Fokstraut and Samba – Dealing with Authentication and Performance Issues On A Large Scale Samba Service

Robert Beck & Steve Holstead – University of Alberta

ABSTRACT

At the University of Alberta, we have approximately 55,000 user id's using central services authenticated by Kerberos. We use AFS for central file service. We use Samba to provide Windows compatible access to much of our central file service. Samba contains a number of useful features for Microsoft Windows compatibility, including a kludge to deal with the problem of Windows sending an all uppercase version of a user's password. We observed that when Windows connects to a share, it frequently attempts many incorrect passwords repeatedly before trying the correct one. This created a very heavy authentication load on our central Samba service when users would connect every morning and authenticate. We observed this load and noticed that most of our problems were caused by repeated attempts to authenticate, and the high cost of checking these attempts.

To help reduce the load due to authentication, we implemented FOKSTRAUT, a set of modifications to Samba to cache recent password failures and successes in a DBM database built by the Samba server as it runs. By caching the recent failures we avoid expensive re-checks of the (many) other passwords Windows likes to send us. We also cache the correct case of the real password, and by doing so we avoid the expensive overhead of "cracking" an all uppercase password When Windows decides to send one. We also use FOKSTRAUT to cache the NT and LanMan password hashes of a users password once we see a successful authentication. This then allows us to use the newer Windows NT password hash after the user has connected once, without having to centrally convert and maintain a large SMB password file, and while maintaining the ability of our server to access services such as AFS which can not be authenticated against using the Windows password hash alone. Performance on our service has been drastically improved since the implementation of FOKSTRAUT.

Introduction

Our experience started with a performance problem. We are a large site which uses Kerberos [1, 2] with approximately 55,000 user ID's. We have a large scale Samba [4] service which serves up files from our central AFS [3] file service using the Microsoft Windows (Windows) [9] SMB protocols [5, 6]. It was and is exceedingly popular with our on-campus clients who use it to access centrally maintained file space when they must use a Windows machine. The problem we had was that while the service would perform well during the day, it would fall over under a crippling load in the mornings when users would establish their initial connections. This problem hit us rather suddenly this last year when various factors resulted in a large increase in use at the start of a school term. We then set out to determine the source of the problem and what, if anything, we could do about it.

The Source of our Problem

We examined the server at various times, and we found that it was completely CPU bound in the mornings when users were connecting, with many individual `smbd` processes competing for CPU. Once this had been determined, we set out to find out where it was spending its time by adding a few key little diagnostic printfs to our Samba code. Watching the users connect in the morning we found the source of our CPU hogs.

On connection, many of our users' machines would try several times with a password that obviously wasn't their password on our service. Several attempts had to be made, and fail, before the correct password would be given by the Windows client machine. The result of this was that our server spent a lot of time failing the same password, for the same user, over and over again.

Our Kerberos passwords may only be changed through a password changing mechanism that forces the use of "good" passwords, and does dictionary and pattern checks against any attempts. Very frequently on our server, users would attempt to connect initially with very bad, easy-to-guess passwords, and then only after several failures, would try the correct password. These initial passwords which the connection would attempt and fail were the same as the users "Windows" password (NT domain or otherwise) with alarming regularity. So, by configuration or design, many users' Windows machines were determined to tell us their local passwords before attempting to give us the one we wanted to authenticate them with Kerberos. This has interesting security implications – it's easy for our clients to configure Windows to leak their passwords to an SMB server. For the purposes of this exercise, we were only concerned about the performance implications.

In addition to the repeated attempts of an incorrect password, many of our users' Windows machines will attempt to use an all uppercase password. We have little or no control over the client software version, and so were not able to avoid this problem at the client end. This means we have to support the password level [10] "crack" feature in Samba where the server would repeatedly try different case combinations. While we knew this was a possible cause of the heavy load generated by authentication, the effect was greatly magnified by the repeated attempts of incorrect passwords – each incorrect password had to be checked repeatedly in multiple cases.

