Global Information Assurance Certification Paper

Secure Execution of Insecure CGI Scripts

SANS GSEC Practical Version 1.4b

By Paul Bryan

05 August 2003

Abstract

The purpose of this project is to provide students with access to a web server which allows them to run Common Gateway Interface (CGI)[1] scripts in a secure manner. Careful application of password protection, "suexec"[2], firewalls, system maintenance/monitoring, and policies leads to a more secure system. These measures also allow for auditing of the system.

The system has benefited with greater security, but there are still potential vulnerabilities. Therefore, it must be monitored constantly and all software kept up to date. Use of a “chroot”[3] jail and a secure version of the Linux kernel could add additional security to the system.

Before

This system will be used by both staff and students. In this case, students must be considered as untrusted users. This poses a number of security risks, especially considering that the administrators of this box will have no control over the scripts that students write. Further to that, as the students are learning CGI scripting, the may not be aware of the security implications of writing CGI scripts.

CGI scripting has been the cause of a number of security holes over the years. There are a wide range of industry best practice documents, outlining ways write safe, secure CGI scripts[4]. The challenge in this case however, is to build a system that is secure regardless of whether the CGI scripts themselves are secure. Realistically, it is unlikely that this system can be totally secured, but improved security, auditing and accountability are achievable goals.

Currently, the only place to provide this service to students is on a server that staff use for publishing their own web pages. The server is running RedHat GNU/Linux 7.3[5] with Apache 1.3[6]. It is set up so that CGI scripts must be placed in a global scripting directory. The scripts are executed with the permissions of the user Apache is running under. This is common for all CGI scripts on this server.

Students log into the system via ssh which uses an organization wide LDAP[7] authentication service in the back end. There is no protection mechanism for accessing user web pages via HTTP[8].

Currently, a security policy is being developed - and more specifically a password policy - for the LDAP authentication service. This document will not delve into the details of such policies, except to state that industry best practices such as password expiration, account lock-out and complex passwords will be included in the policy. This will have an impact on the security of user accounts as they are protected by a username and password.

The potential security problems can be divided into three broad categories. This makes it possible to provide a risk analysis of each category and to see how these risks can be mediated.

The first is that students themselves may write malicious scripts. These might attack the host they are running on, other systems connected to the network, or any other systems connected to the Internet.

The second category pertains to poorly written scripts. An attacker may be able to exploit security vulnerabilities in the students' scripts to their own ends.

Thirdly, students may misuse the web server to conduct illegal activities. This might include things like copyright violation (e.g. distributing copyrighted material illegally).

The threat of students writing malicious scripts is probably a medium threat. Most students will not want to jeopardize their results by launching attacks from this server. However, students often try to “test the boundaries” so there is certainly still a risk.

The vulnerability associated with this threat is medium to high. Scripts on the server execute with minimum privileges, but local exploits may allow a student to obtain higher privileges. Some attacks may not even require high privileges (e.g. Deleting files owned by Apache).

The threat of students writing scripts with security holes in them is quite high. As they will be learning CGI scripting, students will not always be aware of the security considerations required when writing scripts. Hopefully, by the end of the course the will have a good idea though!

This threat, perhaps counter-intuitively, quite possibly has a higher vulnerability than student's writing malicious scripts. This is because, the server is not advertised and the name of the directories changes each semester with new students. It is not likely to be publicly indexed on any search engines (although a student could post links to the server). Attackers looking for vulnerable CGI scripts are not likely to find student scripts on this server because it is highly unlikely that the names of such scripts will match that of known vulnerable scripts. This is assuming that the find the server in the first place.

Therefore, it will be harder for attackers to find, and those that do will most likely be targeting the server for a specific purpose. This implies that any attacks on this server will be undertaken by more skilled intruders than the students themselves, or at least with a definite purpose, hence the higher vulnerability. This doesn't mean that someone won't stumble upon the server, but that it is more likely that a compromise will be attempted by a skilled, purposeful attacker than a casual attacker.

The threat that students might engage in illegal activities on the server is possibly the highest of all. Previous experience at this organization has led to the belief that students are more interested in using facilities for their own gains, rather than for actually attacking any systems.

It should be noted that their purposes may include or require some form of attack, but this appears to be in the minority. That is, in the past, students have not attacked systems to enable some other purpose. Rather, they have found vulnerable systems that already allowed them to do so without needing to cause harm to the system.

