person is the best way for a hacker to cover his tracks and/or to enter a system without trying to break into it.

Denial of Service Attacks: Unlike the previously described attacks, this one is targeted on service providers (mainly) and the two most common effects are crashing or shutting down servers. The goal of denial-of-service attacks is to prevent legitimate users from using a certain service and there are many ways to achieve it: consumption of resources (network bandwidth, memory, disk space, CPU time, etc.), destruction or alteration of network configuration information, and physical destruction or alteration of network components [CERT99]. Here are a few examples of denial-of-service attacks:

- The attacker exploits flaws in the TCP/IP protocol by requiring for TCP connections to be opened without completing the process. This "SYN flood" attack consumes the kernel data structures necessary to create and manage new network connections. Instead of establishing real connections, the machine will be always busy trying to complete the bogus, halfway requests.
- One way to flood the network and consume the entire bandwidth is to generate an enormous number of packets (ICMP ECHO or any other packets) directed to a particular server. Further more, the attacker can orchestrate the attack by simultaneous use of different machines.
- Beside network bandwidth, there are many other resources that can crash by exhaustion. For example, to consume disk space, attackers can send a large number of e-mails, cause errors that need to be logged, or perform any other operation that results in writing on the disk. Some systems protect themselves by limiting the amount of data that can be written, but some do not. Other resources that might need special protection are printers, tape devices, and monitors.

3 SWT (Secure Web Tunneling)

The **goal of this paper** is to provide a *solution to the complex problem of a nomadic user's secure access to Web resources*. The underlying assumption in this paper is that the working environment is a Web-centric world. CoolTown [Cool00] is a vision where people, places and things are first-class citizens of the Web and Web-based appliances and E-services give you what you need, when and where you need it for work, play and managing your life.

From the Security perspective, this Web-centric world is viewed as:

a) Divided into Trusted zones separated by firewalls/proxies (companies) and the rest of the Internet considered as an insecure domain,
b) Populated by users - people with Personal Access Devices (ex: laptops, PDAs) traveling between different zones (both Trusted & Insecure) and
c) Resources - web servers located inside trusted zones.

The network contains both wireless and wired-based systems. The users should be able to transparently access web resources from either inside or outside their companies, using a secure, encrypted channel capable of traversing firewalls and distrusted zones. The simplest scenario is: a user opens a browser and tries to access a web server. He can start several connections with a web server, not all of them needing to be secure.

To make things clearer, here is a scenario: John is an employee of Company A. His tools there are his laptop and a web server (has his documents). His company has implemented a firewall to protect their resources. Every time he travels outside his company, he needs to access the web server using his laptop - changing his location should not influence his work. He should be able to access his resources from the open subnet & from within other firewalls like from within another Company B. Not only should John be provided secure access to his resources but also Company B's resources should be protected from John.

Secure Web Tunneling (SWT) is a tunneling solution that provides secure web access in mutually distrustful environments like the one depicted above. SWT's basic building blocks are SSL & the http proxy server. The SWT's architecture operates on the premise that the 2-ends of the connection are trusted. The user accesses his resources from a trusted system – his PAD (for ex: his laptop). The target resources are in his "home" trusted environment. In SWT, an SSL connection is established between these two end-points. The http request from the user to the web server is tunneled through this SSL connection.
Since the user is nomadic, the SSL connection from the PAD needs to get routed to the resource. Further, this route needs to be configured based on his current location's network configuration. This is achieved by configuring the appropriate proxy server settings.
The SSL tunneling & proxy connections are enhancements to an http proxy server. There is an SWT proxy resident on the PAD and another on the resource-end of the system. The http client (for ex: Web Browser) is configured with its settings pointing to the SWT proxy (referred to as the SWT client) running on the PAD. The SWT client establishes an SSL tunnel to the SWT proxy resident on the resource end (referred to as the SWT server). The http client does not require any modifications. As far the http client is concerned, it is merely interacting with a proxy (that happens to be resident on the same machine). Similarly, on the resource end, the target web server does not require any modifications. All its interactions are with a proxy server - the SWT server. The SWT client & server maintain the Digital Certificates to establish an SSL connection.

3.1 SWT Connection Flow

The SWT provides an end-to-end secure environment. This requires the 2 ends to be installed in systems that are trusted. The SWT client is placed in the user's PAD. The SWT server is placed straddling the resource's domain firewall. Both the SWT client & server need to be configured with the respective Digital Certificates necessary to enable SSL. The user needs to install & configure his PAD with the SWT client proxy. This proxy needs to route the http request/response to/from the target resources. To enable this, the SWT proxy has to know:

1. Current Location's Proxy Setting - The proxy setting in his current location (assuming this is within a firewall) that will enable access to systems outside of this domain.
2. SWT Server Proxy Address - The SWT client needs to route the http request to the SWT Server proxy. This is the upstream proxy that the Current Location's Proxy Server will communicate with for access to the resource.

Figure 3.1 shows the various components of the SWT solution.

