2009

Intrusion Detection And Prevention System: CGI Attacks

Tejinder Aulakh
San Jose State University

INTRUSION DETECTION AND PREVENTION SYSTEM: CGI ATTACKS

A Thesis

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Tejinder Aulakh

December 2009

SAN JOSÉ STATE UNIVERSITY

The Undersigned Project Committee Approves the Project Titled

INTRUSION DETECTION AND PREVENTION SYSTEM: CGI ATTACKS

by
Tejinder Aulakh

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

Dr. Mark Stamp,	Department of Computer Science	Date
Dr. Robert Chun,	Department of Computer Science	Date
Ms. Sunitha Thummuri,	Cisco Systems	Date

APPROVED FOR THE UNIVERSITY

Associate Dean	Office of Graduate Studies and Research	Date

ABSTRACT

INTRUSION DETECTION AND PREVENTION SYSTEM: CGI ATTACKS

by Tejinder Aulakh

Over the past decade, the popularity of the Internet has been on the rise. The Internet is being used by its clients to access both static and dynamic data residing on remote servers. In the client-server interaction, the client asks the server to provide information, and, in addition, the server may also request that clients provide information such as in "web forms." Therefore, the Internet is being used for many different purposes which also include the web servers collecting the information from the clients. Consequently, attacks on the web servers have been increasing over the years. Due to the fact that web servers are now able to produce dynamic web pages based on the received requests, the web servers are now more vulnerable to attack than ever before.

One of the ways to produce the dynamic web page is Common Gateway Interface (CGI) technology. Attackers take the advantage of CGI scripts to perform an attack by sending illegitimate inputs to the web server. This report includes the findings and the results of the thorough research performed on the CGI-related web server attacks during the course of the project. In addition, this report contains a detailed explanation of the design and the implementation of the work done to develop an Intrusion Detection and Prevention System for CGI based web server attacks.

ACKNOWLEDGEMENTS

I would like to thank Dr. Mark Stamp for his continuous guidance and support throughout the period of this project. I would also like to thank the committee members, Dr. Robert Chun and Ms. Sunitha Thummuri, for being part of the committee and providing the valuable assistance and guidance in the completion of this project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

In the past, a web server has been mainly used to serve static HTML requests. A web client would send a static HTML request to the web server, and the web server would respond with a static HTML response (Syroid, 2002). Over the years, there has been a tremendous change in the way we use the Internet. Today, a typical Internet user has much more interaction with the Internet than in the past. The Internet is now being used for sharing pictures, social networking, stock trading, banking, and many other uses. There has been a great increase in the development of new Internet-based tools while taking advantage of the gigantic popularity of the Internet among the general population. Photo editing tools such as Picnik, Picasa Web, video sharing, and editing on YouTube are good examples of the recent Internet-based tools. These popular tools were made possible only because of the abilities of the technologies like Common Gateway Interface (CGI) and Server Side Includes (SSI) to produce dynamic content based on the received requests (Selamt, 2003).

However, this enhanced client-server interaction has made the web server highly vulnerable to attacks. CGI works as a communication gateway between the programs and the web server (Hollander, n.d.). The security risk with the CGI programs is that they can intentionally or unintentionally leak out confidential information to the attacker. In addition, an attacker can also modify the user input for CGI scripts in a way such that the CGI script will execute harmful commands on the server (Presis, n.d.).

There are several existing web server security solutions such as Firewalls to protect the systems from attacks, but they fail to prevent the attacks in most of the cases. For example, a Firewall does not always protect the server against the attacks because for a business to function properly behind a Firewall, some popular ports such as 80 and 21 on the Firewall will need to be left open, hence opening the doors for attacks (Huseby, 2005). An attacker can craft a legitimate HTTP request in such a way that it will look legitimate and perform the evil actions.

In this report, we will begin with providing the detailed information of the Common Gateway Interface Technology and its common uses. Then, we will talk about the security vulnerabilities that CGI technologies contain and well-known CGI attacks, including how they are performed. Next, we will analyze the existing tools that handle CGI related attacks and we will talk about their capabilities and limitations. Further, we will cover the background of the Intrusion Detection Systems, including their types and the components used in implementing them. Then, we will introduce an enhanced Intrusion Detection and Prevention System for CGI based attacks. We will also talk about the services that the proposed system will provide. In the next part, we will comprehensively explain the design and implementation of the proposed system. Finally, in the last part of the report, we will cover the outcomes of the project and make recommendations for the future work in this filed.

2 COMMON GATEWAY INTERFACE (CGI)

2.1 What is CGI?

Common Gateway Interface, commonly known as CGI, is a standard protocol used primarily by the web server to produce dynamic web pages. It is the CGI programs that have enabled the web servers to create the customized response with regard to the received request. With CGI, a web server can communicate with other parts or the programs running on the server to prepare the response. A CGI program calls other applications on the server and passes the user-specific information to the applications to prepare the response for the requested data or output (Gundavaram, 1996). After completing the operations, the CGI program returns the output to the web server and the web server then sends the response back to the client. The CGI programs are not language specific. They can be written in any language that provides standard input (STDIN) and standard output (STDOUT) (Marshall, 2002). The high level model of the CGI is illustrated in Figure 1.

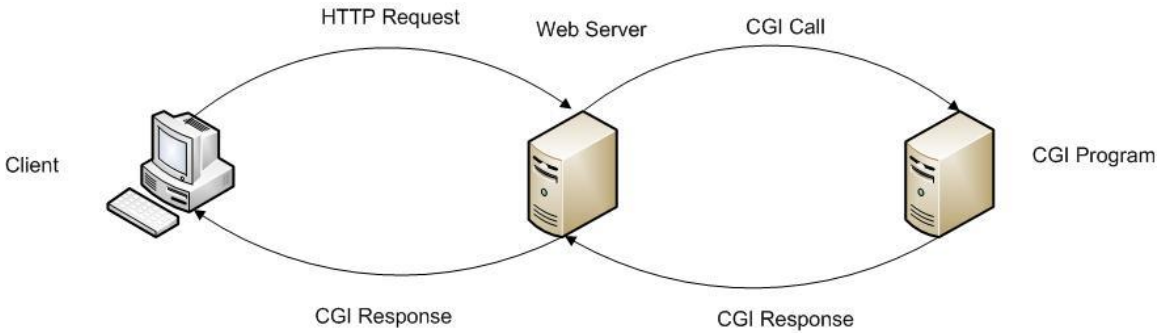


Figure 1: Common Gateway Interface

2.2 CGI Applications

CGI transforms the web server from a static server which primarily stores static web pages to an interactive server that clients can use to provide information to the server or access other applications running on the server. Let us discuss some of the popular applications that use CGI programs.

2.2.1 CGI Forms

CGI programs are most commonly used in the processing the input received from the forms. The forms can be used to gather information from the user or to enable the users run commands on the server. The forms can also be used to access other applications running on the server (Gundavaram, 1996). The CGI program then calls the appropriate application to process the input received the forms. The examples of the actions performed by the CGI include storing the information in the database, searching for documents, and running the Linux/Unix commands on the server etc. A CGI program is linked to the forms embedded in the web pages on the server by using the ACTION attribute of the HTML FORM tag. This is done as follows:

```
<FORM ACTION="/cgi-bin/CGI_handle_forms.pl" METHOD="POST">
```

Every time the user submits the form, the CGI program "CGI_handle_forms.pl" is called to process the input received from the user. A sample form is shown in Figure 2.