For example, one student discovered how usernames for accounts on the organizational HTTP proxy were constructed, and the default password. Using this information, the student was able to obtain free Internet access using accounts where the password had not been changed from the default.

The vulnerability of such activity is high as the organization could be held liable for the actions of students. This may result in legal action, loss of reputation or some other undesirable consequence.

Table 1 summarizes these risks.

Description	***Threat***	***Vulnerability***	***Risk***
Malicious scripts	*Medium*	*Medium*	*Medium*
Vulnerable scripts	*High*	*High*	*High*
Illegal activity	*High*	*High*	*High*

Table 1: Risk assessment

During

A number of security measures are discussed in this section. This includes password protection, "suexec", firewalls, system maintenance/monitoring, and policies.

Password Protection

In the “Before” section, the second category of vulnerabilities was identified. This stated that students may write insecure scripts that could be exploited by an attacker. This leads to perhaps the most obvious security measure. Password protection.

If an attacker is unable to access a student's script over the web, how can they exploit it?

To prevent attackers exploiting student scripts, each user's directory is protected by Apache access controls using basic authentication[9]. The pages are only accessible with SSL/TLS[10] encryption to prevent attackers from “sniffing”[11] user passwords. Only the user who owns the directory and teachers are permitted access to user directories.

This effectively stops attackers from exploiting vulnerabilities in students' scripts, but is not generally considered a particularly strong form of protection. As was noted in the “Before” section, policies relating to user passwords are being implemented, and this document will assume such policies are in place. This is pertinent because these policies should strengthen the security of user passwords.

However, as passwords are generally considered a weak form of security, and as part of a defense in depth strategy, further protection is required. A number of these additional protections apply also apply to the other two categories. They are discussed throughout the remainder of this section.

suexec

One of the biggest problems in allowing students to run CGI scripts is that by default, they all execute with the permissions of the web server. As a best case scenario, a student will be able to read any files that the Apache user

can read and at worst, will be able to write to any files the Apache user has permission to.

What this means, is that a student will be able to read any other student's work and potentially delete other users' work. They may also be able to delete arbitrary files owned by the Apache user. To alleviate this problem, “suexec” is used. “suexec” is a mechanism that allows for user scripts to be executed with the permissions of the individual user rather than the permissions of the Apache user.

This system is set up with web serving from users' home directories with “mod_userdir”[12]. This designates a sub-directory (“public_html” in this case) in each users' home directory, that can be accessed via HTTP with URL's of the form “http://<host>/~<username>/”. Replace <host> and <username> with the web servers DNS[13] name and the user whose web site is to be accessed respectively.

When web serving from user directories with "suexec" enabled, all scripts located in user directories are executed with the permissions of that user. The user directories must be set up to execute scripts. In this case, all users have a subdirectory named “cgi-bin” in their web directory that is configured to allow scripting[14].

“suxec” has certain requirements before it will run. These are primarily security checks, as misconfiguration can cause unexpected results for the unwary administrator. "suexec" requires all user CGI scripts and the directory that contains them, to be mode 755 without the suid or sgid bits set[15]. This makes them readable by any user on the system.

To prevent other users from reading files in these directories, each users' home directory is configured mode 750 and group owned by a common group “webadmins”, that the Apache user and a number of staff are members of.

This way, Apache can read the contents of each users' home directory and execute any scripts within, while preventing unauthorized users from reading the contents of the directory.

Thus the system, based around "suexec", ensures students' CGI scripts execute with only the permissions they have been granted rather than the permissions of Apache. This allows the administrator of the box to tighten security and lock down the permissions for each user. Students are not able to view other students' files, nor are they able to overwrite any files that they don't own themselves.

Of course, this measure applies only to file system permissions. Students are still able to run executables they have permission to. If such an executable has a vulnerability, there may be a way for a user to elevate their permissions. Students will also be able to run scripts that access the network so may be able to launch attacks from this system. In short, the "suexec" mechanism adds a great deal of security to the system, but as with the password protection should be considered as part of a defense in depth strategy.

Firewall

Use of a firewall on the web server itself and use of the corporate firewall can

provide some additional protection. The main reason for running a firewall on the web server is that there are systems on the same subnet as it, that can be accessed directly without passing through the corporate firewall first. Also, it provides another layer in the defense as part of a defense in depth strategy.

