USENIX Association

Proceedings of the 17th Large Installation Systems Administration Conference

San Diego, CA, USA
October 26–31, 2003

A Secure and Transparent Firewall Web Proxy

Roger Crandell, James Clifford, and Alexander Kent – Los Alamos National Laboratory

ABSTRACT

The LANL transparent web proxy lets authorized external users originating from the Internet to securely access internal intranet web content and applications normally blocked by a firewall. Unauthenticated access is still, of course, denied. The proxy is transparent in that no changes to browsers, user interaction, or intranet web servers are necessary. The proxy, a few thousand lines of C running on Linux, has been operating within Los Alamos National Laboratory's firewall since 1999.

Introduction

Our goal was to provide a transparent firewall web proxy, a proxy allowing authorized and secure access to the many firewall-protected intranet web servers at Los Alamos National Laboratory (LANL). Our offsite users required access to internal news, phone directories, web email, the online library, and administrative services. Additionally, they need to download documentation and software. External users often find themselves in the situation where they are forced to use available computers at the sites they are visiting. Non-standard access methods, including browser configuration modification or installing virtual private networking (VPN) software was often impractical or undesirable. At the same time, it was very important to maintain the security of our intranet servers and their data. Only requests from authorized users would be forwarded and all information sent outside the firewall had to be encrypted. We wanted the web proxy to function within the framework of the existing firewall.

We considered several approaches.

- We could make exclusive use of virtual private networking to access internal web servers.
- A proxy could be designed to work only for a single web server, making the design easier.
- Selected web services could be relocated outside the firewall.
- Rules to allow direct access through the firewall to internal web servers could be allowed. These rules could be permanent or dynamic in nature. Dynamic rules would be installed automatically following authentication and removed after a specific time period. Rules would be based upon external IP address and internal web server IP address.
- We could not allow external access to our internal web servers.
- A transparent web proxy could be provided.

We concluded a transparent web proxy would meet all our requirements. The next question was how to implement it in a secure fashion while still providing simple and reliable operation.

This paper focuses on how these needs were addressed through the design of an innovative and user-friendly proxy using available technologies and open source software.

Goals

Controlled access to intranet web servers from the Internet was the goal. VPN access into the intranet was available, but only provided access to users who had the proper VPN client software installed. In addition, a VPN solution was seen as overkill for those users requiring just web-based intranet access. Clearly, there was an accessibility gap needing to be filled. What was needed was an access mechanism that worked with any browser, anywhere. We desired simplified access while also ensuring client transparency, encrypting all data transmitted over the Internet, providing user-based authentication, and requiring no modifications to existing intranet web servers. The authentication method must continue to work in the same manner as the existing firewall. A web single sign on was determined to be a requirement. Once an individual authenticated to the web firewall proxy, they would not need to authenticate again to internal web servers. The authentication on the web proxy would provide authentication credentials necessary to access all internal servers.

Design

Meeting the transparency requirement was our first task. Accomplishing this requires all inbound intranet web requests to traverse a path allowing evaluation of the connection and forwarding as necessary. This evaluation could be done with a broadcast network firewall DMZ where all the incoming HTTP requests would be visible. The proxy could then intercept the requests from the broadcast medium. Another option is to use a stateful Linux firewall [14] which would also act as the proxy. However, we chose a third option: using a router with policy routing [18]. The policy routing rules route packets based upon destination port numbers and sends all HTTP (port 80) and HTTPS (port 443) requests to the proxy server.

The standalone proxy server makes use of the Linux operating system with netfilter [14] and iptables rule set. Through the insertion of iptables rules such as:

```
iptables -t nat -A PREROUTING \
  -ieth0 -p tcp -d INTERNAL_NET \
  -dport 80 -j REDIRECT -to-ports 80
iptables -t nat -A PREROUTING \
  -ieth0 -p tcp -d  INTERNAL_NET \
  -dport 443 -j REDIRECT -to-ports 443
```

the proxy server can accept connections for an entire network and respond using the IP address of the intended destination server. Several router vendors also provide layer-4-based policy routing that provides similar transparent redirection. This policy routing is commonly used for outbound transparent HTTP proxying [4].