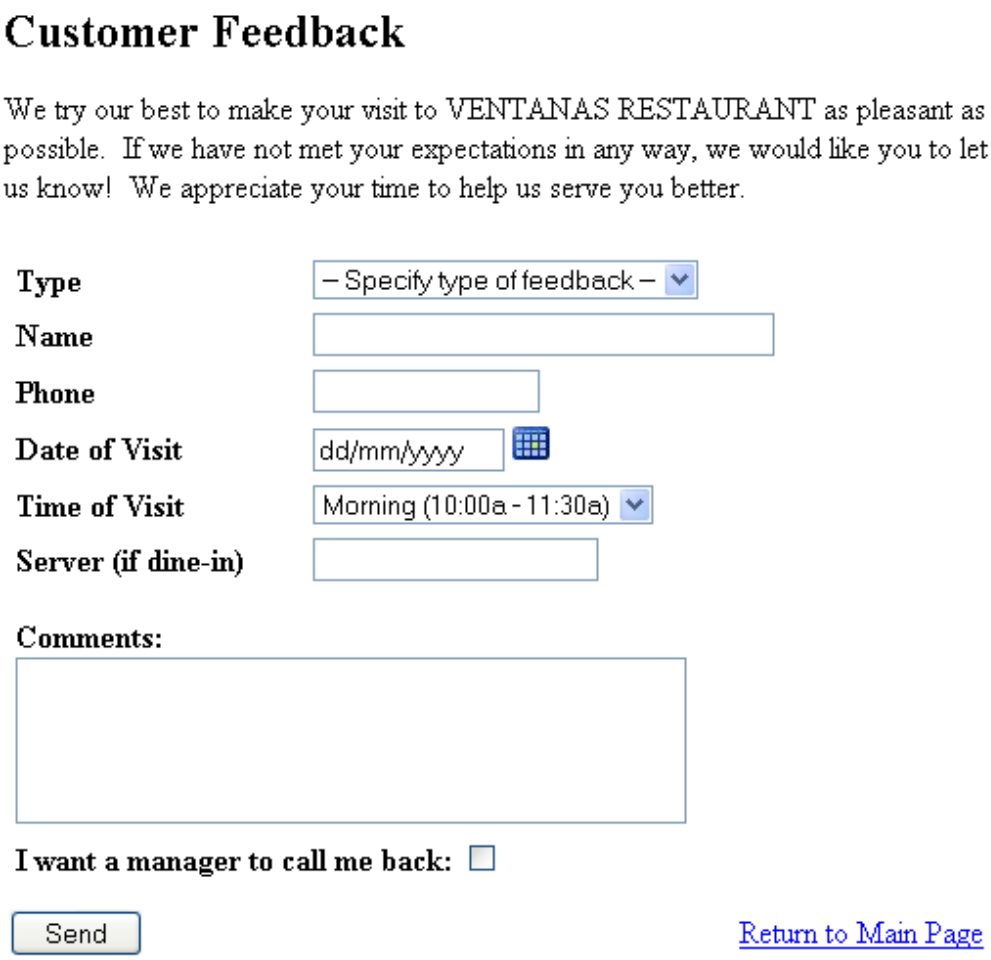


Figure 2: CGI Form

2.2.2 CGI Gateways

Making some crucial information or applications available on the web server for anyone to access is an incredibly useful feature but it generates immense security vulnerabilities. Therefore, direct access to such information or applications is not provided to the clients. Instead, all requests from the clients are passed through the Gateways before they reach the web server for processing. Gateways which use CGI programs perform application level authorization tasks determining if the client is

allowed to be given access to the requested information. If yes, Gateways use the available API's to access the information and prepare the response. The information requested could include file access and database access. Figure 3 includes the example of clients accessing the databases via the CGI Gateway.

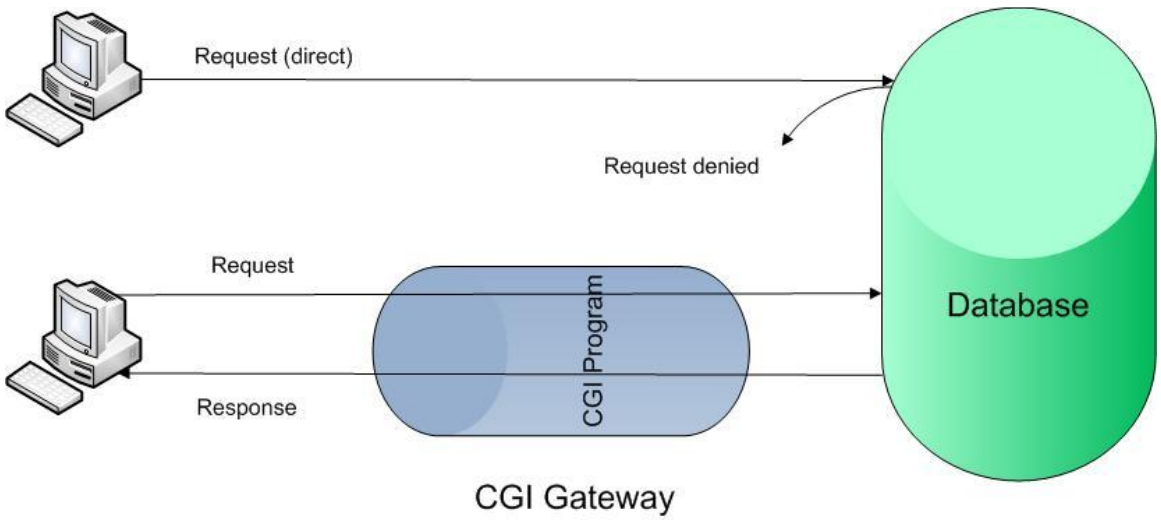


Figure 3: CGI Gateway

2.2.3 CGI Virtual Documents

CGI programs are also used in the creation of the Virtual Documents which are also known as Dynamic Documents. The content of the documents is dependent on the information received from the requests received from the server (Gundavaram, 1996). This is very useful in providing the customized information to the user. An example of a virtual document in presented below.

```
Welcome to Tejinder's WWW Server!
You are visiting from homes.com. The average load on my
server is 1.25.
Happy navigating!
```

Here, the CGI program was automatically able to detect where the client user is visiting from. The CGI program also keeps track of the average load on the server and displays it in the virtual documents. This is a very useful feature to be able to write extra information to the logs and gather other visitor related statistics to better serve the clients.

3 CGI SECURITY

Most of the web servers have a directory named “cgi-bin” which contains the CGI scripts. This directory contains some CGI scripts that present security vulnerability and make the server prone to many types of attacks. By exploiting these scripts, attackers can gain access to the server, modify the files, and retrieve confidential information which could be hidden in a non-public directory on the server. Figure 4 shows the intense interactivity of CGI with other applications such as the database, email, and OS commands. Attackers will try all possible ways to exploit the CGI scripts in order to reach these applications and have them produce the wanted output which is confidential information in most of the cases.