We were faced with a huge overhead on authentication which involved "cracking" passwords, and doing it repeatedly on the same bad password. Short of abandoning our Windows clients we had to find a way to deal with this on our Samba service.

The NT Password Hash

Samba itself supports the Windows NT encrypted password scheme. To use this, a UNIX administrator creates an "smbpasswd" file [11] in which the Windows NT hash of the user's passwords are stored. In this method, the client machine passes the Windows NT style hash, and is not faced with the problem of case insensitive password. One possibility we examined was that of converting our service to require this scheme. We could not do this for two reasons:

1. We maintain our central ID's in Kerberos, We did not want the added administrative cost of maintaining a 55,000 user flat file of SMB passwords.
2. We access AFS file space from this server – There would be no way for the server to get an AFS token to access the user's file with only the NT password hash (and not the user's Kerberos password or a ticket). This will be a problem with any external service that can not be authenticated against using the NT password hash.

Our Solution: FOKSTRAUT

Having ruled out doing it the "correct" way for Windows by using the NT password hash, we tried to examine what would give us a solution for our environment. First, we made a few observations:

1. The users that handed us passwords that failed would usually try them twice before proceeding to the next password (sometimes another bogus one, and sometimes the right one). This was what most of our CPU bound processes were doing – busily cracking a bogus password.
2. If we knew the real password, and were given a case-insensitive version of it, we could start by checking the "correct" case and avoid the expense of repeatedly trying to crack the password.
3. If we knew the real password we could also compute the NT password hash.

Knowing this, we implemented FOKSTRAUT. FOKSTRAUT is a password success and failure cache for Samba. It uses a DBM database indexed by the user name to store the most recent passwords that failed, as well as the last password that worked for each user. It stores the successful passwords both as the clear text password, and as it's Windows NT hash, enabling the FOKSTRAUT cache to be used to authenticate Windows clients passing a Windows NT password hash for a user that has been seen before. Since it stores the clear text password along with the NT password hash, the server then knows the clear text password of a user authenticated with the NT password hash and can use this to authenticate the user for other services (AFS in our case).

How FOKSTRAUT Works

When our Samba server starts up, it checks for the FOKSTRAUT database, a DBM database indexed by user name. This database is created if it does not exist. The Samba server then stores and retrieves from this database each time a user authenticates.

FOKSTRAUT keeps track of the following information:

- The three most recent failed passwords, and a count of how many times each password has

```
Sep 28 07:23:41 samba smbd[76722]: trying password AUCTIONS for user luser
Sep 28 07:24:08 samba smbd[76722]: Bogus password, user luser, password AUCTIONS
Sep 28 07:24:11 samba smbd[76722]: trying password AUCTIONS for user luser
Sep 28 07:24:47 samba smbd[76722]: Bogus password, user luser, password AUCTIONS
Sep 28 07:24:49 samba smbd[76722]: trying password AUCTIONS for user luser
Sep 28 07:25:18 samba smbd[76722]: Bogus password, user luser, password AUCTIONS
Sep 28 07:25:20 samba smbd[76722]: trying password Ac94metoo for user luser
Sep 28 07:25:20 samba smbd[76722]: worked first time for user luser, password Ac94metoo
```

Figure 1: Example of our diagnostic log output. Note the times between success and failure: during this time this smbd was busy using CPU attempting combinations of this password.

been given and failed since the last successful connection.
- The last successful password, stored as clear text, NT Hash, and LanMan hash formats.
- An ‘‘smbpasswd’’ entry [11] for the user – this is used so that we can take advantage of Samba’s NT and LanMan password features if we find a user in the database.