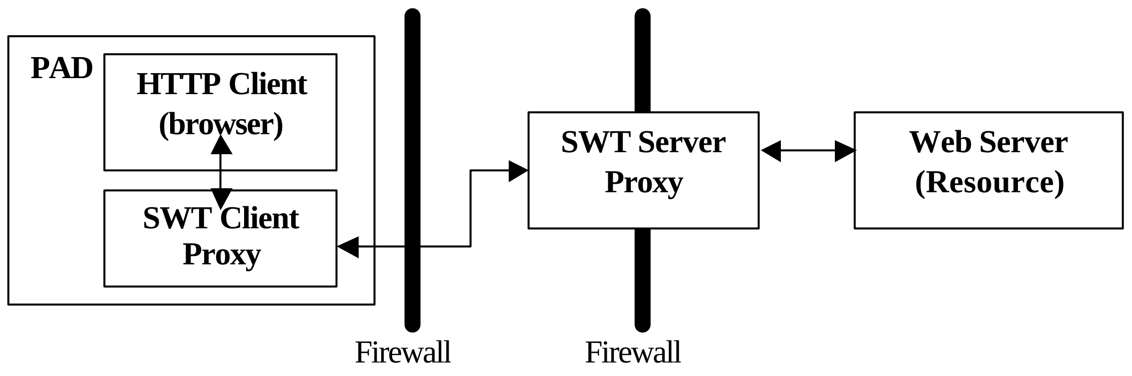


Figure 3-1 SWT Architecture

3.2 Multiple Target Domains

The previous sub-section describes the flow of the http interaction between a PAD and a target. This is generalized in the SWT Client to handle multiple types of connections between the PAD and different targets. A User may want (& may have access rights) to access http resources from different domains. Figure 3.2 illustrates this case.

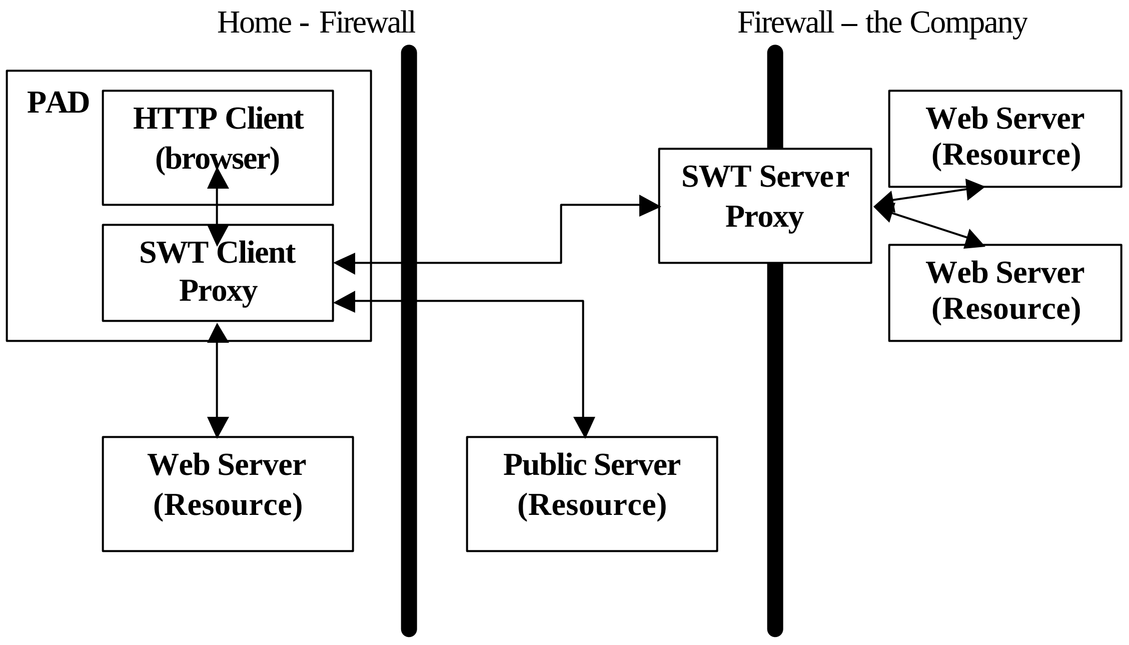


Figure 3-2 Multiple Target Domains

In the example, John has:

1. Some resources (like his work-related documents) in his office computer. This system is behind his Company's firewall.
2. Some other resources (like his personal files) in his home computer. The home computer is behind his Home environment's firewall.

3. Other resources (like his public e-mail account) in the open sub-net.

In such an environment, each of the secure domains can have an SWT server. The SWT client maintains routing information for the different SWT servers. When a connection is requested for a specific domain, the SWT client uses the appropriate Certificate & route and enables SSL. Thus in the scenario described above, the SWT client maintains a table of SWT Server addresses based on domain names. In the case of the Open sub-net case, there is no SSL needed – the SWT client acts as a standard http proxy server (no tunneling).

3.3 Controlled Access Rights to Users

So far this section has described SWT providing secure access for the user (to his resources). That is the user is protected from malicious/accidental access to his data. However, the current location domain's systems need to also be protected from the user. The user (since he is from a different domain) is not trusted – he needs to be:

1. Allowed access to systems outside of this current domain (as explained in the previous sub-section).
2. Allowed controlled access to certain resources in the current domain. For example, access to the Conference Room's printer & projector.
3. Denied access to the domain's protected resources. For example, if John (from Company A) is visiting Company B. Company B does not want John to be able to access the Company B's employee phone book.

All this can be achieved by replacing the current location's proxy server with an SWT server. This server can be configured to route the http requests as stipulated above. Figure 3.3 is an illustration of controlled access.