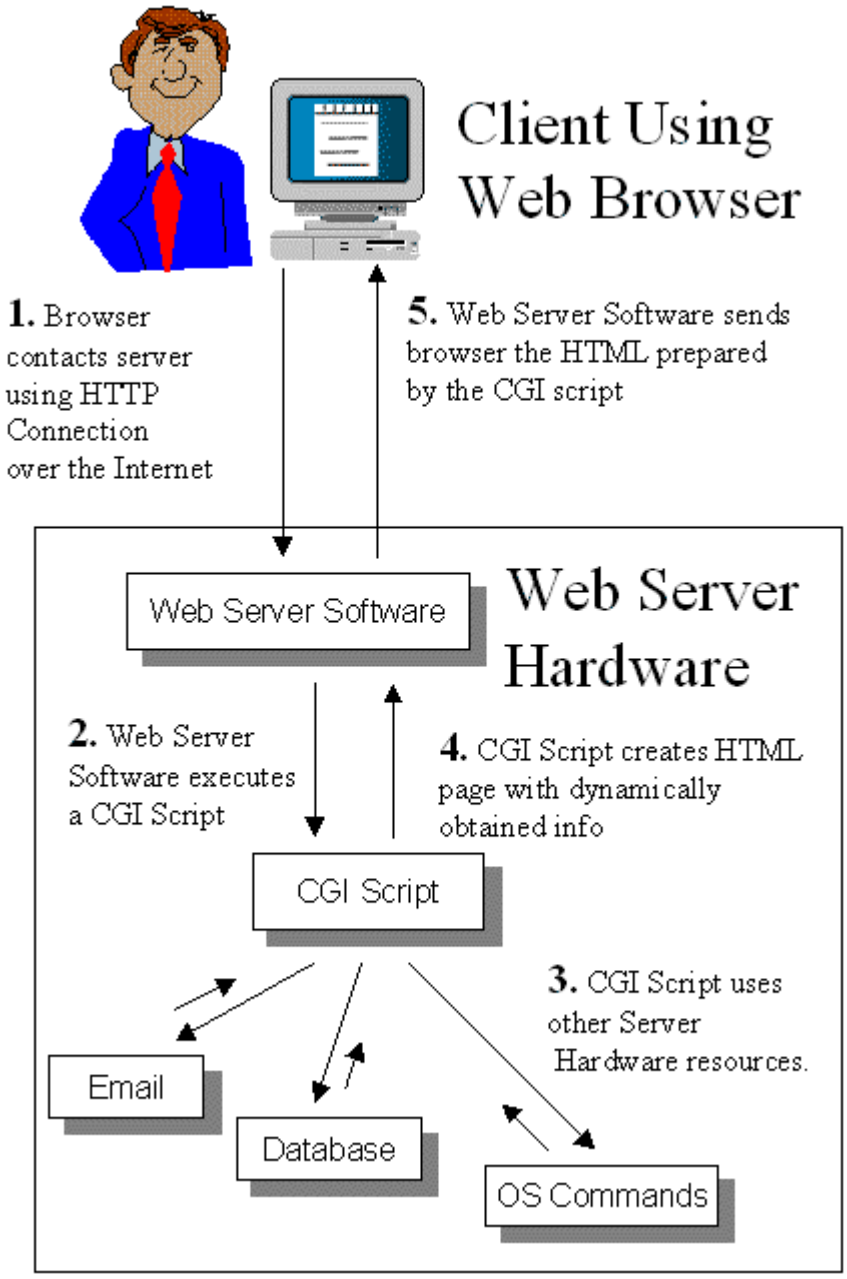


Figure 4: CGI Application Communication

Some of the well-known exploits of the CGI scripts are given below:

3.1 PHF Attack

A script named “phf” which is installed by default in the cgi-bin directory can be used to perform an attack on the web server. The legitimate use of the script is to update the people directory (The Hack FAQ, 2003). The script’s behavior changes if used with the “0a” character in the URL when calling the script. To perform an attack, the attacker appends “0a” to the URL along with some other UNIX command. The following are some of the examples of the attacks performed using the “phf” CGI script.

- Read the password file using “cat /etc/passwd” UNIX command
  http://victim.com/cgi-bin/phf?Qalias=x%0a/bin/cat%20/etc/passwd
- The output of the “id” command
  http://victim.com/cgi-bin/phf?%0aid==haqr==_phone=
- List the files in the home directory the user “john” using “ls –la ~john” command
  http://victim.com/cgi-bin/phf?%0als%20-la%20%7Ejohn==haqr==_phone=
- Copy the password file to another location
  http://victim.com/cgi-bin/phf?%0acp%20/etc/passwd%20%7Ejohn/passwd%0A==haqr==_phone=
- Remove the password file in the home directory of the user “john”
  http://victim.com/cgi-bin/phf?%0arm%20%7Ejohn/passwd==haqr==_phone=

3.2 Test Attack

Another CGI script named “test-cgi” is also included in most of the servers by default and can be found in the cgi-bin directory. This script is used in verifying the environment variables while processing the server requests. This script can be used by the attackers to print the server specific information such as the environment variables (The Hack FAQ, 2003). It can also be used along with the “0a” string to perform other operations. As an example, the attacker is able to get all the server related information by using the following URL.

http://victim.com/cgi-bin/test-cgi?\whatever

Table 1 contains the sample output produced by the test-cgi script.

Table 1: Test-cgi Output

```
I/1.0 test script report:

argc is 0. argv is .

SERVER_SOFTWARE = NCSA/1.4B
SERVER_NAME = something.com
GATEWAY_INTERFACE = CGI/1.1
SERVER_PROTOCOL = HTTP/1.0
SERVER_PORT = 80
REQUEST_METHOD = GET
HTTP_ACCEPT = text/plain, application/x-html,
application/html,
text/html, text/x-html
PATH_INFO =
PATH_TRANSLATED =
SCRIPT_NAME = /cgi-bin/test-cgi
QUERY_STRING = whatever
REMOTE_HOST = something.com
REMOTE_ADDR = 231.100.210.100
REMOTE_USER =
AUTH_TYPE =
CONTENT_TYPE =
CONTENT_LENGTH
```

3.3 JJ Attack

The script named “jj.c” in the “cgi-bin” directory is for the demonstration purposes. When this script is called, it gets the user input and passes it on to the ‘/bin/mail’ without scanning or filtering the input. This is a big advantage for the attackers since the input is not scanned and passed directly to the other programs. Therefore, an attacker can add “|” and a UNIX command in the request to perform an attack.

3.4 Compas Attack

NCSA server comes with a script named “compass.sh” which can also be exploited by the attackers to attack the server. This script does not scan the input from the user and can be used with the “0a” string to issue the UNIX commands for the server to run (The Hack FAQ, 2003). As an example, look at the following URL:

http://www.victim.com/cgi-bin/campas?%0acat%0a/etc/passwd%0

This URL would output the contents of the “/etc/passwd” file.

3.5 Count Attack

The “count.cgi” script is used to find the number of hits for the web pages. The script versions less than 2.4 contained exploits that can be used by the attackers to issue arbitrary UNIX commands at the same level as the httpd daemon (The Hack FAQ, 2003). As an example, an attacker can use the following URL to view the password file:

http://www.victim.com/cgi-bin/count.cgi?%0acat%0a/etc/passwd%0

Similarly, the attacker can use this script to issue other Unix commands on the server to possibly get unauthorized access or to bring the server down.

4 EXISTING TOOLS

There are several tools that deal with the CGI security vulnerability issues. The two popular tools used as the IDS for web based attacks are:

- CGI-IDS
- ModSecurity

The details of these two tools are described below.

4.1 CGI-IDS

This tool is a Perl based Intrusion Detection System and is based off PHP IDS. It scans the requests for any kind of patterns of the known attacks. It reads the attack patterns through an XML file which contains different kind of Filters (Altenburg, 2008). The following contains the structure of the CGI-IDS:

```
use CGI;
use CGI::IDS;
$cgi = new CGI;
# instantiate the IDS object;
# do not scan keys, values only; don't scan PHP code injection filters (IDs
58,59,60);
# All arguments are optional, 'my $ids = new CGI::IDS();' is also working
correctly,
# loading the entire shipped filter set and not scanning the keys.
# See new() for all possible arguments.
my $ids = new CGI::IDS(
whitelist_file  => '/home/hinnerk/ids/param_whitelist.xml',
disable_filters => [58,59,60],
);
# start detection
my %params = $cgi->Vars;
my $impact = $ids->detect_attacks( request => \%params );

if ($impact > 0) {
my_log( $ids->get_attacks() );
}
```

```
if ($impact > 30) {
my_warn_user();
my_email( $ids->get_attacks() );
}
if ($impact > 50) {
my_deactivate_user();
my_sms( $ids->get_attacks() );
}
# now with scanning the hash keys
$ids->set_scan_keys(scan_keys => 1);
$impact = $ids->detect_attacks( request => \%params ); (Altenburg, 2008)
```

The users of this IDS can expand the scope of this tool by adding more filters to the XML file and make the IDS detect additional attacks. A sample XML Filter file looks like the following:

```
<filters>
    <filter>
        <id>1</id>
        <rule><![CDATA[(?:"+.*>)|(?:[^\w\s]\s*\/>)|(?:>")]]></rule>
        <description>finds html breaking injections attacks</description>
        <tags>
            <tag>xss</tag>
            <tag>csrf</tag>
        </tags>
        <impact>4</impact>
    </filter>
</filters> (Altenburg, 2008)
```

The IDS also allows the users to set rules for which the requests are not gone through the intense analysis of the IDS. A White List file which is also an XML file and fully customizable allows the requests to pass through the IDS quickly. It plays a significant role in improving the speed of a busy server. A sample XML White List file is shown below:

```
<whitelist>
   <param>
      <key>scr_id</key>
      <rule><![CDATA[(?:^[0-9]+\.[0-9a-f]+$)]]></rule>
   </param>
   <param>
      <key>uid</key>
   </param>
</whitelist>
```
(Altenburg, 2008)

4.1.1 Disadvantages

This IDS lacks many features to make it a desirable IDS for the administrators to use. Below are some of the important features that this IDS does not provide.