The second task was to force the use of SSL encryption on all incoming HTTP connections. This is accomplished by running a redirector daemon on the proxy server. The redirector listens on port 80 and redirects port 80 connection requests to port 443 on the same requested server. To accommodate the fact that the proxy masquerades as many web servers, a wildcard X.509 [13] certificate is installed on the proxy. The certificate has the form *.lanl.gov, Los Alamos' domain name. This simple redirection from HTTP to HTTPS guarantees all data requests from our external users are encrypted.

Once the port 80 to 443 redirect has been issued, knowledge of the original port requested is lost. All web requests arrive at the proxy on port 443, even the ones that started out on port 80. The proxy must make a decision about which intranet web server port it should connect to following authentication. Keeping state information about previous requests was deemed to be problematic, and a simpler approach was needed. It was decided to first attempt a port 443 connection to the intranet server, and look at the HTTP return code [7]. If the return code was 404, indicating the content is not found, the proxy then attempts a connection to port 80 on the intranet server. If the return code is again 404, this is returned to the web browser. If the return code is not 404, the proxy forwards the server's data to the web browser. Figures 1 and 2 indicate the different possibilities described above. Obviously, the proxy does not support intranet web servers running on non-standard ports.

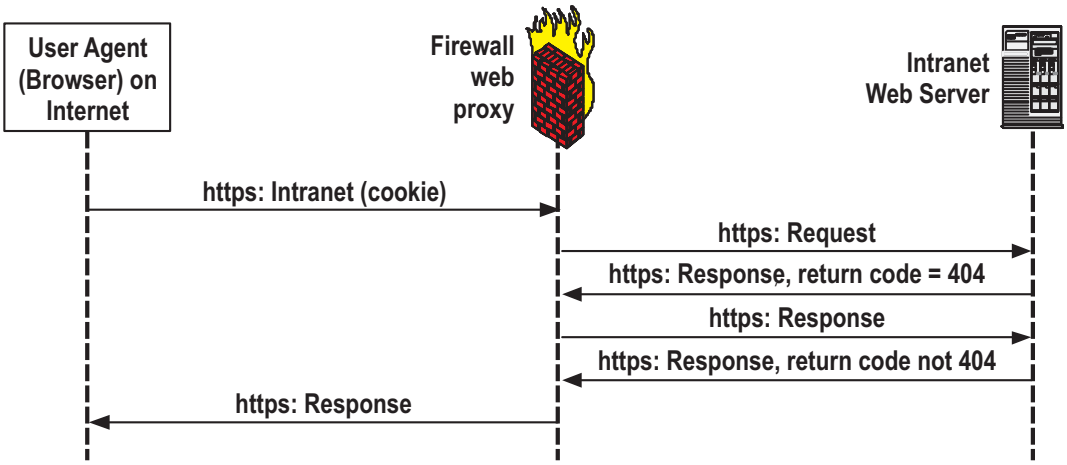


Figure 1: An http request with valid authentication cookie, internal server content on port 80.

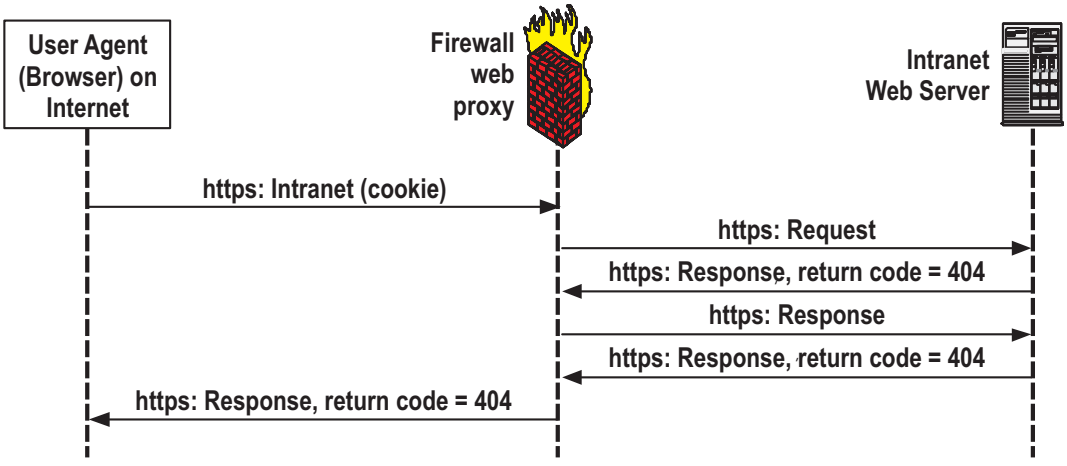


Figure 2: An http request with valid authentication cookie, content unavailable on internal server.