Each of the vulnerabilities listed in the “Before” section can be reduced with the use of a firewall. The access allowed (by the firewall) to and from the web server is limited so that malicious network activity is limited.

There is a certain amount of required “system” traffic that has to be allowed. This includes DNS, syslog[16], and SSH[17]. Of course, HTTP traffic must also be allowed, otherwise the web server would be of no use at all!

SSH, HTTP and DNS replies are allowed incoming, while DNS requests and syslog are allowed outgoing to specific hosts (i.e. the DNS server and the syslog server). ICMP[18] traffic is also limited to essential traffic only. The main effect to usability is that they can't “ping” the server as ICMP echo and reply are blocked. This causes some problems with a small number of user's, but has been deemed an acceptable inconvenience.

These measures limit the scope of an attack someone can launch over the network from this host. Users aren't able to attack any services on another host (e.g. Sendmail vulnerability[19]) except for the necessary DNS and syslog servers.

In the case of DNS, functionality has won out over security. It would be possible to run the server without any DNS although this potentially opens up security holes with respect to reverse look ups, not to mention usability issues.

In the case of syslog however, part of the functionality of the security system itself relies on remote logging, so it is certainly required. As is often the case with security, a trade off has to be made however. The very existence of the logging server makes it open to attack, and has to be left open unfortunately. Keeping the logging server patched helps to protect it from common attacks, but Denial of Service (DoS)[20] attacks are still a major concern.

If an attacker was able to launch a DoS attack on the logging server that prevented the server from accepting logs, they would be able to attack another system knowing that the logs would not be collected on the logging server. A certain amount of security by obscurity limits the risk of this happening, but is not something that can be relied on in any circumstance.

The firewall on the web server is configured to rate-limit traffic sent to the essential services. This will require ongoing “tweaking” to ensure that these services still function, but the possibility of attacks are limited.

Of course, these measures limit the usability of the system itself. For example, a student can not write a script that emails the results of a POST[21] from a web form. Finding a balance between security and usability will be an ongoing process.

The firewall also limits the ways the box can be attacked, but this is outside the scope of this document as incoming HTTP access is allowed, which is what this document is primarily concerned with (access to CGI scripts).

System Maintenance / Monitoring

The system needs to be regularly monitored and actively maintained. Vendor patches are applied regularly to keep the system patched up to date. This is in line with procedures in place for other systems on the network.

The server is also running tripwire[22] which is producing reports twice a day. This includes changes in users' home directories to provide a history of user activity on the system. It also shows up when any system files are changed which should allow the administrator to act quickly if an attacker modifies the system.

Unfortunately, no automatic parsing/notification is enable for this system. The administrator does receive an email for each report, but at this stage it must be manually checked which adds some overhead to the administration of this box. A tired or rushed administrator could miss important information.

System logs are sent to a central secure log server. The protection of this server, whilst important, is outside the scope of this document.

The logs are monitored with “logwatch”[23], which sends summary information via email once a day to the system administrators. This filters out unimportant entries so the it's less of a burden on the administrators, and so the administrators know that most of the contents of the email are important. This is critical because any important information is likely to be spotted by even a tired administrator.

"suexec" logs each request for it to run a CGI script. "suexec" does not make use of the syslog daemon. It writes it's own log files, but syslog is able to monitor these files and forward each log entry to the central log server.

The "suexec" logs include information on the time the script was executed, the user and group that the script runs under and the script that was executed. Any errors are also logged. This provides the administrators with a history of which scripts have been executed and when. It also shows all the errors. These two pieces of information can be very valuable, not only in monitoring for suspicious activity, but also for tracking down attackers.

Policies

All students sign an acceptable usage policy relating to the organization's computing facilities before enrolling. This covers such things misuse of equipment and using the facilities to conduct illegal activity. Before granting accounts to this web server, it can't hurt to remind students of this agreement. Each student is given a fact sheet on the system, which amongst other things, reminds the student of the agreement they signed when commencing study at the organization. It also outlines some of the security measures in place and clearly states that activity on the server is monitored.

This is a very strong deterrent to students intending on attempting to compromise this system. The threat that they may be expelled or have their qualifications withheld will probably mean that most students won't even attempt to violate the usage policy. This certainly is not something to rely on, but impressing the seriousness of such transgressions on the students will improve the overall security of the system.

Staff are generally considered “trusted” users, however it is probably prudent to remind staff that similar policy applies to them also. Staff should also be aware of the policies relating to students.