- This tool does not have a user interface for the administrators of the systems to monitor the IDS activities and gather attack statistics. The console is a very crucial part of an IDS since via the console, the administrators of the systems are able to configure, upgrade, and add additional signature patters to the IDS. In this tool, the administrator would need to manually modify the XML files to add additional filters and White List entries.
- This tool does not use the databases to keep track of the attacks and the attacker's history. Databases are another important component of an IDS which lets the administrators to gather the statistics.
- This tool does not provide the capability to block the attacker for some limited amount of time or permanently. It is believed that this ability is important in some cases.

4.2 ModSecurity

ModSecurity is another application level IDS that operates while fully integrated into the web server to shield it from the application level attacks (ModSecurity for Apache User Guide, 2006). Some of the primary features of this IDS are described below.

- It performs filtering on the received requests which also includes looking for the strings that are commonly used to perform the attacks. Due to the fact this IDS is directly embedded into the web server, the filtering is done at the very low level of the application layer (ModSecurity for Apache User Guide, 2006).
- It provides any invasion capabilities.
- It has a good knowledge of the HTTP protocol. Therefore, it is able to perform specific filtering and also capable of analyzing the HTTP parameters.
- It performs analysis on the POST requests.
- It writes the details of the requests received to the logs for analysis by the administrators.
- It is able to perform HTTPS analysis because the IDS is implemented at the web server level. It is also capable of examining the data after the protocol decrypts the requests.

The configurations for this IDS are performed by editing the configuration file. A recommended configuration file by the ModSecurity team is show below:

```
# Turn ModSecurity On
SecFilterEngine On
# Reject requests with status 403
```

```
SecFilterDefaultAction "deny,log,status:403"
# Some sane defaults
SecFilterScanPOST On
SecFilterCheckURLEncoding On
SecFilterCheckUnicodeEncoding Off
# Accept almost all byte values
SecFilterForceByteRange 1 255
# Server masking is optional
# SecServerSignature "Microsoft-IIS/5.0"
SecUploadDir /tmp
SecUploadKeepFiles Off
# Only record the interesting stuff
SecAuditEngine RelevantOnly
SecAuditLog logs/audit_log
# You normally won't need debug logging
SecFilterDebugLevel 0
SecFilterDebugLog logs/modsec_debug_log
# Only accept request encodings we know how to handle
# we exclude GET requests from this because some (automated)
# clients supply "text/html" as Content-Type
SecFilterSelective REQUEST_METHOD "!^(GET|HEAD)$" chain
SecFilterSelective HTTP_Content-Type \
"!(^application/x-www-form-urlencoded$|^multipart/form-data;)"
# Do not accept GET or HEAD requests with bodies
SecFilterSelective REQUEST_METHOD "^(GET|HEAD)$" chain
SecFilterSelective HTTP_Content-Length "!^$"
# Require Content-Length to be provided with
# every POST request
SecFilterSelective REQUEST_METHOD "^POST$" chain
SecFilterSelective HTTP_Content-Length "^$"
# Don't accept transfer encodings we know we don't handle
SecFilterSelective HTTP_Transfer-Encoding "!^$" (ModSecurity for Apache
User Guide, 2006)
```

4.2.1 Disadvantages

Like CGI-IDS, this IDS also lacks many important features of a good IDS. First, this tool also does not provide a user-friendly console for the administrators to perform the configurations and upgrades. Second, it does not use databases to keep track of the

incidents. In addition, this IDS also lacks the capability to block the attacker for some limited amount of time or permanently.

5 INTRUSION DETECTION SYSTEM

5.1 Fundamentals

Intrusion Detection refers to the process of monitoring the system for unauthorized access incidents which can be the violation of the security policy, system use policy, or any other security standards (Scarfone, 2007). An Intrusion Detection System (IDS) is software that implements the intrusion detection process. On the other hand, an Intrusion Prevention System (IPS) prevents unauthorized access incidents from being successful. To better protect the system from any attacks, Intrusion Detection and Prevention System (IDPS) which provides a completely automated monitoring services, is deployed on the systems.

Most of the IDPS systems log the incident every time an attack on the system is detected and notifies the administration of the system so that all necessary actions can be taken to avoid such incidents again in the future. The administrators of the system can also configure the IDPS to monitor the violations of the end user policies and other unauthorized activities (Scarfone, 2007). As an example, if the user policy of a web server prohibits the users from uploading music files to the server, a properly configured IDPS would record the incidents when the users try to upload music files to the server. In some cases, IDPS are used to gather the attack statistics so that patterns of the attacks can be determined and also to better understand the attacks including how they are performed (Scarfone, 2007). Figure 5 illustrates where an IDPS is placed and how it functions.

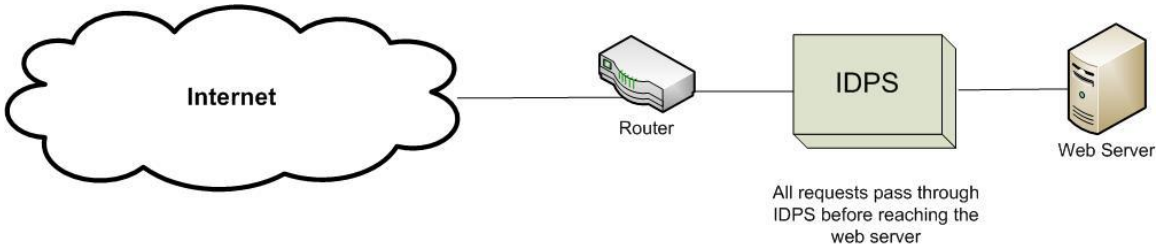


Figure 5: IDPS

5.2 IDPS Detection Models

The models that the IDPS follows to detect attacks can be divided into two categories:

- Signature-Based Detection Model
- Anomaly-Based Detection Model

In some cases, an IDPS is composed of more than a single detection model to provide the most level of security to the system (Scarfone, 2007). The two models are described below.

5.2.1 Signature-Based Detection Model

A signature refers to the pattern in which a previously known attack was performed. The signature-based detection methodology is the process of comparing the current events with the signatures (Scarfone, 2007). Below are some of the examples of the signatures:

- Log in attempt as the “root” which refers to the violation of the security policy of the system
- Emails with the subject “Free Screensavers” and containing an attachment “screensaver.exe” which refers to the spam emails
- The operating system log entry code 645 which refers to disabling of the server’s auditing

This type of IDPS model is a very efficient detection model due to its low complexity in implementation and detection. It simply compares the current activity to the stored signatures to find any pattern similarities in order to detect the attacks. In addition, the signature-based detection model produces very specific attack event reports as oppose to the anomaly-based detection model which we will analyze in the next section (Stamp, 2009). The drawback of a signature-based detection model is its inability to detect new unknown attacks since the system does not have any signature entry in the system for the new attacks.

5.2.2 Anomaly-Based Detection Model

The anomaly-based detection model detects the attacks based on the profiles. The profiles contain the patters or the normal behavior in which the system is being used. The profiles are based on specific users, networks, or the applications. They are created by monitoring the system use over a period of time, known as the evaluation period. This model compares the current activities with the profiles to discover the abnormal activity in progress which in most cases is an attack (Scarfone, 2007). Since the system use and

the network use are not static and always contain some variation over time, the profile must also adjust accordingly. Therefore, after the creation of the profiles in the evaluation period, an IDPS changes the profiles over time. The examples of the profiles are mentioned below.

- A user profile contains an email activity of 5%. When the IDPS using anomaly-based detection model senses that the email activity on the system is more than 5%, it will consider it an attack.
- Over the past few weeks, on average a user performs open, read, and write operations on the file system for 2% of the time. When the IDPS detects a sudden increase in the file access operations, it reports an attack incident.

