A Post-mortem of Yahoo! Account Security

By

Gammarays
(gammarays64@gmail.com)

Contents

Introduction

"Yahoo! Inc. is a leading global Internet communications, commerce and media company that offers a comprehensive branded network of services to more than 274 million individuals each month worldwide." - Yahoo! Media Relations

My initial goal in penetrating Yahoo!'s security was to take action against a constant invasion of advertising bots, booters and credit card thieves that plague the chat system. Without sufficient measures in place to prevent these people coming back I have since realised it would be impossible for any individual to manage the problem.

The purpose of this presentation is to demonstrate the following..

1) *It is possible for an intruder to gain unauthorized access to a Yahoo! Production server (already well known)*

2) *That access can then be used to reverse engineer the authentication system.*

3) *The same access can also be used to access the user database.*

4) *With the correct components it is possible to login in to any Yahoo! Account and in many cases take control of it.*

A quick overview

The Yahoo! network is both vast and complex and while the diagrams that appear later are not representative of this complexity they will show roughly how a process works.

If you have ever used Yahoo! then you may have noticed that once you sign in you can visit any service on the network and be instantly recognised. This is due to the fact that when a user logs in they are issued a cookie that contains all the information necessary to identify them wherever they may go within the Yahoo! network. We will be taking a closer look at this later.

Some terms that will be used throughout..

==========

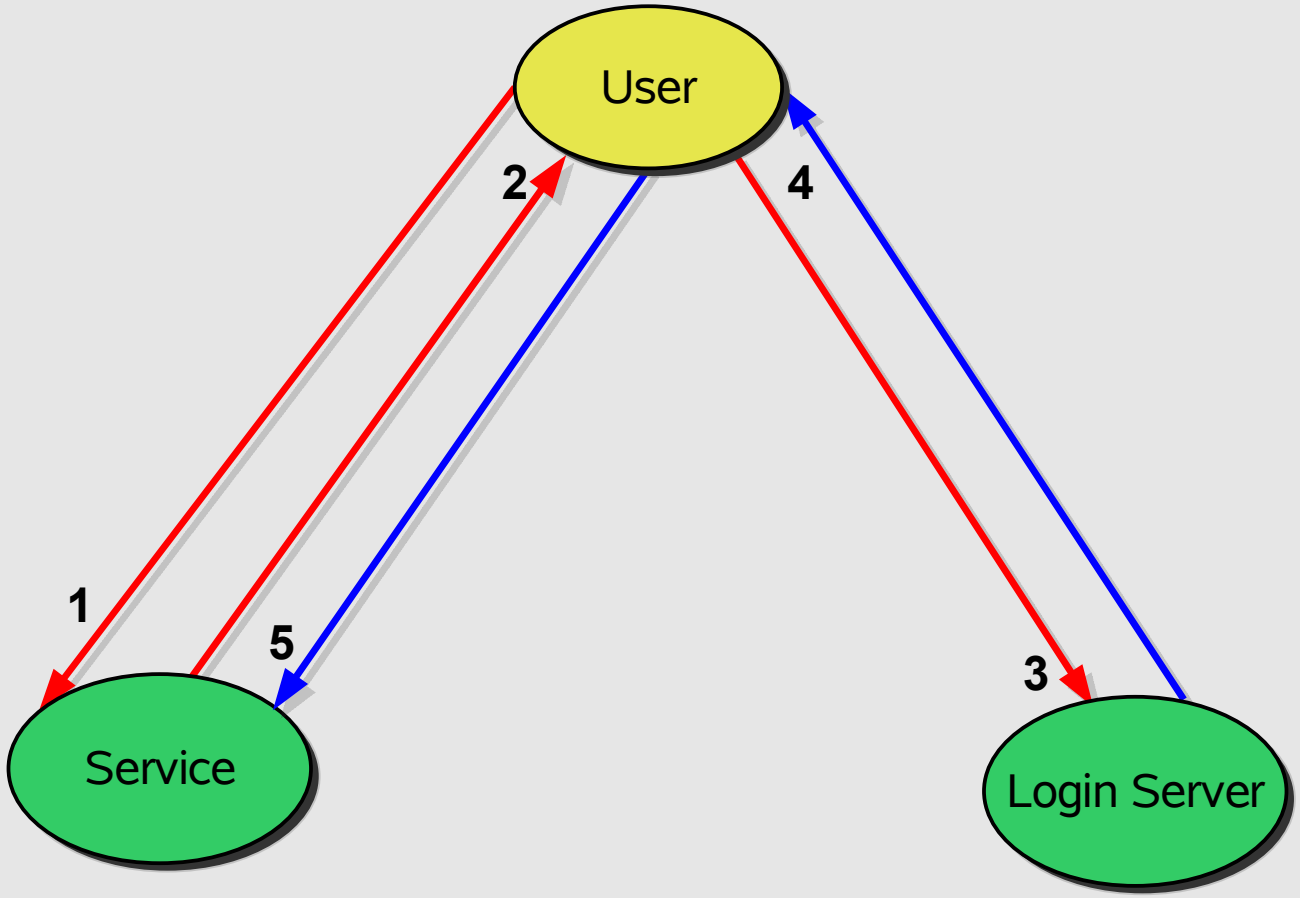
UDB	- User database.
UDB Proxy	- A server which provides other servers with access to the user database.
Y64	- Refers to the process of Y64 encoding, explained later.
YINST	- Yahoo's package manager.
Service	- Simply refers to any service Yahoo! provides such as mail or chat.
YMSG	- The protocol used by Yahoo's chat system.
keydb files	- The XML files that store server-side secrets.
YDBS	- A Package allowing access to the Yahoo database.

