International Journal Of Computer Science And Applications Vol. 1, No. 3, December 2008
ISSN 0974-1003

LAYER X: A Novel Distributed Working System for VCS

Ronak Kumar Samantray
Dept. of Computer Science and Engineering
College of Engineering and Technology
Techno Campus, Bhubaneswar
(+91)-9437272864
ronak.rks@gmail.com

Sonal Das
Dept. of Computer Science and Engineering
College of Engineering and Technology
Techno Campus, Bhubaneswar
(+91)-9861062998
sonaldas.sd@gmail.com

ABSTRACT

In this paper we propose a novel layer (called 'LAYER X') which can be deployed at a Version Control Server to enable client-independent authentication system. This helps us to achieve a distributed working system for VCS. We have designed the protocol in accordance to the CVS server (Concurrent Version System), which is one of the most widely used server for version control. Our protocol has been designed such that, both pserver and the challenge based authentication system become client-independent for a particular session period.

Categories and Subject Descriptors

D.2.7 [**Software Engineering**]: Version Control - Authentication, K.6.5 [**Management of Computing and Information Systems**]: Security and Protection - Authentication

General Terms

Algorithms, Design, Experimentation, Security.

Keywords

Version Control System, CVS, pserver authentication, challenge based authentication system, Client-independent authentication

1 INTRODUCTION

Version Control System is a centralized system for sharing information. At its core is a repository, which is a central store of data. The repository stores information in the form of a file system *tree*—a typical hierarchy of files and directories. Any number of *clients c*onnects to the repository, and then read or writes to these files. By writing data, a client makes the information available to others; by reading data, the client receives information from others. The core mission of a version control system is to enable collaborative editing and sharing of data. What makes any VCS repository special is that *it remembers every change* ever written to it: every change to every file, and even changes to the directory tree itself, such as the addition, deletion, and rearrangement of files and directories [1]. In this paper we propose the protocols in accordance to the CVS server which is one of the most widely used version control servers. CVS is one program, but it can perform many different actions: updating, committing, branching, diffing, etc [3].

In a Version Control System each time a user logs in, the contents of a module (in accordance with the access rights of the user) are copied into the client. After first authentication the user need not provide any login information (unless his session or password expires) for rest of the activities (like commit, update or diff etc)

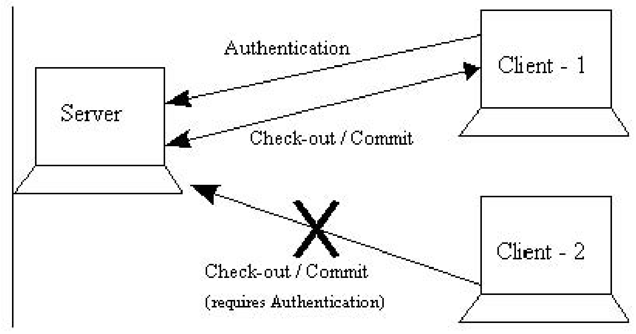

Figure 1. Client-Dependent Authentication System

from the same client. Imagine a situation (see Figure 1) when the user check-outs the module from his office PC (Client-1) and wants to commit or check-out from his home PC (Client-2). In this case the user will have to give the login information again so as to authenticate from the new client (Client-2).

In this paper, we propose a protocol for a novel layer (Layer X) which can be deployed at the version control server to make the authentication client-independent.

2 LAYER X: PROTOCOL(S)

The two primary types of authentication systems for any server are the password-based and challenge-based authentication system. In case of CVS (which one of the most widely used server for version control) the password-based authentication system is known as the pserver method. This method is adopted when rsh is not feasible (for example when the server is behind a firewall).

In the next sub-section we describe the Layer X protocol for password-based authentication system (pserver) method in CVS servers. The *pserver* method allows users to connect to the repository with a username and password that are stored on the repository server. The main advantage of *pserver* is that it permits anonymous, passwordless, read-only access. The main disadvantages of *pserver* mode are that it uses a trivial encoding scheme for passwords and the data stream is not encrypted. *pserver* mode is included in CVS automatically, but it requires a password file that needs to be configured [2].

2.1 Protocol for password-based system

On a CVS server the pserver method uses the 2041 port. The user logs in by the following command [3].

```
$cvs -d:pserver:bach@faun:/usr/local/cvsroot login
CVS password:
```

After user enters the password, CVS verifies it with the server. If the verification succeeds, then that combination of user name, host, repository, and password is permanently recorded, so future transactions with that repository will not require any login from the user. The records are stored, by default, in the file `'$HOME/.cvspass'`. The format of that file is human-readable, and to a degree human-editable, but note that the passwords are not stored in clear text--they are trivially encoded to protect them from "innocent" compromise (i.e. inadvertent viewing by a system administrator or other non-malicious person).

When the user shifts from Client-1 to Client-2 (see Figure 1), these details stored in `'$HOME/.cvspass'` are absent in the new client. User will have to login from Client-2 before any transactions takes place. So to enable distributed working system in pserver method, we propose the following protocol.

Step 1: Copy `.cvspass` file into the home directory of Client-2.