Third, the proxy needs a web authentication system to tell whether or not to forward the user's HTTPS requests. One method we have seen is to remember authenticated users by source IP address. The drawback to this method is you allow access to an IP address, not necessarily a specific user. Connections from remote sites using network address translation (NAT) [14], for example would allow multiple IP addresses access because NAT changes all internal IP address to a single external IP address. We needed to authenticate individual users rather than IP addresses.

The authentication methods used by the web firewall vendors we investigated [1, 2, 3, 4, 5] were deemed unacceptable. Their methods were either unacceptable from a security standpoint, or their authentication scheme did not fit into our existing authentication methods. Consideration was given to the HTTP CONNECT [12] method, which tunnels HTTPS and bypasses normal application layer functionality. This method presents a set of security issues, which we choose to avoid. Final design choices were two-factor authentication, a transparent proxy, and a secure authentication methodology. Implemented correctly, these would guarantee reliable access only to authenticated users and not IP addresses.

This authentication service, at a minimum, must provide the following capabilities. First, the proxy must be able to tell if an HTTPS request comes from an authenticated user session. Second, if the HTTPS request does not include valid authentication credentials, the proxy needs the URL of a web site that handles an initial, interactive user login.

There are two popular methods for keeping track of authenticated web sessions: one camp uses Kerberos tickets [16], (RFC 2478) [10], and one uses cookies [6, 7, 8], RFC 2965 [9]. Since the authentication system determines what is allowed through the proxy, careful design [17] and implementation is crucial.

LANL had an existing cookie based central web authentication service with the following security features:

- Two-factor CRYPTOCard [11] tokens with one-time passwords are used instead of traditional reusable passwords.
- The cookie sent to the browser is not persistent.
- The cookie string is random and contains no intrinsic value or information beyond being reasonably unique and hard to guess.
- The browser is instructed to send the cookie only to "*.lanl.gov" sites and only within an SSL session.
- The cookie is bound to the browser's IP address.
- Every new HTTPS request causes the cookie to be re-verified.