The advantage of the anomaly-based detection model is that it is able to detect even unknown attacks by comparing the current abnormal events with something that is considered normal. Further, this model can also be more efficient than the signature-based model given that there are a huge number of signatures to compare with in the signature-based detection model (Stamp, 2009). On the other hand, the attack incidents that anomaly-based model produces are not very specific and it takes some efforts by the administrator to pin-point the root of the attack (Stamp, 2009). In addition, this model is subject to a "slow attack." In this type of attack, the attacker first finds out the threshold between the normal and abnormal activities. The attacker then would slowly attack the

system making sure that the activities during the attack do not reach the threshold which results into the anomaly-based detection model not detect the attack.

5.3 IDPS Components

An IDPS is composed of the following components.

- **Agent**

  Agent is the module that listens for the events and analyzes the system activities. In case of IDPS used for network attacks, the agent is called "sensor" (Scarfone, 2007).

- **Management Server**

  Management server is responsible for analyzing the received information from the current activity to decide if an attack is in progress. It would use the information from other elements also such as signatures and profiles to complete the analysis.

- **Database Server**

  Database server is used to store the information received from the agents and the management server. The management server also uses the database server to complete it operations.

- **Console**

  Console or management interface serves as the interface between the IDPS and the administrator. The console is used to monitor the system events produced by the IDPS. Some consoles are also used to configure the agents and perform software upgrades (Scarfone, 2007).

Figure 6 illustrates all the required components that an IDS is composed of.

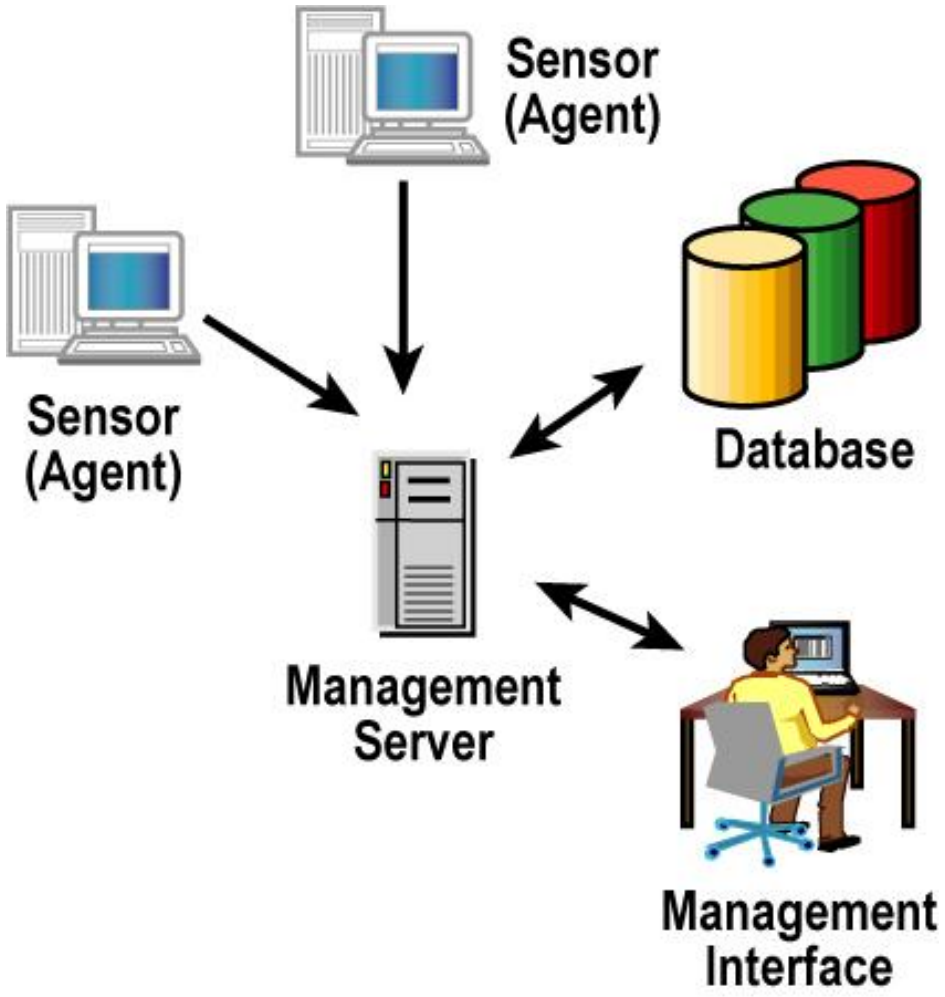


Figure 6: IDPS Components

6 DEVELOPED IDPS

The system designed and implemented during this project is a complete IDPS which uses a signature-based detection model. The system contains a list of signatures that are used to detect the attacks. The signatures are well-known CGI patterns that have been used in the past to attack the systems. The system contains the following crucial components to provide a robust web server security against the CGI attacks.

6.1 Graphical User Interface (GUI)

The system provides a user-friendly interface for the server administrator to monitor the attacks. Through the user interface, the administrator can view past CGI attacks, block attacker IP addresses, unblock IP addresses etc. The GUI contains the following interfaces:

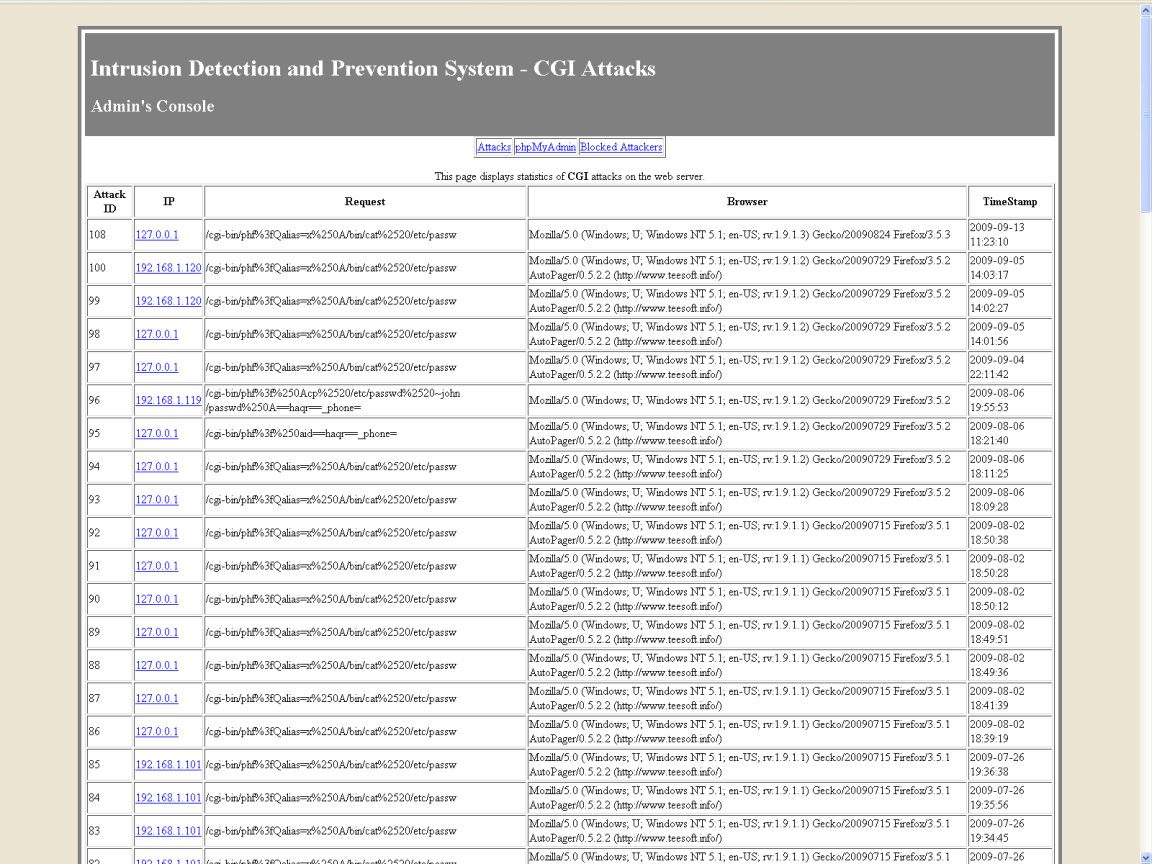
6.1.1 CGI Attacks Interface

The interface named “CGI Attacks” lists all past CGI attacks detected by the system with the most recent attacks listed on the top. Table 2 contains the fields and their description that are shown on this interface.