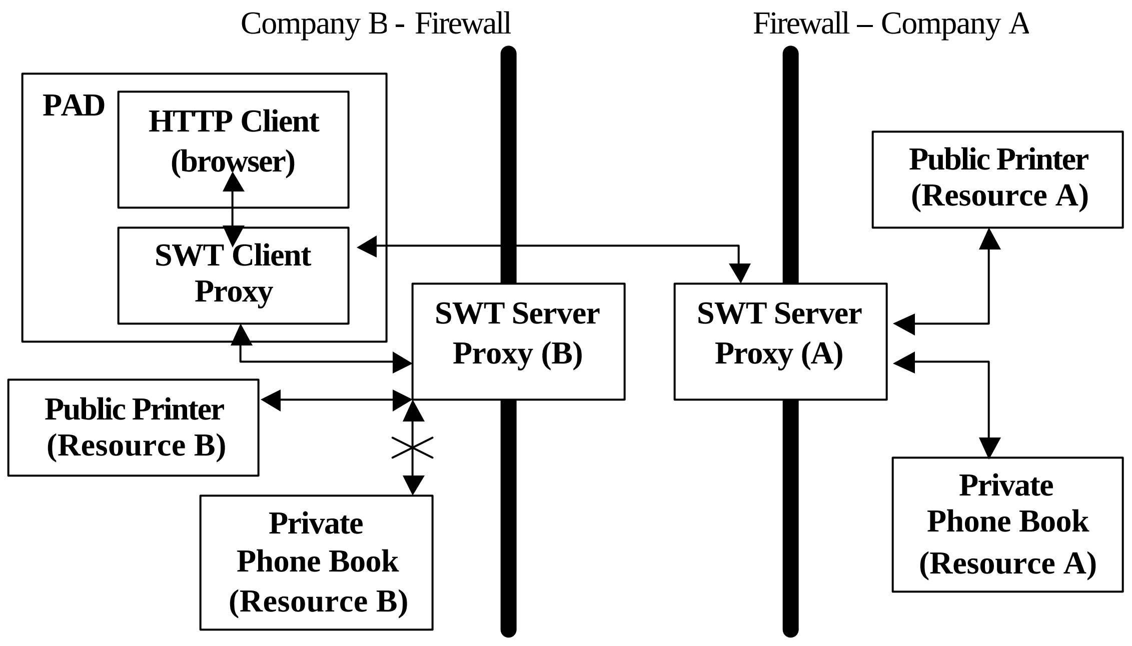


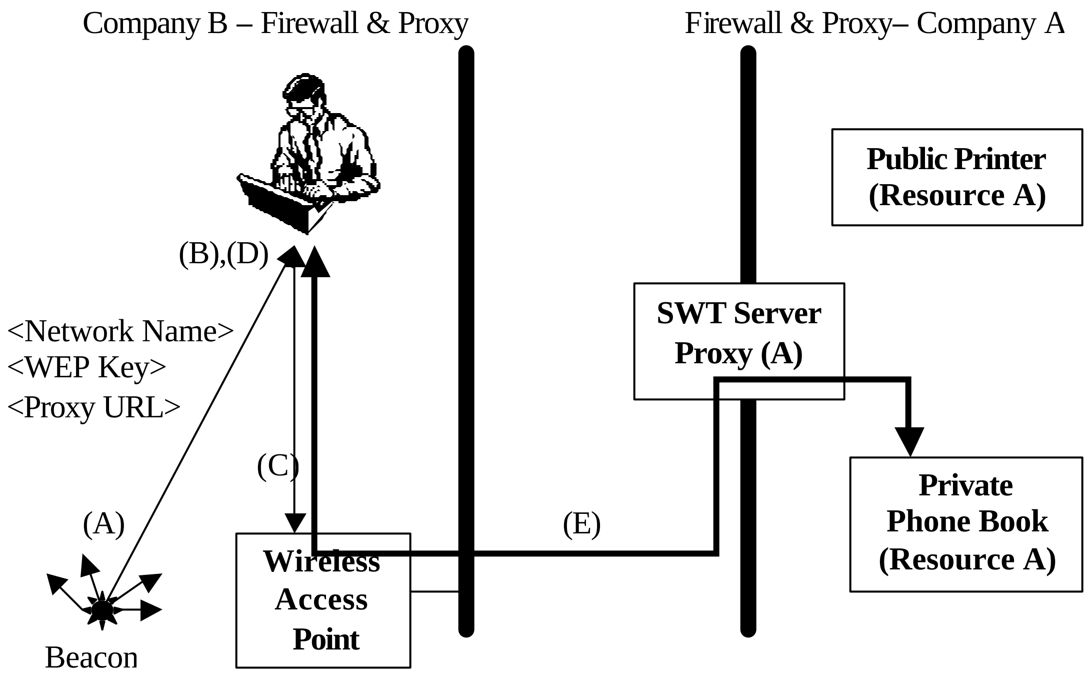
Figure 3-3 Controlled Access

3.4 Wireless Integration

By using Wireless LANs (like 802.11b), the nomadic user can have convenient network connectivity. To facilitate secure access, SWT can be integrated with 802.11b access points. As described in the previous section, the 802.11b has the notion of network name & WEP key. These settings are specific to a given domain. The user on entering a new domain has to configure his PAD with these settings – only then can be gain network access. Further he needs to configure his SWT client with the proxy server settings of this domain. Manually setting all these parameters while feasible is cumbersome. The user will have to get this information (from the local system administrators) & then manually set up the configuration. An alternate approach is the E-Squirt based Auto-Configuration scheme.