As staff need to apply to the administrators of this system to have an account created, this is the perfect time to inform them of the policies regarding this system. A reminder of this, and other documentation on the system will be sent out to all staff with accounts on the system at the beginning of each semester as well.

After

The mechanisms put in place should yield a more secure system. There are still a number of improvements that could be made, but the inherent risks associated with this problem are still there. They have been minimized, but for a system such as this, it is very difficult to state that all the risks have been removed. However, the current state of the system is more secure that before and provides an adequate level of protection.

Perhaps the strongest tactic used to mitigate the risk is the usage policy. Whilst this is not a tangible physical security measure per se, it does have a very strong practical effect. If students aren't attempting to break the security, then the risk is much lower.

In practice, visible security deterrents go along way to preventing attacks from occurring. Most people will not open a door marked “Staff only”, and security cameras will deter many casual attackers. In this case, the “Staff only” warning is the usage policy, and the security cameras, the visible security measures like "suexec" logging script access.

In the case of the usage policy, it doesn't enforce the security of the system and most certainly should not be relied on. It does however, keep the students mindful of their responsibilities in using the system. Clear reminders of the usage policy before access is granted ensures that students are thinking about security and about their actions on the system.

Of course, this leaves the class of attackers that, even with all the deterrents, are still willing to attempt to subvert the security of the system. These attackers are often more skilled (at least the ones that don't get caught immediately are!), and the subject of the attack is most likely of greater worth than that of the casual attack. This equates to a lower threat, but one with a potentially higher impact. This leads in to the other security measures put in place.

Password protection on user web-space provides protection from attackers exploiting security holes in student CGI scripts. Other organizational policies and procedures are either in place, or being developed to secure user names and passwords. While this mechanism prevents an attacker from accessing student CGI scripts, it relies on what is generally considered a weak form of protection. Some of the risk has been mitigated by password policies and procedures, yet there still remains a vulnerability. The short duration of user accounts on this system limits the time when a user account is vulnerable.

This protection, whilst not totally removing the possibility of an outside

attacker from exploiting vulnerabilities in student CGI scripts, does prevent attacks from all but more determined attackers. This is because more than one layer of security must be compromised for an attack to succeed.

By limiting what a user's script can do with "suexec" and file system permissions, both attacks by students and attacks from outside are limited in scope. If an attacker is able to get through the system security, or if a student writes a malicious script, the range of attacks they are able to perform on the web server are limited by the permissions built in to the operating system.

As CGI scripts are executed with the permissions of the user, the damage is limited to within that user's own home directory. The "suexec" mechanism removes a whole range of attacks on the system.

There is however, the possibility to exploit a vulnerability in "suexec" itself. Keeping the system patched is the primary defense against this, and is performed as part of the entire organization's systems administration procedures.

There is still yet another possibility: that an attacker can exploit some other program/service running on the web server. This could potentially be used by an attacker to escalate privileges on the system, thereby bypassing the protection "suexec" offers. All non-essential services and programs have been removed from the system to minimize this risk. Patching of the server has also minimized this risk.

Using "suexec" means that an attacker must compromise either "suexec" itself or some other part of the system to obtain the necessary permissions in order to carry out an attack on the system. Even then, system maintenance limits the possibility of this happening.

The removal of non-essential services does however, reduce the usability of the system. An ongoing part of the maintenance of this system will be finding the balance between usability and security.

If there is no mail service running, then it can't be attacked. On the other hand, this prevents users from writing scripts that send email (a common usage of CGI). If this was opened though, a student could use the service for sending Spam. Providing a secure mechanism for user's to send email by scripts will need to determined. As this system is used more, other similar requirements are sure to surface, which will be dealt with as needed.

Firewall software is installed on the web server itself and all network traffic destined for hosts on a different subnet to the one the web server is connected to passes through the corporate firewall. This limits the possible attacks that can be launched from the web server against other hosts. By limiting network traffic to and from the web server to essential traffic, attackers are only able to attack these essential services/systems. This has protected other systems but does leave the possibility of an attack on the essential services.

Rate limiting can reduce the risk of attacks on essential services, but also reduces the functionality of these services. In the case of remote logging, this impacts the overall security of the system. For example, if the firewall drops a security alert sent to the logging server because of rate limiting, then it will

likely go unnoticed. Therefore, a balance needs to be struck between security and functionality of essential services. This will require further testing and, ongoing monitoring and adjusting of the firewall as needed.