The Login Process

Key

No Cookie

With Cookie



1 – User requests a service without a cookie and is rejected.

2 – The server detects the user needs to login and redirects them to a login server.

3 – User enters their Yahoo! id and password to login.

4 – Once the user is logged in they are issued a cookie and redirected back to the original server.

5 – User requests service with a cookie and is accepted.

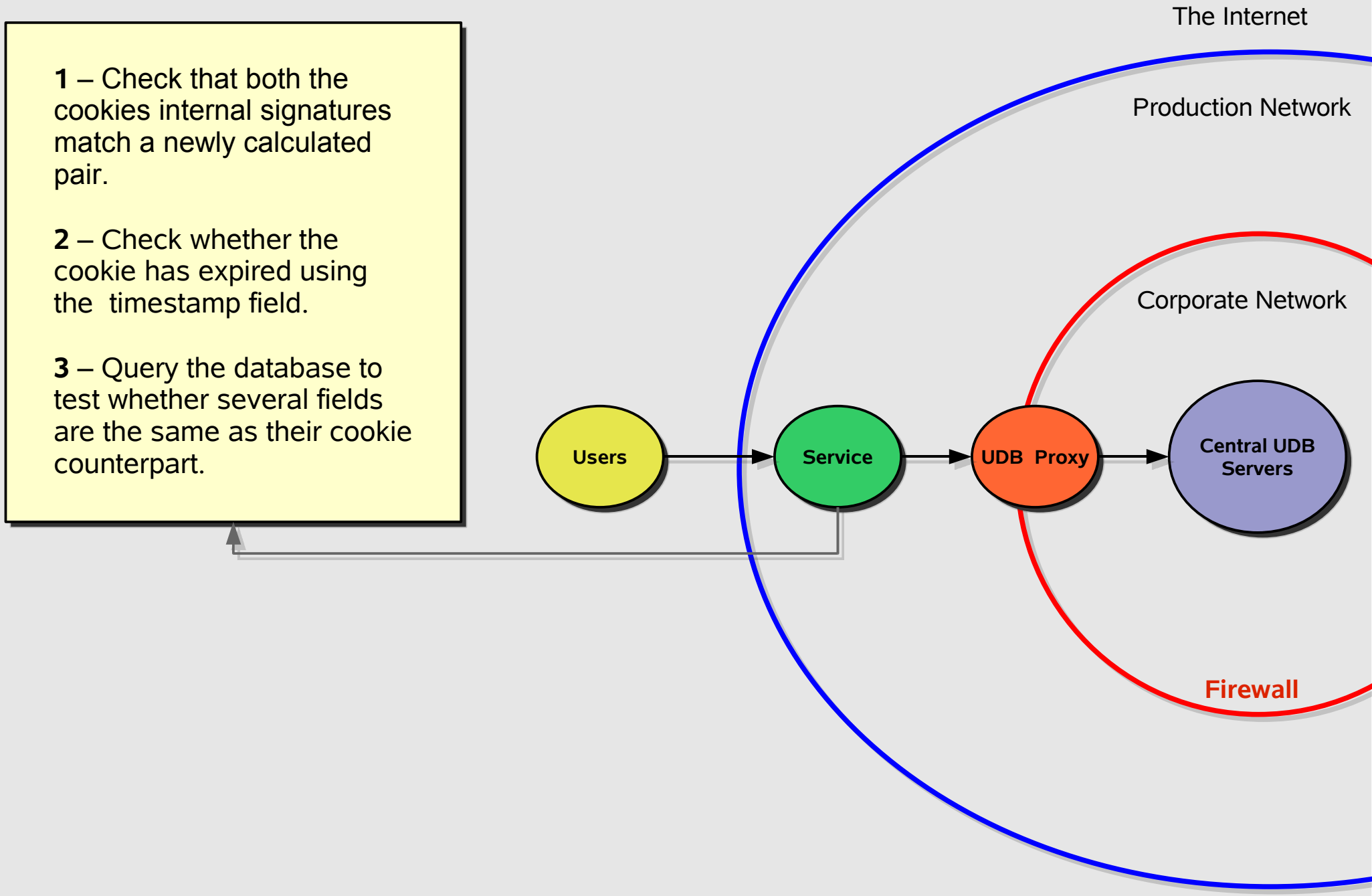
The Authentication Process

(more a hypothetical model)

1 – Check that both the cookies internal signatures match a newly calculated pair.

2 – Check whether the cookie has expired using the timestamp field.

3 – Query the database to test whether several fields are the same as their cookie counterpart.



The Authentication Cookies

When a user logs into their account they are issued with cookies that authenticate them across the whole of Yahoo!. Once in possession of these cookies a user may access a wide range of services.

The two cookies that actually authenticate a user are named 'Y' and 'T', each of these consists of several sub-fields. Some of these fields are vital to the authentication process while others are less important.

Y64 Encoding

In the following sections I will refer to a process called Y64 encoding, anyone familiar with base64 can probably guess that this is Yahoo!'s variation of the process. Y64 encoding is used to encode strings, integers and in some cases the raw result of hashing functions. Y64 Encoding uses the following character set.

“ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789._”

and '-' is used instead of '=' for padding.

The Y Cookie

Cookie Name: v
Other Names: Version
Sub-fields: N/A
Parent Field: N/A
Name in DB: N/A
Description:

Indicates the cookie version, the current version is 1.

Cookie Name: n
Other Names: ID
Sub-fields: N/A
Parent Field: N/A
Name in DB: id
Description:

$Base32-Charset = "0123456789abcdefghijklmnopqrstuv"

ID is a base32 encoded 64 bit integer. The value is randomly generated when the user first creates the account and is stored in the database. This value is checked in the cookie against the database for validation purposes, if it does not much then many of Yahoo!'s servers reject the cookie. Not all id's are forced to be the result of base32 encoding since different generation methods have existed in the past.

The Y Cookie

Cookie Name: l
Other Names: Login, Username
Sub-fields: N/A
Parent Field: N/A
Name in DB: login
Description:

$unscrambled = "abcdefghijklmnopqrstuvwxyz0123456789._@-+"
$scrambled = "0123456789abcdefghijklmnopqrstuvwxyz._@-+"

The login field contains a scrambled username. The algorithm consists of a simple substitution cipher where each character in $unscrambled is replaced by it's partner in $scrambled. For example the username "testuser123" becomes "j4ijki4hrst". The scrambled username is then appended with "/o".