3.4.1 Network & SWT Auto-Configuration

E-Squirt beacons are placed in the connecting domain physical environment. The beacons broadcast periodically some tags. The tags relevant to Auto-Configuration are: (1) Network Name (2) WEP Key & (3) The URL that contains the proxy settings for SWT.
The nomadic user's PAD has a beacon reader. The PAD reads the tags and (a) configures & activates the 802.11b connectivity and (b) configures its SWT client by getting the web document that provides the domain's proxy settings. The SWT based secure system is now in place.



Company B – Firewall & Proxy
Firewall & Proxy– Company A
Public Printer (Resource A)
(B),(D)
SWT Server Proxy (A)
<Network Name>
<WEP Key>
<Proxy URL>
(C)
Private Phone Book (Resource A)
(A)
(E)
Wireless Access Point
Beacon

Figure 3-4 SWT Protocol

Figure 3.4 shows the steps taken to obtain a secure connection.
(A) Beacon broadcasts Information Tags (<Network Name>, <WEP Key>, <Proxy URL>)
(B) The PAD reads the beacon

(C) Configures & sets up its WaveLAN network – show the Access Point
(D) Configures the SWT client after getting the Proxy settings from the access point
(E) Then gets secure access.

3.5 Prior Systems

Prior solutions exist to enable this level of secure connectivity. The Digital SWT [Abad99] is very simple, but does not work through multiple firewalls. Unlike the previous model, the ATT SWT [Gilm99] does not require any modification of the client's browser and the firewall. It builds all the modules complementary to already existing software. The disadvantage is the necessity to modify the client's configuration (browser) every time it accesses resources protected by other proxies.

3.6 Test-bed

The SWT was implemented in java and tested on a simulated enterprise environment. The test-bed contained two private companies (Hughes' and Anca's) with wired and wireless networks, and the rest of the Internet (open-subnet) between them.

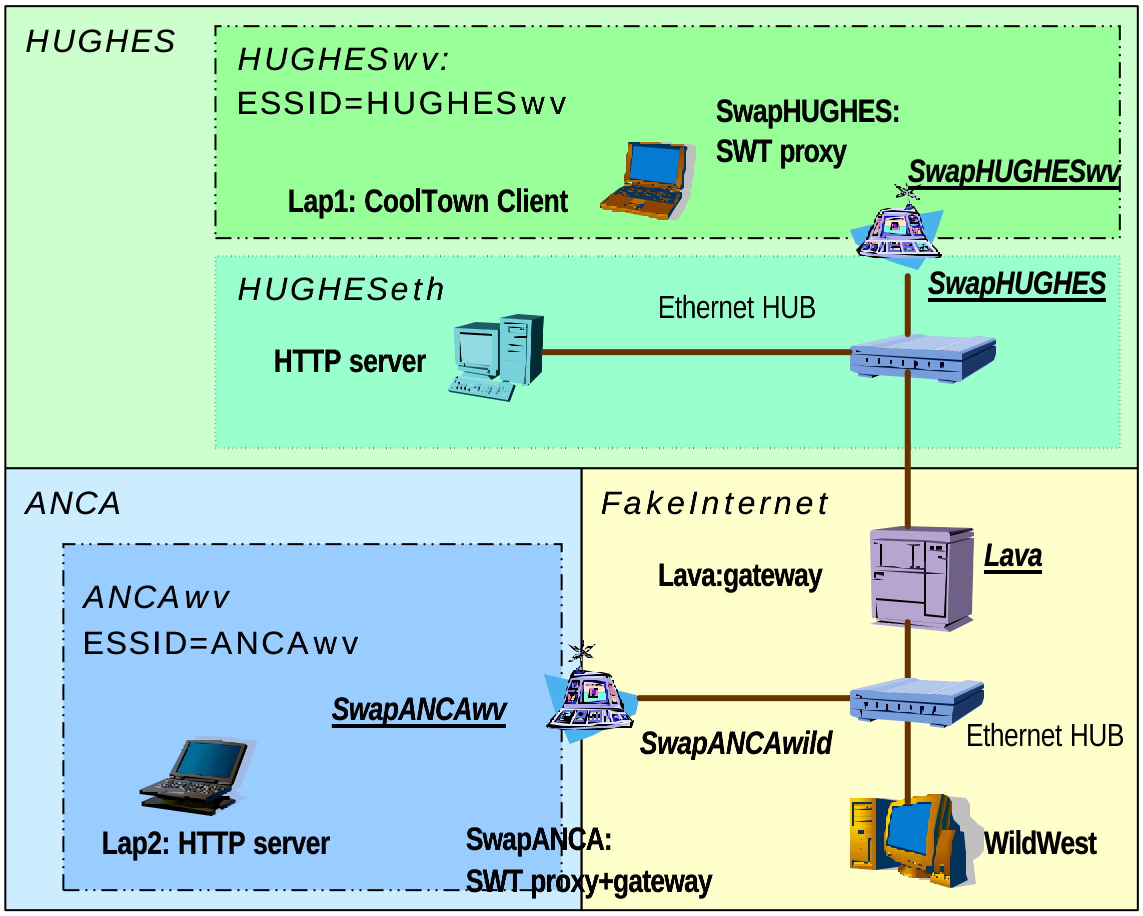


Figure 3-5 Test-bed

3.6.1 Scenario 1

Lap1 (employed by ANCAwv and located inside HUGHESwv) connects to Rama (owned by HUGHES and located inside HUGHES). In this scenario, Lap1 is the client and Rama is the web server. Lap1 does not have the right credentials to access a resource inside HUGHES and his request is denied.