When a user connects to our FOKSTRAUT modified Samba server, the server will look up and retrieve the information about previous connection attempts in the FOKSTRAUT database. It then compares the password the user is attempting to any previously cached failed passwords. If that password is the same as a previously cached failure, the connection is immediately failed without checking against the system, and the failure count for that password is incremented in the database, and saved for the next time. When the failure count for a failed password reaches three, this password is removed from the database. This ensures that a cached password failure can not continuously fail without being re-checked against the system authentication methods.

Assuming the password didn’t match one of the cached failures, the password is then case insensitively compared to the password recorded as being successful previously. If the passwords match, the correct case stored in the database is used, and authentication proceeds against the system authentication methods. If authentication is then successful, this connection succeeds. All previous failure counts are cleared back to zero, and the resulting record is stored back into the database for the next time.

If the password presented doesn’t match anything FOKSTRAUT knows about we attempt to authenticate in the usual manner. If the password succeeds, the correct case is recorded in the cache, the NT and LanMan hashes are computed, and the entry is saved as in the above case. If the new password fails, it will be added to the cache of failed passwords, replacing the least used (least failed against) of the three saved entries.

If our modified Samba service receives an NT or LanMan hashed password, it checks it against the entry saved in the FOKSTRAUT database. If it is present, an smbpasswd entry from the database is used for the user, and authentication proceeds as it would in the usual case of an smbpasswd based connection in Samba [4]. If that authentication then succeeds, in our case, the server then gets Kerberos Tickets and an AFS token for the user based on the saved clear text password from FOKSTRAUT. At this point if both the NT hash authentication and the Kerberos/AFS authentication are successful, the connection is deemed to be successful. Otherwise, it is considered a failure. and the cache entries (including the password hashes) are cleared.

The choice we made of saving three different cached failures was based on our own observations of what we saw from our users. Three different bogus passwords failing was the most we typically saw before seeing the real password. The choice of failing against a cached failure up to three times before we checked again against he system was based both on our observations of the typical number of tries we would see, along with a desire to make sure we didn’t hold up a user unduly in the (unlikely) case where they make a connection attempt, the password fails, and they then change their password to what they just recently failed with.

Advantages

We have seen two primary results:

1. We have seen a huge performance improvement, we can now easily deal with the several hundred users we get every morning where we could not before. Simple load average during peak times has dropped from peaking at 30 to peaking at 3, with a sustained average dropping from over 10 to less than 2.
2. We now have a mechanism by which we can support the non-cleartext password SMB connections to our server, while still making use of Samba as a gateway to AFS, Users are simply told to connect once from a login server, or with a Windows machine set to send cleartext passwords on connection, and then after that the non-cleartext SMB connection will work.

We are very happy with the increased level of performance these modifications have given us.

Date	runq-sz	%runocc
Tue Feb 29 08:30:05 2000	3.700	80.000
Tue Feb 29 08:40:05 2000	8.900	100.000
Tue Feb 29 08:50:05 2000	10.800	100.000
Tue Feb 29 09:00:06 2000	12.600	98.000
Tue Feb 29 09:10:06 2000	17.500	100.000
Tue Feb 29 09:20:14 2000	21.400	97.000
Tue Feb 29 09:30:51 2000	4.500	97.000

Figure 2: Run queue size and utilization, 8:30 AM to 9:30 AM, week before implementation.

Date	runq-sz	%runocc
Tue Mar 7 08:30:06 2000	1.700	37.000
Tue Mar 7 08:40:06 2000	1.400	48.000
Tue Mar 7 08:50:06 2000	1.800	55.000
Tue Mar 7 09:00:06 2000	1.800	60.000
Tue Mar 7 09:10:06 2000	1.400	37.000
Tue Mar 7 09:20:06 2000	2.000	53.000
Tue Mar 7 09:30:06 2000	1.800	28.000

Figure 3: Run queue size and utilization, 8:30 AM to 9:30 AM, week after implementation.