Table 2: CGI Attacks Interface Fields

Field	**Description**
Attack ID	ID of the attack
IP	Attacker's IP address linked to "Attacker History" interface
Request	Attacker's browser that was used in the attack
Browser	Browser used to perform
Timestamp	Timestamp at which the attack was detected.

The IP address entries of the attacker are clickable and linked to the "Attacker History" page displaying the attack history from the clicked IP address. In addition, the interface also contains the links to the "Blocked Attackers" page and "phpMyAdmin". Figure 7 contains the screenshot of "CGI Attacks" interface.



Intrusion Detection and Prevention System - CGI Attacks

Admin's Console

Attacks | phpMyAdmin | Blocked Attackers

This page displays statistics of **CGI** attacks on the web server.

Attack ID	IP	Request	Browser	TimeStamp
108	127.0.0.1	/cgi-bin/phf%3fQalias=x%250A/bin/cat%2520/etc/passw	Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.3) Gecko/20090824 Firefox/3.5.3	2009-09-13 11:23:10
100	192.168.1.120	/cgi-bin/phf%3fQalias=x%250A/bin/cat%2520/etc/passw	Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.2) Gecko/20090729 Firefox/3.5.2 AutoPager/0.5.2.2 (http://www.teesoft.info/)	2009-09-05 14:03:17
99	192.168.1.120	/cgi-bin/phf%3fQalias=x%250A/bin/cat%2520/etc/passw	Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.2) Gecko/20090729 Firefox/3.5.2 AutoPager/0.5.2.2 (http://www.teesoft.info/)	2009-09-05 14:02:27
98	127.0.0.1	/cgi-bin/phf%3fQalias=x%250A/bin/cat%2520/etc/passw	Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.2) Gecko/20090729 Firefox/3.5.2 AutoPager/0.5.2.2 (http://www.teesoft.info/)	2009-09-05 14:01:56
97	127.0.0.1	/cgi-bin/phf%3fQalias=x%250A/bin/cat%2520/etc/passw	Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.2) Gecko/20090729 Firefox/3.5.2 AutoPager/0.5.2.2 (http://www.teesoft.info/)	2009-09-04 22:11:42
96	192.168.1.119	/cgi-bin/phf%3f%250Acp%2520/etc/passwd%2520~john /passwd%250A==haqr==_phone=	Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.2) Gecko/20090729 Firefox/3.5.2	2009-08-06 19:55:53
95	127.0.0.1	/cgi-bin/phf%3f%250aid==haqr==_phone=	Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.2) Gecko/20090729 Firefox/3.5.2 AutoPager/0.5.2.2 (http://www.teesoft.info/)	2009-08-06 18:21:40
94	127.0.0.1	/cgi-bin/phf%3fQalias=x%250A/bin/cat%2520/etc/passw	Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.2) Gecko/20090729 Firefox/3.5.2 AutoPager/0.5.2.2 (http://www.teesoft.info/)	2009-08-06 18:11:25
93	127.0.0.1	/cgi-bin/phf%3fQalias=x%250A/bin/cat%2520/etc/passw	Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.2) Gecko/20090729 Firefox/3.5.2 AutoPager/0.5.2.2 (http://www.teesoft.info/)	2009-08-06 18:09:28
92	127.0.0.1	/cgi-bin/phf%3fQalias=x%250A/bin/cat%2520/etc/passw	Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.1) Gecko/20090715 Firefox/3.5.1 AutoPager/0.5.2.2 (http://www.teesoft.info/)	2009-08-02 18:50:38
91	127.0.0.1	/cgi-bin/phf%3fQalias=x%250A/bin/cat%2520/etc/passw	Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.1) Gecko/20090715 Firefox/3.5.1 AutoPager/0.5.2.2 (http://www.teesoft.info/)	2009-08-02 18:50:28
90	127.0.0.1	/cgi-bin/phf%3fQalias=x%250A/bin/cat%2520/etc/passw	Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.1) Gecko/20090715 Firefox/3.5.1 AutoPager/0.5.2.2 (http://www.teesoft.info/)	2009-08-02 18:50:12
89	127.0.0.1	/cgi-bin/phf%3fQalias=x%250A/bin/cat%2520/etc/passw	Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.1) Gecko/20090715 Firefox/3.5.1 AutoPager/0.5.2.2 (http://www.teesoft.info/)	2009-08-02 18:49:51
88	127.0.0.1	/cgi-bin/phf%3fQalias=x%250A/bin/cat%2520/etc/passw	Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.1) Gecko/20090715 Firefox/3.5.1 AutoPager/0.5.2.2 (http://www.teesoft.info/)	2009-08-02 18:49:36
87	127.0.0.1	/cgi-bin/phf%3fQalias=x%250A/bin/cat%2520/etc/passw	Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.1) Gecko/20090715 Firefox/3.5.1 AutoPager/0.5.2.2 (http://www.teesoft.info/)	2009-08-02 18:41:39
86	127.0.0.1	/cgi-bin/phf%3fQalias=x%250A/bin/cat%2520/etc/passw	Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.1) Gecko/20090715 Firefox/3.5.1 AutoPager/0.5.2.2 (http://www.teesoft.info/)	2009-08-02 18:39:19
85	192.168.1.101	/cgi-bin/phf%3fQalias=x%250A/bin/cat%2520/etc/passw	Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.1) Gecko/20090715 Firefox/3.5.1 AutoPager/0.5.2.2 (http://www.teesoft.info/)	2009-07-26 19:36:38
84	192.168.1.101	/cgi-bin/phf%3fQalias=x%250A/bin/cat%2520/etc/passw	Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.1) Gecko/20090715 Firefox/3.5.1 AutoPager/0.5.2.2 (http://www.teesoft.info/)	2009-07-26 19:35:56
83	192.168.1.101	/cgi-bin/phf%3fQalias=x%250A/bin/cat%2520/etc/passw	Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.1) Gecko/20090715 Firefox/3.5.1 AutoPager/0.5.2.2 (http://www.teesoft.info/)	2009-07-26 19:34:45
82	192.168.1.101	/cgi-bin/phf%3fQalias=x%250A/bin/cat%2520/etc/passw	Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.1) Gecko/20090715 Firefox/3.5.1	2009-07-26

Figure 7: CGI Attacks Interface

6.1.2 Attacker History Interface

The interface “Attacker History” contains attacker history of the attacker whose IP address was clicked on the “CGI Attacks” interface. In addition to the attacks table and the links, the interface also displays additional fields which are shown in the Table 3 below.

Table 3: Attacker History Interface Fields

Field	**Description**
Attacker IP	Attacker's IP address
Total Attacks	Total number of attacks performed by this attacker
Action	"Block Attacker" button to block this IP address from accessing the server again

"Block Attacker" button calls another script which blocks the IP address and takes the administrator to the confirmation page displaying the success of the operation. It also gives the option to undo the operation which means unblocking the IP address which was just blocked. Figure 8 contains the screen shot of this interface.

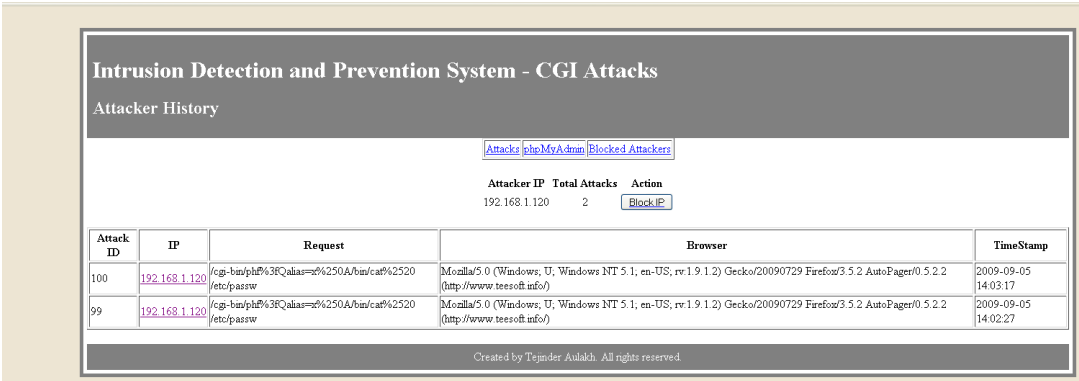


Figure 8: Attacker History Interface

6.1.3 Blocked Attackers Interface

This interface displays the IP address of all the attackers that have been banned from accessing the server. This interface contains links to other interfaces and also a table of banned IP addresses. The fields that are shown on this interface are explained in Table 4.