3.6.2 Scenario 2

Lap1 (employed by ANCAwv and located inside HUGHESwv) connects to Lap2 (owned by ANCAwv and located inside ANCAwv). Lap1 is the client and Lap2 is a web server. Lap1 possesses the required certificates and is able to start his secure connection with the web server.

4 Conclusions

As seen in the previous scenarios, the HP implementation of SWT ensures secure communication between a client and a resource through multiple firewalls and does not require any modification on the client and the server. The automatic configuration feature based on the E-Squirt eliminates security risks and does not require special knowledge on the user's part. The SWT builds on the secure connectivity provided by SSL & exploits the proxy for tunneling. However it requires an added installation of the SWT Client Proxy on the client's machine. Some next steps include: (1) Simplifying the SWT Client proxy. This may reduce the requirements on the PAD, (2) Incorporating non-IP based PADs like WAP & BlueTooth and (3) Defining the Certificates & SWT server mechanisms.

5 Appendix A

This section provides an overview of the different protocols used in the security domain. It also provides an analysis of their features. Finally, it looks at the use of these protocols in this paper's problem domain – nomadic environments.
Internet would not have developed so fast if IETF had not fought to preserve user's privacy and secrecy. Unfortunately, there is no universal panacea. The protocols designed by IETF and implemented by the security community are each trying to address and resolve a small area of problems in the immense universe of Internet attacks. IPSEC secures the communication between machines; SSL/TLS creates encrypted channel between applications; SSH offers a secure solution to the problem of remotely access resources.

5.1 Internet Protocol Security (IPSEC)

This architecture was designed to provide various security services for the traffic between two machines at the IP layer. IPSEC [CoSe99] was developed to provide interoperable, high quality, and cryptography based security for Ipv4 and Ipv6. IPSEC [Ipse00] uses:

1. ***Authentication Headers*** (AH) for connectionless integrity, data origin authentication, and an optional anti-replay service.
2. ***Encapsulating Security Payloads*** (ESP) for confidentiality, connectionless integrity, data origin authentication, and an anti-replay service, and
3. The ***IKE*** protocol to set the necessary parameters and authenticate the two devices involved in the communication.

At initialization, the two communicating nodes need to select the required security protocols, to determine the algorithm(s) to use for the service(s), and put in place any cryptographic keys required to provide the requested services. There is a lot of information to be managed and IPSEC handles this task by using Security Associations. A ***Security Association*** (SA) is a data structure that describes which transformation will be applied to a datagram. Typically, the SA contains [CoSe99]:

- The authentication algorithm (AH or ESP)
- The encryption algorithm for ESP
- The encryption and the authentication keys
- Lifetime of encryption keys
- The lifetime of SA and
- The destination host's address

SA is unidirectional. To ensure typical, bi-directional communication between two hosts, or between two security gateways, two SAs (one in each direction) are required. All the parameters (listed above) need to be negotiated before the actual secure communication commences. IPSEC gives the user two options for the initial configuration: manual or automatic. Automatic Configuration is performed by IKE a hybrid protocol which implements the Oakley key exchange and Skeme key exchange inside the ***Internet Security Association and Key Management Protocol*** (ISAKMP) framework.

After all the necessary parameters are established, IPSEC provides protection to the IP traffic, based on the requirements defined by a ***Security Policy Database*** (SPD) established and maintained by a user or system administrator. Each packet is transformed according to IPSEC security services, discarded, or allowed to bypass IPSEC, everything based on the applicable database policies. The SPD is consulted during the processing of all traffic (inbound and outbound), including non-IPSEC traffic.

According to a report from the industry research firm GIGA Information Group, "The IPSEC protocol will be most widely used to provide ***network-level*** packet encryption and authentication for remote access over the next three years". The reasons for such optimism are transparent security mechanisms, interoperability, and modularity.

5.2 Secure Sockets Layer / Transport Layer Security (SSL/TLS)

SSL [SSL96] was developed by IETF as a solution to some of the most important security problems of the Internet. It provides privacy, integrity, authentication, and authorization. Unlike IPSEC, which is a replacement protocol for the IP layer, with new security characteristics added on top, SSL is a new protocol layered on top of TCP/IP, with a unique scope: security. SSL is a client-sever protocol with very well defined roles. The client initiates the communication and offers the available options to the server. The server chooses which option will be used and sends its final decision to the client.
SSL can be used to:

- Establish encrypted communications
- Authenticate the server's identity and
- Authenticate the client' identity

Even though the protocol is fairly simple and straightforward, it is designed to resist against a certain number of attacks, including the man-in-the-middle attack. The client initiates the negotiation by sending a ClientHello packet. The server answers back by sending a ServerHello packet and the supported SSL options. In this scenario, the server chooses to authenticate by sending its public key certificate and requires for client authentication as well. The client answers by sending its public key certificate, its public key information, encrypted with the server's public key, and a signature of important information about the session; the server will verify the client's identity by decrypting the signature using the client's public key. The client chooses the SSL options it supports from the list sent by the server, but the server is the one that takes the final decision. After all these negotiations, the secure channel is opened and ready to use. As pointed out by [SSL96], TLS is not a new security protocol developed by IETF. There are some differences between SSL and TLS, as highlighted in Annex A, but the protocols are basically identical. TLS [TLS99] is considered to be SSL version 3.1 and was designed to be backward compatible.