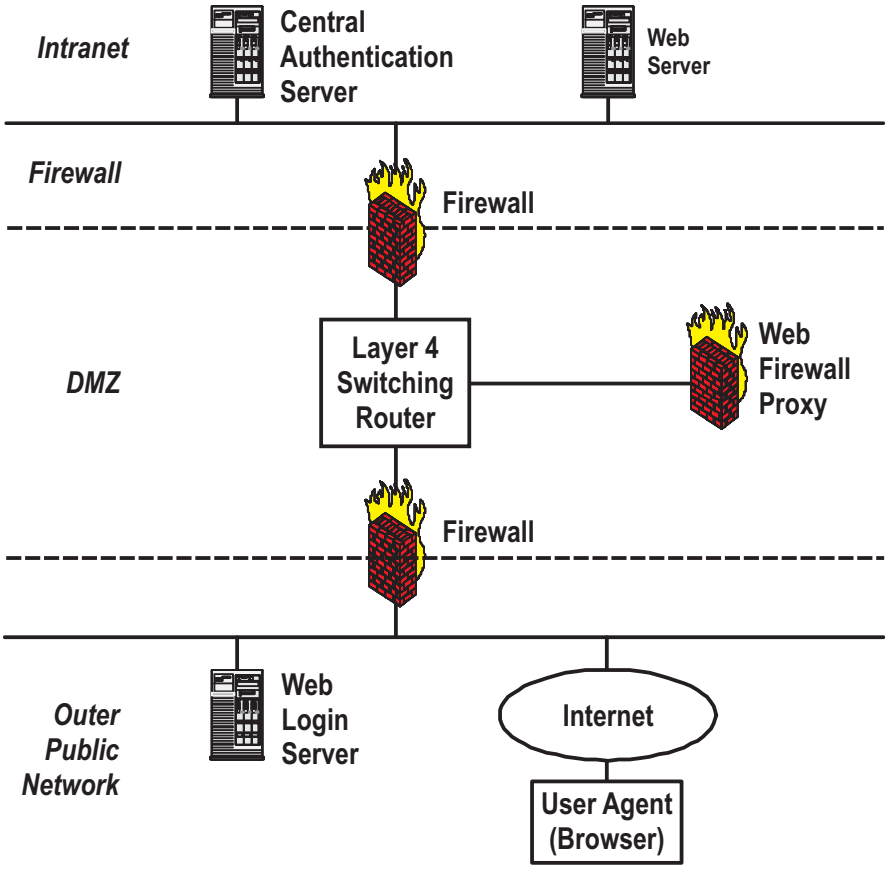


Figure 3: Web firewall architecture.

- To address the risk posed by a user walking away from a logged in session at an Internet kiosk, cookies have a maximum server-side lifetime after which the user must login anew.
- Cookies are also invalidated (server-side) if they are not used (idle time out).
- The user can voluntarily log out.

These features lent themselves well to the authentication needs of the proxy so this preexisting system was used.

Implementation

Our actual proxy implementation is comprised of several parts. See Figure 3, Web Firewall Architecture for details.

The first part requires the use of policy based routing to deliver HTTP/HTTPS messages to the proxy server. The firewall Demilitarized Zone Network (DMZ) [15] is contained within a commercial router that provides layer-4 policy routing. Incoming packets addressed to URLs behind the firewall are policy routed to the proxy server using layer-4 rules. All destination IP traffic addressed to any IP address behind the firewall going to ports 80 or 443 is policy routed to the web proxy.

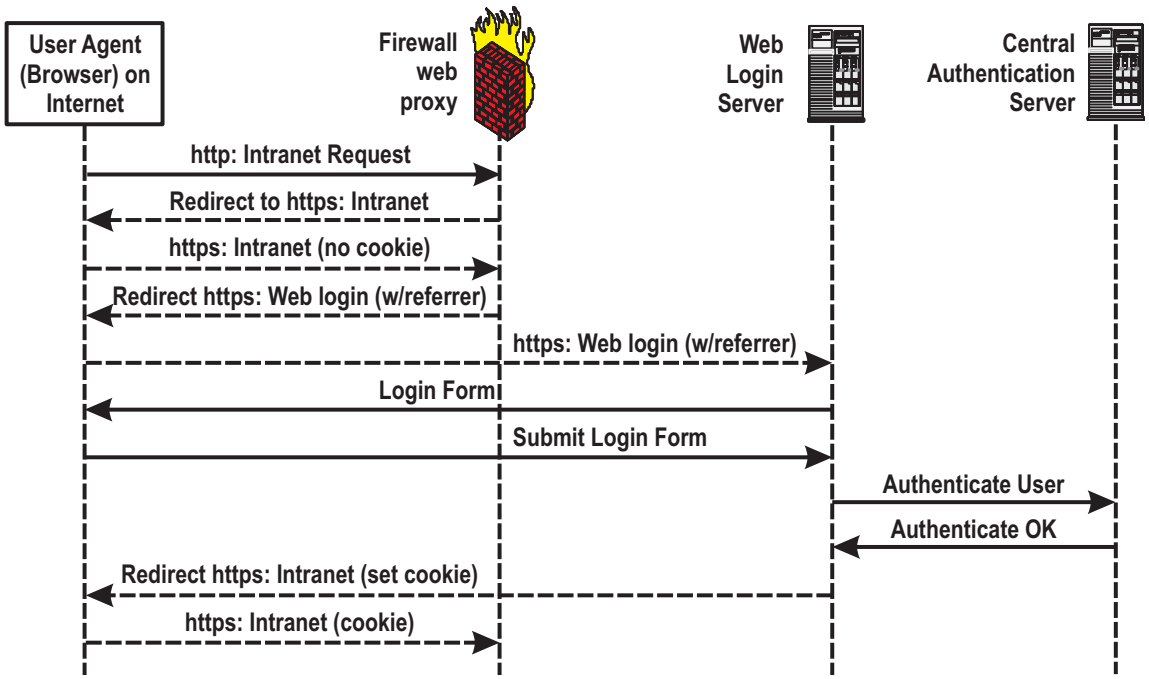
The second part is the web proxy server platform. We use the Linux operating system. Netfilter and iptables are part of the operating system. Netfilter runs within the kernel. Iptables provides the configuration interface to netfilter [14]. As described earlier, using iptables redirect, all packets policy routed to the web proxy are redirected to the web proxy. The proxy then masquerades as the internal web server when responding to the external web client.