Step 2: Copy the working directory (recursively) so that all the subdirectories and the hidden files are also properly transfered into Client-2.

Step 3: For each sub-directory under the working directory repeat from Step 4 to Step 5

Step 4: Open `CVS/Root` file. Edit path of root repository according to Client-2.

Step 5: Open `CVS/Repository` file. Edit path of the working directory of the project in accordance with Client-2.

Step 6: Configure the `CVS_RSH` variable of the Client-2.

```
client2$ CVS_RSH=rsh; export CVS_RSH
```

(NOTE: In Step 1 we copy the password file into the new client. But plain copy of the file will create problem as the password is not stored in plain-text. So, proper steps must be taken while copying this file into Client-2). In the next sub-section we describe our Layer X protocol for challenge-based authentication system.

2.2 Protocol for challenge-based system

In this kind of authentication system the user generates a key-pair. User stores the public key in the server and preserves the private key in the client from which he wants to work from. The detailed working of the challenge-based authentication system (which includes generation of the key-pair, installation of the public key on the server etc) can be found in [4][5].

In case of CVS, *ext* is the most commonly used access method, and it is usually used with SSH [3]. *ext* stands for external, which refers to an external (to CVS) *rsh* or remote shell program. This method uses a remote shell program to connect the sandbox client computer to the repository server computer. The *server* method is almost the same as the *ext* method.

To enable a distributed working for such a system we propose the following protocol.

<u>LAYER X</u>

Step 1 : For each incoming request repeat from Step 2 to Step 13

Step 2 : Check the *database* if any such entry with **U**=*user* matches with that of the incoming packet from *user@client*.

Step 3 : If match found then

Step 4 : Change the *user@client* in the incoming packet to *user@org_client* corresponding to the value of **T** in the entry and set **Tt** = *user@client*.

Step 5 : Forward the request to the CVS server and wait for the response from the server.

Step 6 : On receipt of the response from the server change the target tag in the outgoing packet to *user@client* from *user@org_client*.

Step 7 : Else If match not found & login request is received then

Step 8 : Forward the request to the CVS server and wait for the response.

Step 9 : On receipt of the response from the server check if the login was successful

Step 10 : If successful then

Step 11 : Create an entry as **T**=*user@org_client,* **Tt**=*null* and **U**=*user* as in the outgoing packet.

Step 12 : Set the timeout period for the entry as η.

Step 13 : Else Forward the packet to the target system.

Step 14 : Check the *database* for any entry if the time of residence of the entry has exceeded the time-out limit η. If any such entry is found then remove the entry.

(NOTE: When the user shifts from Client-1 to Client-2, he must also carry his private key into the new client. Since the private key of the user is still required by Client-2 so no adversary can take any unnecessary advantage of the distributed working system.)

Symbols Used in the above Protocol

T : This entry in the database corresponds to the original host name of the user through which he had successfully logged in.

Tt : This entry in the database corresponds to the new client from which the user tries to log into the server (within the given time period). This client may not be same as initial client from where user had authenticated himself.

U : This entry in the database stores the user name of the current session of the user. We know that on a public network (for a particular server) each user is provided with a unique name. So this can field serves as the primary key in the *database.*

η : A suitable time period within which distributed working is allowed for any user. This value is set by the server administrator according to the convenience and security rules. If this is set to zero, then Layer X becomes transparent and it corresponds to the present client-dependent system.

*database / knowledge base***:** The database used for storing the data for Layer X. This may be a plain text file or any database like MySQL (for increased efficiency).

3 LAYER X: ARCHITECTURE

The Layer X lies as an intermediate layer between the CVS server and the client systems (see figure 2). It receives all the incoming requests from the clients and applies an appropriate transformation to these incoming packets on basis of the authentication system being used between the client and the server. The reply packet from the server is again transformed by Layer X in accordance with the user session.

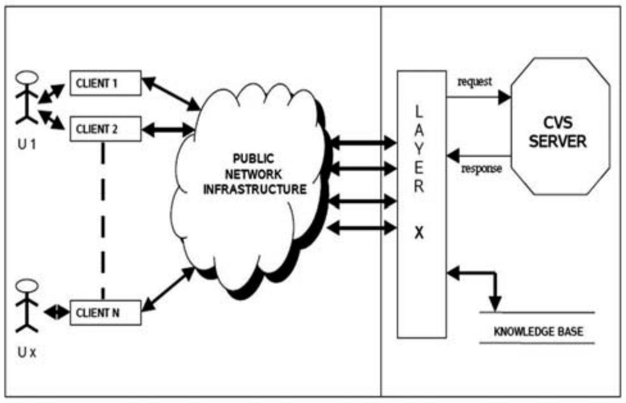

Figure 2. Implementation Architecture of Layer X

The complexity of transformation being carried out is one of the primary factors in the effective response time of the server. So we set the η value such that the burden on Layer X even in peak periods would be less. The issues related to the security and scalability of Layer X has been dealt with in the next section.

4 IMPLEMENTATION ISSUES

There might be some common questions which might be raised regarding the Layer X that we have proposed. Some issues have been dealt with in the following sections.