5.3 Secure Shell Protocol (SSH)

SSH was initially designed by IETF for secure remote login, TCP and X11 forwarding [Pete98] and other secure services. Before SSH, the most used tools for remote access to resources were telnet, rlogin, rsh and rcp and they were well known for their security weaknesses: the data was sent in clear over an insecure Internet, there was no machine authentication, and the only authentication was based on IP addresses [Pete98]. All these security flaws open the remote sessions to eavesdropping, IP address spoofing, DNS spoofing, and manipulation of data at router level. Even if SSH protects users against these types of attacks, there are a few situations where SSH is helpless:

- Incorrect configuration
- Misusage of SSH
- Attacker has root rights on either the client or the server and
- Attacker has the access rights to modify files in the client's home directory

Like SSL/TLS, SSH is a client-server protocol [SSHA00] with client and server authentication. In SSH, it is very important that all servers have host authentication keys. The client can verify the server's identity only if it knows the <host name, host key> pair, a priori. SSH supports several trust models:

1. The client has a database with all <host-name, host key> pairs. The advantage is that the client has all the information it needs for server identification and it is necessary for the initial exchange of messages. The downside is that the database is static, and it needs to be updated every time a server is changed, created or destroyed. Also, the database needs to be stored in a secure fashion and with fast access whenever needed.
2. A certificate authority trusted by the client certifies each <host name, host key> pair. In this scheme, only the CA's key needs to be stored in a secure place.
3. The client can choose to ignore the authentication step, when connecting to the host for the first time (not recommended).

The three major components of SSH are:

1. ***Transport layer protocol*** [SSHT00] provides strong encryption, host authentication, and integrity protection. It does not offer user authentication.
2. ***User authentication protocol*** [SSHU00] runs on top of the transport layer protocol and starts only after (1) the server is authenticated, (2) a secure channel is established, and (3) a unique session identifier is computed. Its only goal is the client's authentication. The server supports multiple authentication methods and gives the client the freedom to choose the ones it considers the best:
   - 2.1. *Public key authentication method*: The client has a <public key, private key> pair and creates a signature using its private key. The server checks the signature using the client's public key. If the signature is valid, the server accepts the connection. Otherwise, the connection must be rejected.
   - 2.2. *Password authentication method*: The server manages a <user, password> pair database. If asked, the client supplies the server with a name and a password. The server looks up for the pair in its database and accepts/rejects the connection based on the answer of its query.
   - 2.3. *Host-based authentication method*: Each machine has a different <public key, private key> and the client uses the host's private key to create a signature. The server verifies only this signature. The authentication step finishes here, but, later, the server checks the client's authorization, based on the user's name.
3. ***Connection protocol*** [SSHC00] runs on top of the User Authentication protocol and Transport Layer Protocol. Its only role is to multiplex one physical connection into

several logical channels. The user can open remote sessions, execute remote commands, and open remote X11 windows.

5.4 Security Protocols Analysis

Due to the varying characteristics of the security attacks in the Internet world there is no universal panacea. Each protocol presented above is focused on solving a subset of the security issues with its own advantages and disadvantages. In order to make an educated choice, the problem to be tackled is first defined. Then the protocol(s) to be applied is identified.

From all protocols enumerated above, IPSEC is the only one that runs at the network level and works for UDP, TCP/IP, or any other protocol that uses IP, while SSH and SSL/TLS require the reliability offered by TCP/IP. **SSH is ruled out** by the nature of the problem space – A Web (http) oriented view of the world. SSH was designed for remote login access, remote execution of commands, not for Web access. IPSEC creates secure channels between two machines. It assumes that security is a function of the particular system and that all applications within that system need the same security services. Every packet that circulates between those two machines will be either encrypted or not, depending on the negotiated policies. The machines are not capable of deciding at the IP layer which packet belongs to which application and take a decision based on that. Therefore, the user will not be able to open simultaneous secure and non-secure channels between the browser and the web server.

SSL [SSLT00] was designed to build secure communication channels at the application level. Unlike IPSEC, which encrypts none or all packets between two physical machines, SSL can choose which packet to protect depending on its use. HTTP over the SSL layer provides secure Web communications. For example, a user wants to connect to his bank's web site and check his account. This particular channel needs to be a secure one. At the same time, the user opens another browser session to read about some new services offered by the same bank. This channel does not require security because the information is publicly available. HTTP over SSL can handle this scenario. In an IPSEC based environment, all channels can be either protected or not. There is no way of distinguishing between sessions and taking different actions depending on that.

In SSL, the client & server exchange messages and establish a channel for encrypted communications. This is the basic core feature of SSL and is performed by exchange of Digital Certificates that authenticate the client & server entities. By layering HTTP over SSL, users (with Digital Certificates) from web browsers can have secure access to Web resources.