The third part is the web authentication system. LANL's web authentication system is single sign-on. The session cookie obtained when the user authenticates works with all participating intranet servers. User authentication is two-factor authentication using CRYPTOCards [11]. This type of authentication makes use of one time passcodes. Following successful authentication, the password generated from the CRYPTOCard may not be used again. Our cookie attributes are:

```
Name = "SessionID",
VALUE = "Authentication random number",
Domain =".lanl.gov", Path ="/",
Max age = 0,
Secure = "1".
```

All other attributes are default values. The Max age attribute keeps the cookie from being stored on the web browser and causes the destruction of the cookie when the web client application is terminated. The secure attribute keeps the cookie from being transmitted in a non SSL request.

Cookies provide a security plus: the ability to revoke the authentication cookie at any time. Figure 4, Initial HTTP Request and Authentication, describes an initial connection without a valid authentication cookie. Figure 5, HTTP Request with Valid Authentication Cookie, describes a normal connection request that has a valid cookie. When an HTTPS request is forwarded though the proxy, the cookie is passed along too. Checking to see if a cookie is valid on the intranet



Note: Dashed lines are the result of redirects

Figure 4: Initial HTTP request.

servers can be tricky if the cookie is associated with the browser's IP address. The intranet web server sees the proxy as the user agent, but the cookie is associated with the end user's IP address. Our solution is to trust any cookie with a non-intranet address originating from the proxy. Integrating the proxy with other authentication systems may require a different approach.

Our web authentication system uses two Linux servers: a login web server to provide the interface allowing the entry of a CRYPTOCard-generated password, and a back-end authentication server to validate the password.

The fourth and primary component of the system is the web proxy daemons. They handle redirection of HTTP to HTTPS requests and the actual connection proxying. The proxy consists of approximately 2500 lines of C code. The first daemon, called REDIRD, listening on port 80, redirects port 80 intranet requests to port 443. The second daemon, WFD, handles the port 443 connections. WFD handles header parsing to collect the authentication cookie and response codes, connections to intranet web servers, connections to the central authentication server to verify cookies, and bi-directional forwarding of allowed browser and intranet web server traffic.

Reliability

In general, issues that affect other Internet service availability will apply to the proxy. The usual hardware, operating system, server software, networking and power concerns are relevant and are not discussed here. Attacks from the Internet, however, present some different problems for the proxy.

The good news is the proxy does not need to be directly addressable from the Internet. Policy routing directs only port 80 and port 443 traffic destined for intranet hosts to the proxy. The proxy itself cannot be scanned or attacked in the usual ways.

The bad news is that the proxy must respond for all intranet web servers. When someone scans your intranet address space for ports 80 or 443, the packets are routed to the proxy. The impact to a single host from this type of scan is minimal. In the case of the web proxy, scans are multiplied by the number of intranet web server addresses which are policy routed the proxy. Under the right conditions, a simple scan turns into a denial of service attack on the web proxy.

For example, suppose the proxy is set up to respond for and entire class B network to avoid the administrative chore of tracking intranet web servers. A network scan of the site's intranet web servers will send 64K requests to the proxy. This generates a bit of extra work, but the proxy can keep up as long as the requests are sequential and use 3-way TCP handshakes. If the scan involves a burst of SYN packets sent to 64K addresses, it turns into a SYN flood attack. Connection queues for the proxy's daemon fill up so additional connection requests are dropped.

One solution is to throttle connection requests from individual source IP addresses. Since the proxy is responding for many destination addresses, the rule to limit connections applies to any destination IP address. A second alternative is to dynamically block source IP addresses sending malicious packets. Finally, to minimize the problem, one can limit the number of addresses the proxy responds for to real intranet servers that need to be accessed from the Internet. This also improves security.

Performance

The proxy software makes light demands on the hardware. The application does some network I/O, some string manipulation, and SSL encryption. A modestly configured PC should be able to keep up with most site's workloads. Our proxy runs on a 800 MHz Intel Pentium with 512 Mbytes of memory and 100 Mb/s Ethernet and enough disk storage for a minimal Linux operating system and logging. With this hardware and current workloads, there is no user detectable difference in response in fetching web pages through the proxy and from within the intranet.