4.1 Purpose of a time-out period

We have set a time out value as η as the time out value for a user. This value signifies the amount of time the user can have the flexibility of distributed working. By the term distributed working we mean that, if user had logged in from a a system say Client-1 then within a specified time period he can directly login (without giving any passwords) from any other client. The only constraint is that the new client should have the private key of the user for file management operations. So the user would have to manually transfer his private key from Client-1 to the new client.

Now one question might arise in your mind that 'Why did we set a time out period, why cannot we have a permanent distributed working for a user?' Well imagine a situation when users had a permanent record in the knowledge base of Layer X to enable permanent distributed working facility. But this would in turn increase the response time of the server, because as the number of entries in the knowledge base would increase the Layer X would take more time in taking decisions and delay the server response time. It would also be an unnecessary overhead on the server.

4.2 Security of a user's session

The most probable question that can be raised is about the security of Layer X. Layer X might seem very vulnerable in the first instance because 'it provides no-password login from any client for a user within a specific time period'. But if we look into the architecture of Layer X (as explained in section 3.) we would observe that during every transaction between the server and client the private key of the user is always required to sign the packets sent by the user. And since this key is private with respect to any user we can be sure that no client can claim to be any user without any proof/certificate!! (Base of Layer X still remains the challenge-based system. Layer X can be thought as just a plug-in for the challenge system). Thus our Layer X is also secure.

4.3 Denial-of-service attack

The Layer X we have proposed has a loophole (though its not a major flaw in the design). We do not monitor the frequency of incoming requests from a particular client. An adversary might send huge number of requests to the CVS server just to increase the load on the server. This would lead to denial of service to some legitimate users also. But this is not a major issue. A trivial solution to this problem can be to set another time period for any client, such that it wont be able to send multiple requests within that specific period. A more efficient method can be designed by setting proper firewall filters.

5 FUTURE SCOPE

5.1 Dynamic Session Periods

In our Layer X we have used η to store the time-out value for a session. This value is unique to all the users. But we can make it more intelligent by making the time-out value dependent on the access rights level of a user. For example an administrator will have longer session than a normal user. This would give more flexibility for the administrator.

Another approach to make the session periods dynamic is by monitoring the load at the server. i.e. on basis of the load at server we can adjust the session periods. For example during peak load period the period value should be low. Because Layer X would have a large number of entries to handle. This would further increase the time response time of the server. To achieve this goal of we need to access the actual test our Layer X on a server like sourceforge.net or freshmeat.net.

5.2 Web based CVS access

Since Layer X makes the server access client independent so we can en-corporate web based access to CVS servers. Today we have only command line access to these servers because of the speed and handling of file transfer. So if an efficient file transfer system for HTTP can be developed then along with Layer X we can have a robust and flexible system.

5.3 Single-Sign-On

A Single-Sign-On protocol can be designed for all the CVS servers using any of the lightweight libraries like YADIS etc. This would allow the users to access all the CVS servers (like sourceforge.net, freshmeat.net etc) using a single login id. But to design the respective governing companies must agree upon a common license design for the projects handled by them. The concept of OpenID is widely used today for accessing mails and managing different online tools. We need to study the licensing terms of the VCS servers so as to know the feasibility of such a Single-Sign-On protocol.

6 ACKNOWLEDGEMENT

We sincerely thank Mr Subhransu Behera of RedHat Technologies for his encouragement, support and ideas.

7 CONCLUSION

An in-depth analysis of distributed working approach for version control system implemented under the pserver and challenge based authentication was carried out. It gives much flexibility to the users. This makes record keeping and collaboration much easier. A distributed system shall enable us to access from different clients and implement the project changes successfully.

It was observed that the pserver mode is a password authentication protocol. The encrypted passwords have to be present on the client, as well as the cvs working directories need to be incorporated. Thus the distributed working can be ensured by coping the .cvspass file into the new client's home directory as well as the various working modules and configure the cvs_rsh variable of the new client.

However, the password based system has its own flaws (as discussed in section 3.3). Thus the challenge based authentication is a much more preferable authentication system. This works by generating a key pair and the access is granted by the unique combination of user name and host name as studied. So, we need to have the same set for proper implementation.

Keeping these in view, we proposed the Layer X method in which a layer was placed on the CVS Server. When the user logged in this layer transformed his login into the global identity which the server recognized. Thus, we see that irrespective of the host name the client could be authenticated. Client-independent access enables it to be implemented in web based access also. We have also minutely analyzed all the aspects of Layer X and the future work has been appropriately suggested.

8 REFERENCES

[1] B.C.Sussman, B.W.Fitzpatrick and C.M.Pilato. 2007. Version Control with Subversion, Second Edition.

[2] J. Vesperman. 2003. Essential CVS. pp. 197-200.

[3] Karl Fogel and Moshe Bar. 2000. Open Source Development with CVS, 3rd Edition.

[4] R. Bragg, M. R. Ousley and K. Strassberg. 2004. Network Security: The Complete Reference. pp- 139

[5] J. Maliery. 2005. Hardening Network Security. pp-104