In a nomadic environment, the network configuration for the Client is based on the policies of the system administrators of the infrastructure provider. Connectivity in a Cyber Café will be in the open subnet while access in a company will be from within a firewall. The IP addresses & the network policies will not be consistent across these diverse network providers. Deploying a consistent IPSEC across these heterogeneous systems is not practical. On the other hand, SSL provides application level security with user authentication. For SSL the client machine's IP address is not critical, authenticates the user& the server resources based on Digital Certificates.

Web (Http) based access is widely deployed. HTTP proxies provide a well-established conduit for access to Web resources outside of environments with firewalls [Wps98]. By combining SSL & proxy servers, secure access is enabled in a web environment with firewalls. The Secure Web Tunnel (SWT) is such a solution.

A tabular comparison between the three protocols can be found in Appendix D.

6 Appendix B - The Wireless Networks

Nomadic users in a pervasive computing world like CoolTown use wireless networks for Web Connectivity. Wireless networks use Infrared (IR) or Radio Frequency (RF) technologies for network connectivity. IR & RF technologies have distinct different characteristics and are well suited for specific application domains - based on usage & environment.

6.1 Radio Frequency Technologies

RF signals travel through walls and are preferred where there is no direct path between machines. Standards like the *IEEE 802.11b* define specifications based on RF technologies for wireless LAN networks [Wlan99]. The *802.11b* standard has characteristics that match well for IP based traffic – high bandwidth, low latency [W80295]. *802.11b* based LANs are being widely deployed and several companies provide products/solutions (ex: Lucent's WaveLAN, Aironet) for corporate, public & home environments.
The components of a wireless LAN consist of (1) a wireless Network Interface Card (NIC) installed on the Personal Access Device (PAD) & (2) a wireless local bridge, which is often referred to as an *access point*. The access point acts as bridge between the wired & wireless networks.

6.1.1 Security in *802.11b*

Because of the open broadcast nature of wireless LANs, *802.11b* defines 2 mechanisms to: (1) constrain access to the network & (2) protect the network data from outsiders. Wireless networks have options that need to be configured in the PAD for participation in the infrastructure.

2. Extended Service Set (ESS) Network - A set of *802.11b* access points can be configured (by the system administrator) to belong to a ESS Network – identified by an ESS ID or *Network Name.* Only PADs with knowledge of the *Network Name* are allowed to participate in the Network.
3. Wired Equivalent Privacy (WEP) – Because of the open broadcast nature of wireless LANs, *802.11b* describes a symmetric key based authentication service called WEP. WEP is a shared key mechanism – all PADs share a key and data in that network is encrypted with that key. This prevents network eavesdropping.

6.1.2 Wireless Configuration Issues

In a nomadic environment, a user wandering across different network providers/domains needs to configure his PAD with a new Network Name & WEP Key every time he enters a new network. This can be a cumbersome step requiring the user to have an understanding of the *802.11b* network system of each of the network domains.

6.1.3 *802.11b* Security Solutions' Problem Domain

The *802.11b* Security mechanisms prevent unauthorized users from participating in the network. Encrypting the data prevents non-participating wireless users from eavesdropping on the wireless media. However, these mechanisms do not protect users that belong to the network from one another. All PADs in a given network share the same encryption key.
In a nomadic connected world, users with different trust relationships need to participate in the same network. *802.11b* does not provide a solution for such a secure environment. The SWT solution described in the next section addresses this problem.

6.2 Infrared Technologies

The IR transceivers propagate information by light in the invisible part of the frequency spectrum. It does not penetrate any solid material and it works for less than 3 feet of line of sight of the connection. The Infrared Data Association (IrDA) specification is an interconnection standard that defines protocols for IR based devices. IrDA can be used (configured) as a wireless LAN environment. However as IR is directional & with short range, it is better suited for point & shoot environments.
CoolTown's underlying Appliance Computing Architecture (ACA) [Aca00] exploits IR for setting the context of Web connectivity. ACA introduces the notions of Beacons & Squirts (E-Squirt) [Esq00] – using URLs to establish a Client's context. A Beacon sender broadcasts, periodically a URL that defines the context that it represents. A receiver in a PAD receives beacons & uses the beacon URL to establish its context. As the Beacon sender protocol is very trivial, it can be implemented as a simple low-powered, low-cost device. In CoolTown, hardware beacons are used to define the notion of a Space. A Users with a PAD entering a Space, accesses Space specific services. This Space context is established in the PAD by receiving a beacon. The beacon has the URL of the Web server that represents the Space. The PAD (if has network access) can then retrieve this web page.
IR's characteristics (directionality, fast discovery, light-weight stack, low cost) make it a good fit for E-Squirt.

7 Appendix C

No.	SSL (Secure Sockets Layer)	TLS (Transport Layer Security)
1.	The last SSL protocol has version number 3.0.	TLS uses protocol number 3.1.
2.	SSL has error message 41: "NoCertificate"	TLS eliminates this message and adds a dozen other values.
3.	SSL supports the Fortezza/DMS cipher suite.	TLS does not support the Fortezza/DMS cipher suite, but it offers a very easy way to create and install new cipher suites and compression methods.
4.	SSL computes the CertificateVerify hash using the master secret.	TLS has a slightly different hash calculation for the CertificateVerify hash. It does not use the master secret (special value computed for the CertificateVerify message).
5.	The Finished message is formed by two hash values.	TLS uses a slightly different algorithm to compute the hash for the Finished message.
6.	SSL uses either RSA's Message Digets 5 (MD5) or Secure Hash Algorithm (SHA) to compute MAC.	TLS uses the Hashed Message Authentication Code (H-MAC) algorithm to compute MAC.
7.	SSL uses the master secret (computed for CertificateVerify message) to create the cryptographic parameters.	TLS uses a completely different algorithm to generate key material (H-MAC).