The web proxy currently proxies for one class B address range. There are approximately 1000 individual users who access the web proxy. The proxy typically handles 10,000 to 12,000 authenticated requests per day.

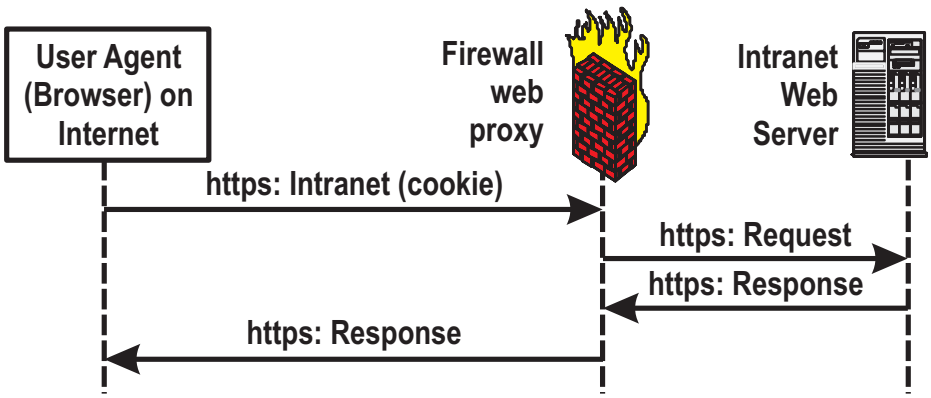


Figure 5: HTTP request with valid authentication cookie.

The average number of simultaneous connections is 12 with peaks to 36. We have seen no performance issues since installing the proxy. Port 80 and 443 scans of the class B address space do on occasion cause a slowdown in the proxy's response to valid connections.

The best way to improve performance is the same as improving availability: keep the Internet noise away from the proxy. Also, a layer-4 switch or policy router using proxy load balancing would allow for multiple proxies, resulting in scalable performance and improved availability for large applications.

Topology

The web proxy service consists of three pieces: a web login server, an authentication server (e.g., a Kerberos key distribution center-KDC), and the proxy itself. Where you place these components in your firewall architecture involves some security trade offs. The weblogin server, a real web server, is the only component that needs to be addressed directly from the internet.

Our weblogin server is outside the firewall, where it is subject to attack. If compromised, the intruder can capture names and passwords of users on the weblogin server and use them to get access through the web proxy. Intruder access is limited to firewall services and some users' accounts.

Placing the weblogin server inside the firewall with a special rule permitting access from the Internet creates a bigger problem if the server is compromised. The intruder is no longer limited to firewall services for certain users; you have an intruder inside your intranet. Our authentication server is inside the firewall, but is not directly accessible from the Internet. With the weblogin server outside the firewall and the authentication server inside, we have a special rule (hole) that allows them to talk. Capturing the weblogin server allows the intruder to attack the authentication server.

Issues

Forcing the use of SSL was recognized to present issues to user agents that the end user must deal with. Some user agents will complain about wildcard certificates and the user must specifically accept the certificate. We have observed some vendor proprietary systems using HTTP protocols that have been coded never to accept wildcard certificates. Another issue arises when system administrators have image source tags that are absolute, rather than relative. When the image tag for example is http://www.foobar.net.com/image, the image will not always display properly. Because we have forced the user agent to use SSL, it should comment that it is about to fetch an insecure document, and give the user the option to proceed or not. Instead, some agents will silently ignore the image tag, resulting in an incomplete page being displayed to the external web client.

Web servers that run both secure (HTTPS port 443) and insecure (HTTP port 80) on the same server will cause problems. When the user connects to the server on port 80, and the proxy has established the session with HTTP keepalive active, and then the user selects a link on the returned page sending them to the HTTPS server on the same web server, the proxy will continue to proxy the request to port 80. It will receive an HTTP 404 response code and can recover from this error for any request types except posts. In these cases the proxy saves the original request and retransmits the saved request to port 443. However, due to the format nature and potential large size of post requests, the original post is not kept and the request will fail. It is not conceivable to save and reissue the post requests. In practice, however, this has not actually been a problem for us because our web server system administrators do not operate their servers in this manner.