The Y Cookie

Cookie Name: r
Other Names: regweek
Sub-fields: N/A
Parent Field: N/A
Name in DB: reg (unix timestamp from when the user registered)
Description:

$Base32-Charset = "0123456789abcdefghijklmnopqrstuv"

Using reg from the database regweek is calculated by first working out when the user registered their account relative to June 30th 1996 - 6:00pm, should a user have registered before this time it is treated as if they had registered at that exact moment. The relative time is then converted to weeks and base32 encoded. The base32 string is trimmed to before the first occurrence of '0' to preserve only the significant portion of the encoded integer. Just like id this value is used for validation purposes.

The T Cookie

Cookie Name: a
Other Names: Age
Sub-fields: N/A
Parent Field: N/A
Name in DB: N/A
Description:

The age field is a Y64 encoded 16-bit variable that contains flags relating to the users age. At least one of these flags indicates if the user is 18+ and can be used as a mechanism to determine whether to display mature content without the overhead of querying the database. both "QAE" or "YAE" are acceptable values.

Cookie Name: d
Other Names: Data
Sub-fields: a, sl, g, ok, zz, tip
Parent Field: N/A
Name in DB: N/A
Description:

The purpose of the data field is simply to hold several sub-fields in one string. it is formed in the following manner.

<fieldname> 0x01 <value> 0x01 <fieldname> 0x01 <value>

The fieldname and value are terminated using the 0x01 character except for the final value. Once the string containing the values is formed it is Y64 encoded.

The T Cookie

Cookie Name: sk
Other Names: Version 3 Signature
Sub-fields: N/A
Parent Field: N/A
Name in DB: N/A
Description:

A version 3 signature is the first of the security mechanisms in the authentication cookies. The purpose of the signature is to verify the integrity of the other fields, in this way a user cannot change any single field without causing a signature mismatch. The signature is created by first forming a string containing the important fields of the cookie in addition to a secret key stored on Yahoo's servers. The string is MD5'ed and the raw hash is Y64 encoded. The encoded string is then trimmed to 11 bytes and prepended with the version string.

Cookie Name: ks
Other Names: Version 4 Signature
Sub-fields: N/A
Parent Field: N/A
Name in DB: N/A
Description:

The version 4 signature is similar to the version 3 signature but with a few additions. The v4 signature incorporates the value of the v3 signature and also uses an extra server-side secret. The v4 signature is also appended with '~' followed by an uppercase letter indicating the version of the tcookie secret. The key version can be deduced by subtracting 64 from the ascii value of the character ie: A = 1, B = 2, C = 3.

The T Cookie

```
// Code to generate a v3 sig given l,n,d and a sufficiently sized return buffer.
// Note: I have obfuscated part of the secret due to the abuse that may occur if it were
// made public.

bool generate_v3_sig(char* l, char* n, char* d, char *result)
{
  MD5_CTX a;
  char  secret[] = "M1+7]69Z![^-#####################################-Xc<O@D1<W#";
  char  hashstring[2048] = "";
  char* pointer;

  unsigned char md5hash[16];
  snprintf(hashstring, 2048, "l=%s&n=%s&d=%s&secret=%s", l, n, d, secret);

  MD5_Init(&a);
  MD5_Update(&a, hashstring, strlen(hashstring));
  MD5_Final(md5hash, &a);
  pointer = base64(md5hash, 16, true); //Y64 Encode

  if(pointer != NULL)
  {
    strncpy(result, "DAA", 3);  // Version 3 (Y64 Int)
    strncat(result, pointer, 11);
    free(pointer);
  }
  else
  {
    return false;
  }

  return true;
}
```