Appendix C - Comparison between SSL and TLS

8 Appendix D

IPSEC	SSH	SSL/TLS
IPSEC secures the network layer, adding security regardless of the application. [Tech97]	SSH is a security protocol designed to run at the transport layer and the application level.	SSL addresses security issues for applications running on top of TCP/IP (at the application layer).
IPSEC works at the network layer; therefore, it can be used by itself or in conjunction with other security mechanisms (like SSL). [Cylan97]		
IPSEC can be used for different protocols, not necessarily based on TCP.	SSH is used mainly for secure remote login. It runs on top of TCP/IP or any other protocol that ensures reliability.	SSL is used for HTTP, NNTP, and FTP (restricted number of protocols, based on TCP). It does not work for UDP based protocols. SSL requires reliable transport.
IPSEC can be combined with e-commerce protocols (SET) to enhance the security.	SSH works for secure POP.	
IPSEC is a replacement protocol for IP, with new security characteristics added on top.	SSH is a new protocol, not based on pre-existent protocols.	SSL is a new protocol with its unique scope: security.
		SSL does not support non-repudiation (a party cannot falsely deny that it was the source of some data that it really created). [SSL00]
There is no difference between the two end nodes in the communication.	SSH is a client-server based protocol.	SSL is a client-server based protocol.
IPSEC has a very complex initial phase. Resuming the communication between two entities does not necessarily require a complete negotiation. Previous information (IKE security association) can be used. Only the IPSEC SA will be renegotiated.		SSL is a very complex process that requires an intensive interchange of messages. Restarting an SSL channel between a server-client pair does need a total renegotiation.

Appendix D – Comparison between IPSEC – SSL/TLS – SSH

9 Appendix E - Bibliography

1. [Abad99] **"Secure Web Tunneling"** –
Martin Abadi, Andrew Birrell, Raymie Stata, Edward Wobber
Systems Research Center
Digital Equipment Corporation
http://www.research.digital.com/SRC/personal/birrell/tunnel/206.html

2. [Aca00] – "**Appliance Computing Environment Architecture Overview**"
http://www.cooltown.hp.com/papers/jam/ApplianceComputingArchitecture.htm

3. [Cool00] – "**CoolTown – Web Presence for People, Places & Things**"
http://cooltown.hpl.hp.com

4. [CoSe99] **"SSH IPSEC Express – White Paper version 2.1.1"** –
SSH Communications Security Ltd
http://www.ipsec.com/tech/whitepapers/ipsec-wp.pdf

5. [Esq00] – "**The CoolTown E-Squirt API**" -
http://www.cooltown.hp.com/code.htm

6. [Gilm99] **"Secure Remote Access to an Internal Web Server"** –
Christian Gilmore, David Kormann, Aviel D. Rubin
ATT Labs – Research
IEEE Network – November/December 1999
http://www.research.att.com/resources/trs/

7. [Ipse00**] "White Paper – IPSec – Executive Summary"** –
Cisco Systems Inc., 2000
http://www.cisco.com/warp/public/cc/techno/protocol/ipsecur/ipsec/tech/ipsec_wp.htm

8. [Pete98] "**Pete's Wicked World**" –
Pete Galvin
http://www.sunworld.com/swol-02-1998/f_swol-02-security.html

9. [SSHA00] – "**SSH Architecture**" –
INTERNET DRAFT, draft-ietf-secsh-architecture-07.txt
Network Working Group, IETF, May 11th 2000

10. [SSHC00] - "**SSH Connection Protocol**" –
INTERNET DRAFT, draft-ietf-secsh-connect-07.txt
Network Working Group, IETF, May 11th 2000

11. [SSHT00] - "**SSH Transport Layer Protocol**" –
INTERNET DRAFT, draft-ietf-secsh-transport-07.txt

Network Working Group, IETF, May 11th 2000

12. [SSHU00] - "**SSH Authentication Protocol**" –
INTERNET DRAFT, draft-ietf-secsh-userauth-07.txt
Network Working Group, IETF, May 11th 2000

13. [SSL96] - "**Secure Sockets Layer Protocol Version 3.0**" –
Alan O. Freier, Philip Karlton and Paul C. Kocher,
Netscape Communications Corporation, March 4th 1996
http://www.netscape.com/eng/ssl3

14. [SSLT00] – "**SSL and TLS Essentials**" -
Stephen Thomas
John Wiley & Sons, Inc.

15. [TLS99] - "**Transport Layer Protocol Version 1.0**" –
T. Dierks and C. Allen – RFC 2246
Network Working Group, IETF, January 1999
http://www.ietf.org/html.charters/tls-charter.html

16. [Wlan99] – "**Wireless LANs**" -
Jim Geier
Macmillan Network Architecture & Development Series.
John Wiley & Sons, Inc.

17. [Wps98] – "**Web Proxy Servers**" -
Ari Luotonen, 1987
Prentice Hall, NJ.

18. [W80295] - "**Project 802 – Local & Metropolitan Networks**" –
Editors of IEEE 802.11
Network Working Group, IETF, July 1995
http://www.ietf.org