The firewall system has improved the security of the overall environment by preventing the system from sending and receiving any traffic from non-essential system. However, the essential systems need to be monitored and well maintained to reduce the risk of attacks on those systems.

The system monitoring processes put in place also add to the overall security of the system. Although this is partially a reactive measure, it does allow administrators identify potential security problems and helps them discover security breaches.

In some cases, security problems can be identified before they happen enabling the administrators to address these problems. It also may be possible for administrators to identify attacks in progress, making it possible to stop the attack as it happens.

At the very least, it should be possible to discover attacks that have already taken place. With the information collected, an administrator should be able to discover how the attack was executed, and fix any vulnerabilities associated with the attack to prevent it from happening again.

Overall, the security of the system and the environment around it has been improved but there are still potential problems. There are several additional measures that have been identified that may further improve the system. Two of these are a “chroot” jail and a secure version of the Linux kernel.

Apache could be run under a “chroot” environment to gain some measure of separation between it and the rest of system. This enables the administrators to limit which files are accessible by Apache and therefore student CGI scripts. It also means that a successful compromise of this system does not yield access to the entire system, but only to the file system below the “chroot” jail.

Secure versions of the Linux kernel allow the system can be further locked down. Processes can be assigned roles on top of the basic file system protection. This would allow an administrator to tightly control the sort of things (e.g. Network access, system calls) a CGI script could do on the system.

Both of these measures however, require a reasonable amount of forethought and planning before implementation. This is certainly not trivial, and will be looked into as part of the ongoing development and maintenance of the system.

Overall, the system security has been improved by the measures outlined in this document. Although a number of potential vulnerabilities remain, careful monitoring and maintenance of the system should alleviate these risks. Ongoing maintenance is also required to maintain the balance between security and usability of the system. There are also several additional means of improving the security of this system.

Bibliography

Apache Foundation, “Authentication, Authorization, and Access Control” “Apache HTTP Server Project” URL: http://httpd.apache.org/docs/howto/auth.html (03 Aug. 2003).

Apache Foundation, “Apache suEXEC Support” “Apache HTTP Server Project” URL: http://httpd.apache.org/docs/suexec.html (03 Aug. 2003).

Apache Foundation, “Dynamic Content with CGI” “Apache HTTP Server Project” URL: http://httpd.apache.org/docs/howto/cgi.html (03 Aug. 2003).

Cole, E. AND Fossen, J. AND Northcutt, S. AND Pomeranz, H., 2003 “SANS Security Essentials with CISSP CBK Volume 1”. Version 2.1. USA: SANS Institute

Cole, E. AND Fossen, J. AND Northcutt, S. AND Pomeranz, H., 2003 “SANS Security Essentials with CISSP CBK Volume 2”. Version 2.1. USA: SANS Institute

Phillips, Paul. “Safe CGI Programming” CGI Security. 03 Sep. 1995. URL: http://www.improving.org/paulp/cgi-security/safe-cgi.txt (05 Aug. 2003).

End Notes

1. http://www.w3.org/CGI/
2. http://httpd.apache.org/docs/suexec.html
3. http://unixhelp.ed.ac.uk/CGI/man-cgi?chroot+2
4. http://www.improving.org/paulp/cgi-security/safe-cgi.txt
5. http://www.redhat.com/
6. http://httpd.apache.com/
7. http://www.ietf.org/rfc/rfc1777.txt
8. http://www.w3.org/Protocols/
9. http://httpd.apache.org/docs/howto/auth.html
10. http://www.ietf.org/rfc/rfc2246.txt
11. http://www.linuxsecurity.com/resource_files/network_security/sniffing-faq.html
12. http://httpd.apache.org/docs/mod/mod_userdir.html
13. http://www.ietf.org/rfc/rfc1035
14. http://httpd.apache.org/docs/howto/cgi.html
15. http://www.onlamp.com/pub/a/bsd/2000/09/06/FreeBSD_Basics.html
16. http://www.ietf.org/rfc/rfc3164.txt
17. http://www.ietf.org/html.charters/secsh-charter.html
18. http://www.ietf.org/rfc/rfc792.txt
19. http://www.sendmail.org/ftp/RELEASE_NOTES
20. http://www.denialinfo.com/
21. http://www.w3.org/Protocols/HTTP/Methods/Post.html
22. http://www.tripwire.org/
23. http://www.logwatch.org/