As written, the proxy only supports standard HTTP ports 80 and 443. Other ports could be supported, but would require additional code. One way to address this would be to maintain a configuration file with a list of servers and associated non-standard ports.

For those who are so inclined, one can use self-signed certificates, as opposed to paying for certificates signed by recognized organizations. Doing so will generate a warning message on the browser, requiring the user to purposely accept the self-signed certificate.

Revocable authentication credentials is a valuable security tool. We limit the authentication credential lifetime. When the cookie expires, whoever is sitting at the browser has to login again. Inactive sessions are closed. If the proxy does not hear from the browser within the idle time out interval, the user must login again. Finally, the mindful user can explicitly logout or simply exit the browser; both remove the authentication cookie from the web client.

Even with all the built in security protections, it is still possible a session could be compromised. Good logs are critical for detection and recovery. The proxy logs all successful transactions with the source and destination IP addresses, the user's ID, and the HTTP request. The logs can be fed to a real time anomaly detection system and reviewed manually. Previous login successes, failures and logouts can be displayed after a successful weblogin so the user can report unexplained activity.

Security Analysis

Given the trusted nature of the web proxy, potential security risks are of particular concern. Below is an attempt to identify and show mitigation efforts for such concerns.

First, we must deal with network-based risks. Given that the web proxy is an inbound firewall service proxy, the following three network-based risks are salient [15]:

- Session hijacking
- Packet sniffing
- False authentication

Session hijacking is best mitigated by the use of SSL/TLS connections for all data movement beyond the initial referral from port 80. Since the proxy presents a properly (VeriSign) signed wildcard certificate, users can be relatively assured that a man-in-the-middle situation is not ongoing. As indicated in the SANS SSL Man-in-the-Middle Attacks paper [19], using older Internet Explorer browsers can make it easier to accomplish man-in-the-middle attacks.

Packet sniffing is also well protected against by the encryption provided by the SSL/TLS connections. Usernames, passwords, and subsequent authentication cookies are all contained within the encrypted channel. The proxy does continue to support shorter key lengths for older international browsers, mostly from a capability need. Currently, we feel the capability need outweighs of the risks of the reduced encryption quality, though such decision may be reversed in the near future.

Finally, false authentication risks are well mitigated by several aspects of the system. The use of the short-lived authentication cookies, the relationship of the cookie to originating IP address, and the random content of the cookie mitigate the risk of stolen or forged authentication cookies. The use of two-factor generated one-time passwords significantly lowers the risk of stolen passwords.

The transparent nature of the proxy may give some level of protection through its obscurity and difficulty in understanding the proxy's relationship to the web servers it sits in front. While the obscurity benefit may be minimal, it can definitely make scan probes of the internal network look unusual. One negative concern related to this transparent aspect is potential denial of service to the proxy, both intentional and unintentional. Since the server answers for a wide range of addresses (class B in our case) on port 80 and 443, there is significant potential for overwhelming the server. Experience has shown this can be mitigated through the use of SYN cookies [20] and keeping the port 80 redirector lean and simple since it receives the majority of denial attacks. Active methods of intrusion detection and automatic host blocking have also mitigated the impact of scans.

A second concern is logic and programming errors. These are the standard inherent low-level system risks relevant to any security-centric application. These risks include logic errors (e.g., unintended granting of authentication cookies) and programming errors (e.g., buffer overruns). The resulting unauthorized access could be either unintended access to the protected websites or administrative access to proxy itself. Obviously such risk must be addressed.

A multi-tiered mitigation approach to such errors has been taken. First, the proxy has undergone both peer design and code reviews to help remove both logic and programming errors. Formalized system configuration (we use a combination cfengine/cvs approach) including a minimized system install, helps to ensure a potential compromise has few other applications to leverage. Helping to ensure least privileged access for the proxy to the internal network, the router ACLs only allow the proxy to have TCP ports 80 and 443 access, again mitigating exposure should the proxy application or server be compromised. Minimizing vulnerabilities on internal web servers also reduces the risks of a compromised proxy. This mitigation is done with regular vulnerability scans of the internal network web servers. One should check the code using tools like Flawfinder and Rough Auditing Tool for Security (RATS). These tools will possibly identify security problems with the code. The proxy also performs off host system logging. To date, there have been no known compromises of the web proxy system.