Disadvantages

FOKSTRAUT has several disadvantages. The first one that comes to first one that comes to mind is security. If a Samba server is compromised and is running FOKSTRAUT, all the users’ passwords are available to the attacker through the database. This is a serious concern to us, and we therefore ensure that FOKSTRAUT is only run on a dedicated, secured machine running no other services and that there is no access by regular users to the machine. We judge this risk as acceptable to us, given that even if the machine were not running FOKSTRAUT, an attacker compromising the Samba server could install a trojaned daemon and collect the same passwords with very little extra effort. The results of a compromise of the Samba server would be bad for us with or without FOKSTRAUT, considering that we must run Samba with cleartext passwords enabled.

The other drawback from the point of view of the Windows NT passwords is that the server does not know the Windows NT hash until the client has successfully authenticated to it once so that the server knows the cleartext password. We get around this by providing a simple script on a login server in which a user can make an SMB connect and authenticate themselves, making themselves "known" to the Samba server. Alternatively, the users can make an initial connection to the service from a Windows machine set to send the clear text password, then switch to using the NT password hash.

Conclusions

A password cache is a major performance win when used with Samba service that must support clear text passwords and can not rely exclusively on the NT password hash protocol. In our case, with the ability to use the NT password hash for authentication and still have the server know the real password, it enables us to use the NT password hash connection mechanism while still supporting our external authentication mechanism (AFS) which is incompatible with the NT password hash. The performance wins we saw changed our service from one that would collapse under the load to one that now remains up for months at a time, providing reliable file service. If you are in a position where you have to offer large scale Samba service, we think this is definitely something to consider.

Availability

Our code is available as a patch for a current Samba distribution. It has been tested and used on OpenBSD [7] and AIX [8] (and should port very easily to anything else remotely Unix-like). It is made available under BSD style license terms, and can be obtained at ftp://sunsite.ualberta.ca/pub/Local/People/beck/fokstraut/ .

Author Information

Bob Beck has a Masters degree in Computing Science from the University of Alberta. He has worked in a variety of systems administration and programming positions at the University of Alberta since 1990. He also works as a consultant, instructor, and programmer with Obtuse Systems Corporation. He is currently the Secure Systems Specialist for the University of Alberta, as well as working on several free software projects, especially OpenBSD. You can reach him by postal mail at Computing and Network Services; 352 General Services Building, University of Alberta Campus, Edmonton, Alberta, Canada, T6G 2H1. You can reach him by e-mail to beck@bofh.ucs.ualberta.ca or beck@obtuse.com .

Steve Holstead has been responsible for programming and system administration duties working at the University of Alberta since 1978. His experience has taken him from the centralized mainframe systems support to a distributed client server environment in which he works today. He can be reached via e-mail to Steve.Holstead@ualberta.ca .

References

[1] J. Steiner, C. Neuman, and J. Schiller, *Kerberos: An Authentication Service for Open Network Systems*, Usenix Association's Conference Proceedings, Dallas, Texas, February, 1988, p. 191-202.

[2] J. Kohl, and C. Neuman, *The Kerberos Network Authentication Service (V5)*, September 1993.

[3] *AFS filesystem information from Transarc Corporation,* http://www.transarc.com/Product/EFS/AFS/index.html .

[4] *The Samba project*, http://www.samba.org/ .

[5] *CIFS information*, http://anu.samba.org/cifs/ .

[6] Sharpe Richard: *Just What is SMB? (v 1.2)*, http://anu.samba.org/cifs/docs/what-is-smb.html .

[7] *The OpenBSD Project*, http://www.openbsd.org/ .

[8] *The IBM AIX Operating System*, http://www.ibm.com/servers/aix/ .

[9] *Microsoft Windows*, http://www.microsoft.com/ .

[10] "Password Level," *Samba Documentation,* http://us1.samba.org/samba/ftp/docs/htmldocs/smb.conf.5.html .

[11] "smbpasswd," *Samba Documentation,* http://us1.samba.org/samba/ftp/docs/htmldocs/smbpasswd.5.html .