Table 4: Blocked Attackers Interface Fields

Field	Description
Action	Unblock action which contains the link to the script which performs the unblock operation
IP	IP addressed that has been banned
Timestamp	Timestamp at which the IP address was banned

Clicking on the unblock action calls another script which unblocks the IP address and takes the administrator to the confirmation page displaying the success of the operation. Figure 9 contains the screen shot of this interface.

Figure 9: Blocked Attackers Interface

6.2 MySQL Database

The MySQL database keeps track of all the attacks and stores the necessary information for analysis by the administrators later on. The database is also used for blocking and unblocking the IP addresses.

6.3 Management Module

The management module is a fundamental component of the implemented system. It is responsible for detecting the attack by checking the current activity pattern against

the signatures. It also provides access to the database and stores the attack information in the database.

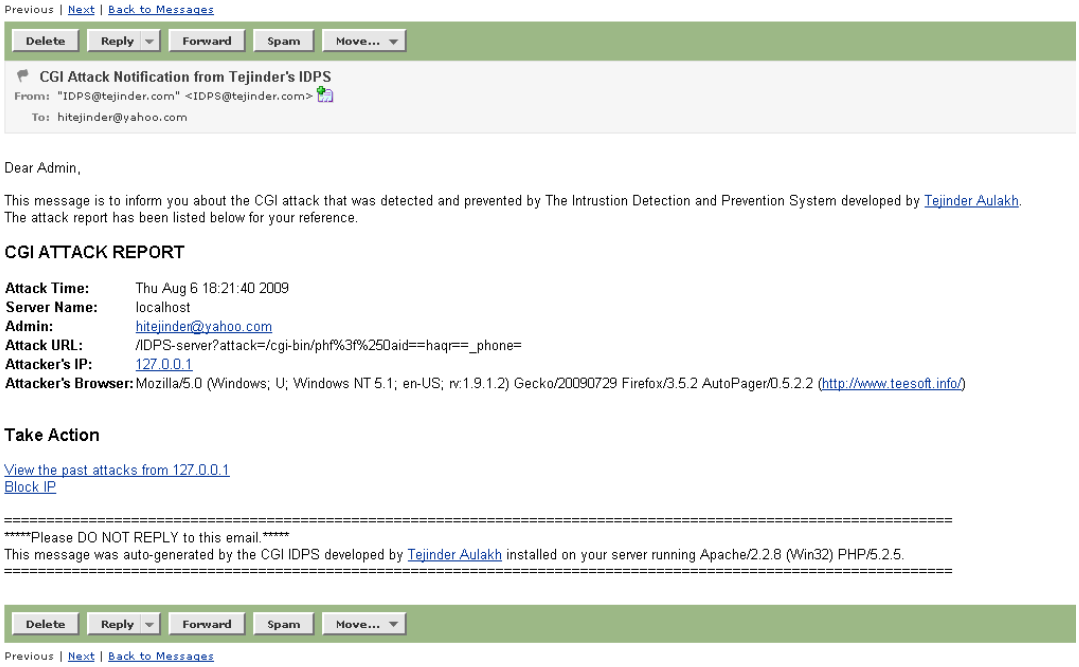
6.4 Email Generation

Every time an attack event is detected, an email is generated and sent to the administrator of the system. This is done to ensure that the administrator is always kept informed about the attacks. If any additional actions are needed, the administrator can perform them after receiving the attack email and being informed about the CGI attack. The email contains the following information that is explained in Table 5.

Table 5: Email Message Fields

Field	**Description**
Attack Time	The timestamp for the attack
Server Name	The hostname of the server at which the attack was aimed
Admin	The administrator's email address
Attack URL	The signature of the URL used in the attack
Attacker's Browser	The web browser used in performing the attack

The screenshot of the email message is shown in Figure 10.


Previous | Next | Back to Messages

Delete | Reply | Forward | Spam | Move...

CGI Attack Notification from Tejinder's IDPS

From: "IDPS@tejinder.com" <IDPS@tejinder.com>

To: hitejinder@yahoo.com

Dear Admin,

This message is to inform you about the CGI attack that was detected and prevented by The Intrustion Detection and Prevention System developed by Tejinder Aulakh.
The attack report has been listed below for your reference.

CGI ATTACK REPORT

Attack Time: Thu Aug 6 18:21:40 2009
Server Name: localhost
Admin: hitejinder@yahoo.com
Attack URL: /IDPS-server?attack=/cgi-bin/phf%3f%250aid==haqr==_phone=
Attacker's IP: 127.0.0.1
Attacker's Browser: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.2) Gecko/20090729 Firefox/3.5.2 AutoPager/0.5.2.2 (http://www.teesoft.info/)

Take Action

View the past attacks from 127.0.0.1
Block IP

==========
*****Please DO NOT REPLY to this email.*****
This message was auto-generated by the CGI IDPS developed by Tejinder Aulakh installed on your server running Apache/2.2.8 (Win32) PHP/5.2.5.
==========

Delete | Reply | Forward | Spam | Move...

Previous | Next | Back to Messages

Figure 10: CGI Attack Email Notification

7 DESIGN AND IMPLEMENTATION

The system was implemented using various languages such as PHP, Perl, and HTML. For database connectivity SQL query language was used.

7.1 CGI Attack Detection

Apache config file httpd.config was modified to allow the execution of Perl scripts by the server. In addition, it was also modified for the detection of the CGI attacks. Using the ReWriteCond the system scans the request for the attack patters. If the patter is found in the request, the system uses the RewriteRule to forward the whole request to a Perl script called “IDPS-server” which will use the request to extract the required information.

7.2 Storing Attacks

A MySQL database is used to store the attack information to be reviewed by the administrator. The database contains a table named “attacks” in which the attack information is stored. The module “IDPS-server” is responsible for storing the attack information into the database. IDPS-server module uses MySQL API’s for Perl to interact with the database. When an attack event is detected, it stores the following information in the database:

- The URL or the request used to perform the attack.
- Attacker’s browser that was used in the attack.
- Timestamp at which the attack was detected.

7.3 Blocking Attackers

A table named “blocked_hackers” is used to store the IP addresses of the attackers that have been marked “blocked” by the administrator to ban the access to the server. When the server receives the request, the first thing that is checked is the IP address of the client to see if it is marked as “blocked” in the database. If it is marked “blocked”, the request is ignored by the server.

7.4 Email Notifications

After storing the necessary information in the database, the module “IDPS-server” then sends the attack information to the administrator of the server via an email message. A procedure named send_mail() was written to send the email notifications to the administrator. The procedure uses the Perl API “Mail::Sendmail” to talk to the mail server and send the emails.

8 IDPS REQUEST HANDLING FLOW CHART

When the web server receives the CGI request from the client, it is forwarded to the developed IDPS. After the IDPS receives the forwarded request, it checks if the client is marked as block in the databases. If it is, then the access to the server is denied. Otherwise, the IDPS moves on to check if the request contains any attack signatures to detect for a possible CGI attack in the process. If no CGI attack signatures are found in the request, then the request is given back to the web server for normal request processing. Otherwise, the request is handed over to the IDPS attack handler module to store the attack information, notify the administrator and finally deny the access to the server for the request. The Attack Detection Flow Chart diagram is shown in Figure 11.