The last aspect of security concern involves end-user behavior. User behavior can make or break any good security stance. A major security concern with the web proxy is user error. The proxy allows authenticated users access intranet services. The service is designed for users coming from public systems they do not control like kiosks, cyber cafes, or conference terminal rooms. Many users do not understand web cookie management or are forgetful. Walking away from a ''logged in'' browser is a real possibility and concern. The ability to revoke credentials is an important feature, as is their max age attribute of zero.

Conclusion

Users have received the proxy positively; with the LANL proxy handling over 10,000 authenticated HTTPS requests per day. The fact that the proxy is transparent, not requiring any browser preferences to be set, is particularly beneficial. This feature allows users to access LANL intranet web sites from anywhere they have access to a web browser. In addition, the per-user authentication has worked well. We have encountered no technical problems with the use of authentication cookies and no security issues with this mechanism have been detected. The proxy's simplicity and non-intrusive nature into the end data exchange has allowed it to work effectively with Java, Javascript, and all other web-based application extensions currently used and foreseeable. Overall, the proxy provides a highly flexible yet secure mechanism for Internet users to access intranet web content.

Author Information

James (Jim) Clifford is the Network Service Team Leader and a Systems Software Engineer for the Network Engineering Group at Los Alamos National Laboratory. His interests include Internet technology, Linux, and practical computer security. Jim has a BS from the University of Michigan. He may be reached

view e-mail: jrc@lanl.gov or smail mail: MS B255, Los Alamos National Laboratory, Los Alamos, NM 87545.

Alexander (Alex) Kent is the Deputy Group Leader for the Network Engineering Group at Los Alamos National Laboratory. His primary development projects include Laboratory-wide authentication and user account systems, network information propagation and architecture, and the Los Alamos firewall system. Alex has an BS and MS in CS from New Mexico Tech and an MBA from the University of New Mexico. He may be reached at <alex@lanl.gov>.

Roger Crandell is a Staff Member in the Network Engineering Group at Los Alamos National Laboratory. His primary role is firewall development and software engineering. Roger holds a BSEE from the University of Nebraska. He may be reached via email at rwc@lanl.gov.

References

[1] *Checkpoint FireWall-1*, http://www.checkpoint.com .

[2] *Cisco PIX firewall series*, http://www.cisco.com .

[3] *Secure Computing Sidewinder G2*, http//www.securecomputing.com .

[4] *Squid Web Proxy Cache*, http://www.squid-cache. org .

[5] *Netscreen 5000 series,* http://www.netscreen.com .

[6] Web Initial Sign-on (WebISO), http://www.middleware.internet2.edu/webiso .

[7] Thomas, Stephen, HTTP Essentials, 2001.

[8] Krishnamurthy, Balachander and Jennifer Rexford, *Web Protocols and Practice*, 2001.

[9] Kristol, David and Lou Montulli, ‘‘HTTP State Management Mechanisms,’’ *RFC 2965*, IETF, October, 2000.

[10] Baize, E. and D. Pinkas, ‘‘The Simple and Protected GSS-API Negotiation Mechanism,’’ *RFC 2478*, December, 1998.

[11] CRYPTOCard Corporation, Kanata, Ontario, Canada, http://www.CRYPTOCard.com .

[12] ‘‘Expired IETF Internet-Draft,’’ Art Luotonen, 1998.

[13] ‘‘ITU-T Recommendation X.509(03/00): Information Technology-Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks.’’

[14] Ziegler, Robert L., *Linux Firewalls*, 2001.

[15] Chapman, Brent D., and Elizabeth D. Zwicky, *Building Internet Firewalls*, 1995.

[16] http://www.globecom.net/ietf/draft/draft-brezak-spnego-http-03.html .

[17] http://www.pdos.lcs.mit.edu/papers/webauth%3Asec10.pdf .

[18] Marsh, Matthew G., *Policy Routing*, http://www.policyrouting.org/PolicyRoutingBook/ONLINE/TOC.html .

[19] http://www.sans.org/rr/papers/60/480.pdf .

[20] Lemon, Jonathan, ‘‘Resisting SYN flood DoS attacks with a SYN cache,’’ *Usenix BSDCon*, 2002.