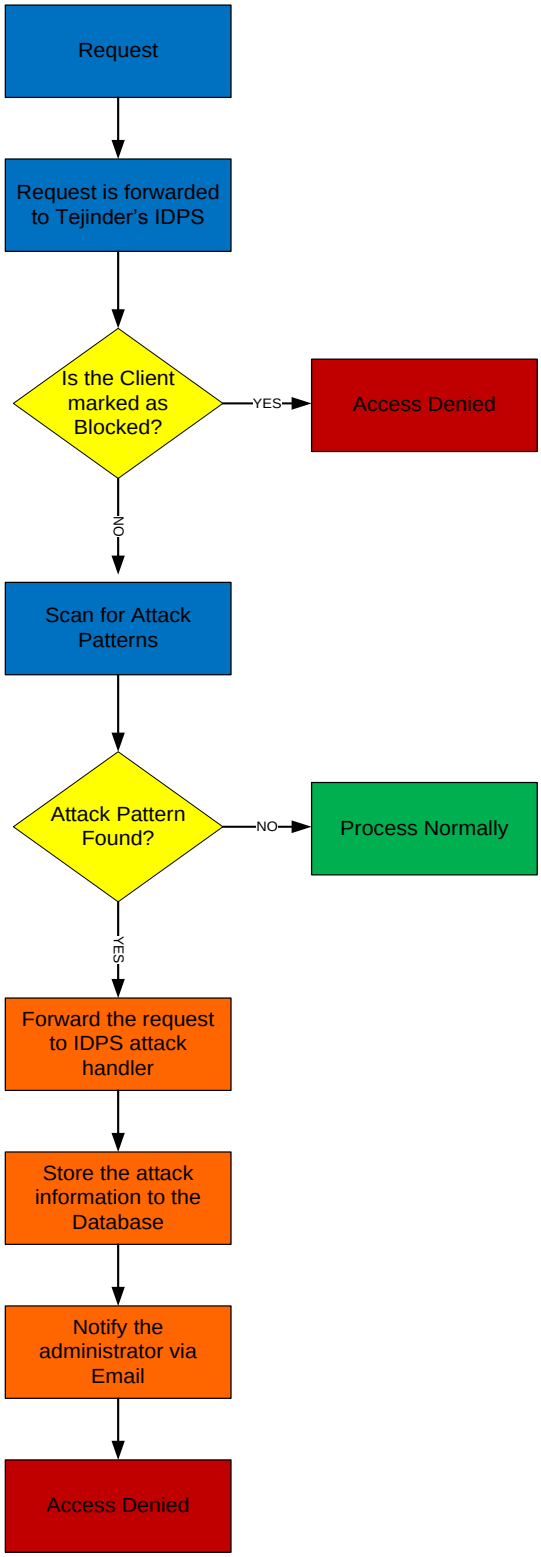


Figure 11: Attack Detection Flow Chart

9 RESULTS AND OUTCOMES

The developed system was tested by sending several bad CGI requests to perform the attacks. The system successfully prevented the attacks and sent the email notifications to the server administrator. The system was also tested the ability to block and unblock the attackers.

9.1 Performing the Attack

When sending an attack URL to the server, the web server is successfully able to prevent the attack, log the attack entry in the database, and notified the administrator of the server about the attack. Figure 12 shows the screenshot of the GUI displaying the confirmation of the attack detection and prevention.

Figure 12: Attack Prevention Output

10 FUTURE WORK

The developed system can be further enhanced adding the capability to protect the server from other types of attacks such as:

- **PHP related attacks**

  The support for PHP related attacks can be added to this IDPS.

- **SQL Injection**

  Another module can be developed and added to the IDPS which will prevent SQL Injection attacks by scanning the input entered in the forms.

- **An attempt to access private documents**

  This IDPS can be further modified to track and block the attempts to access the private documents. The administrator can choose which documents should be considered private.

- **Option to block attacker for limited time**

  If useful, the administrator can be given the option to block the attacker for some limited period of time such as an hour or so. To add this functionality the cron jobs will have to be regularly scheduled on the web server to unblock the attackers if the time duration for which the attacker was blocked has expired.

11 CONCLUSION

In this paper, I presented detailed information of the CGI technologies. I discussed different applications in which CGI technology is being used. I then stated numerous kinds of security vulnerability that the use of CGI technology presents. I demonstrated that the attackers perform the attack on the web server by crafting illegitimate HTTP requests into legitimate HTTP requests so that the web server would process the requests without detecting the hidden harms. By doing so, the attackers are able to perform prohibited actions on the server which includes getting unauthorized access to the web server, bringing the web server down, and making it unable to serve the genuine clients.

In addition, I analyzed two commercial tools that are being used to protect the web servers from the attacks. I analyzed CGI-IDS and ModSecurity in details and concluded that neither of the tools are user-friendly in performing configurations and monitoring attacks. I mentioned that these two tools lack major IDS components such as the management console and the database to keep track of the past attacks. I provided evidence that there is a need for another tool which should consist of all significant IDS components and better protect the web servers from CGI-related web attacks.

I discussed the design and implementation of the new tool that was developed for this project. I demonstrated that the tool provides a robust web-server security against CGI-based attacks. The tool contains crucial IDPS components and provides capabilities that other tools failed to provide. The features include IDPS management configuration interface, ability to block the attackers, and automatic email notification.

REFERENCES

Altenburg, H. (2008). CGI-IDS. *CPAN.* Retrieved March 22, 2009, from http://search.cpan.org/~hinnerk/CGI-IDS/lib/CGI/IDS.pm

Gundavaram, S. (1996, March). CGI programming on the world wide web. *O'reilly Online Catalog.* Retrieved March 29, 2009, from http://oreilly.com/openbook/cgi/

Hollander, Y. (n.d.). The future of web server security. *Entercept Security Technologies.* Retrieved March 4, 2009, from http://www.cgisecurity.com/lib/wpfuture.pdf

Huseby, S. (2005, June 1). Common security problems in the code of dynamic web applications. *Web Application Security Consortium.* Retrieved February 3, 2009, from http://www.webappsec.org/projects/articles/062105.shtml

Intrusion detection system. (2007, August 17). *Hill Associates.* Retrieved March 25, 2009, from http://www.hill2dot0.com/wiki/index.php?title=IDS

Kazienko, P., & Dorosz, P. (2004, June 14). Intrusion detection systems. *Windows Security.* Retrieved March 22, 2009, from http://www.windowsecurity.com/articles/Intrusion_Detection_Systems_IDS_Part_I__network_intrusions_attack_symptoms_IDS_tasks_and_IDS_architecture.html

Marshall, J. (2002, April 12). Writing CGI scripts to process web forms. *CGI Made Really Easy.* Retrieved March 25, 2009, from http://www.jmarshall.com/easy/cgi/

ModSecurity for apache user guide. (2006, April 10). *ModSecurity.* Retrieved March 28, 2009, from http://www.modsecurity.org/documentation/modsecurity-apache/1.9.3/html-multipage/index.html

Presis, B. (n.d.). On the security of CGI scripts. *Website of Bruno R. Preiss.* Retrieved March 10, 2009, from http://www.brpreiss.com/talks/1996/padsgroup/slides.pdf

Scarfone, K. (2007). Guide to intrusion detection and prevention systems. *National Institute of Standards and Technology.* Retrieved March 13, 2009, from http://csrc.nist.gov/publications/nistpubs/800-94/SP800-94.pdf

Selamt, S. (2003). Web server scanner: scanning on IIS, CGI and HTTP. *IEEE.* Retrieved May 23, 2009, from http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1274232

Sole, S. (n.d.). Server-side scripting. *Web Developer's Virtual Library.* Retrieved March 20, 2009, from http://www.wdvl.com/Authoring/Scripting/WebWare/Server/

Stamp, M. (2009). *Information security: principles and practice*. San Jose, CA: Wiley-Interscience.

Stanley, N. (2008, January 4). Intruder alert. *Server Management.* Retrieved April 2, 2009, from http://www.server-management.co.uk/features/643

Syroid, T. (2002, September 1). Web server security. *IBM.* Retrieved February 22, 2009, from http://www.ibm.com/developerworks/linux/library/s-wssec.html

The hack FAQ. (2003, July 25). *Nomad Mobile Research Centre.* Retrieved March 20, 2009, from http://www.nmrc.org/pub/faq/hackfaq/hackfaq-09.html