The T Cookie

```
// Code to generate a v4 sig given l,n,d,sk and a sufficiently sized return buffer.

bool generate_v4_sig(char* l, char* n, char* d,char* sk, char *result)
{
  MD5_CTX a;
  char  secret[] = "M1+7]69Z![^-####################################-Xc<O@D1<W#";
  char  tcookie[] = "lElk-################################-lnk-";
  char  hashstring[2048] = "";
  char* pointer;
  char  key_version[4]= "~C"; // Key Version 3 in the keydb file.
  unsigned char md5hash[16];

  snprintf(hashstring, 2048, "l=%s&n=%s&d=%s&secret=%s&sk=%s",l,n,d,secret,sk);
  MD5_Init(&a);
  MD5_Update(&a, hashstring, strlen(hashstring));
  MD5_Update(&a, tcookie, strlen(tcookie));
  MD5_Final(md5hash, &a);
  pointer = base64(md5hash,16,true);

  if(pointer != NULL)
  {
    strncpy(result,"EAA",3);  // Version 4 (Y64 Int)
    strncat(result,pointer,32);
    strcat(result,key_version);
    free(pointer);
  }
  else
  {
    return false;
  }

  return true;
}
```

The T Cookie

Cookie Name: a
Other Names: Age (2)
Sub-fields: N/A
Parent Field: d
Name in DB: N/A
Description:

Since the original age field is not verified by the cookie signatures, this field can be used to verify it's integrity. If both fields are the same then it can be assumed they have not been tampered with.

Cookie Name: g
Other Names: Member Global Unique Identifier
Sub-fields: N/A
Parent Field: d
Name in DB: mbr_guid
Description:

This value serves as the name implies as a GUID and is of course unique to each user. It's most notable use is in the profile system. When a logged in user alters their profile they are directed to the following page.

http://profiles.yahoo.com/u/<guid>

<guid> corresponds to the mbr_guid in the database for the current user.

The T Cookie

Cookie Name: sl
Other Names: sled ID
Sub-fields: N/A
Parent Field: d
Name in DB: sid
Description:

The sled id value in the cookie is formed by taking the sid value from the database and Y64 encoding it. This serves a similar purpose to the id field in that it helps to verify the user is who they claim to be.

Cookie Name: ok
Other Names: EmailVerified
Sub-fields: N/A
Parent Field: d
Name in DB: N/A
Description:

The presence of this field indicates the user has verified their email address and has the value "ZW0-" which is the Y64 encoded string "em". It is likely that the significance of "em" is that it is also the name of the email address field in the user database.

The T Cookie

Cookie Name: zz
Other Names: timestamp
Sub-fields: N/A
Parent Field: d
Name in DB: N/A
Description:

The first 6 characters of this field are a Y64 encoded 32 bit integer, representing the time the session was created as a unix timestamp. This timestamp is used as a reference to detect when a cookie has expired. The next 3 characters must be the string "A7E", the exact purpose is unknown but it too appears to be a Y64 encoded integer.

Cookie Name: tip
Other Names: T-Cookie IP
Sub-fields: N/A
Parent Field: d
Name in DB: N/A
Description:

The tip field is used to store an IPv4 address. It consists of a Y64 encoded 32 bit variable that stores the IP address in numeric form. This field can contain either the user's IP address or in some situations that of a Yahoo! server, this appears to be dependant on which server requested the user to login.

The Authentication Cookies

Once the correct authentication cookies have been constructed then they can be used to access the following services...

Mail
Profiles
Web Messenger
WAP Messenger
Games
Geocities
Web Hosting
360 Blog
Briefcase
Address Book
Calender
Notes

One other important thing to note is that some services do less checks than others, for example WAP and Geocities don't seem bothered when several important fields are incorrect.

The User Database

(common fields)

a	- List of aliases associated with account
abuse_sid	- Same as sid *(deactivated account only)*
abuse_ym	- Same as em *(deactivated account only)*
abuse_pw	- Same as pw *(deactivated account only)*
abuse_id	- Same as id *(deactivated account only)*
abuse_comments	- Reason for account deactivation *(deactivated account only)*
abuse_time	- Time of account deactivation as a unix timestamp *(deactivated account only)*
audit	- Log of recent account changes and secret question with answer
bd	- Birthday
co	- Country as 2 letter code
demog	- Contains name, title, address, zipcode
em	- Email address
gt	- Greeting Message
id	- Cookie ID
lg	- Users preferred language as 2 letter code
login	- Login name
mbr_commchannel	- Contains information used to contact user about account changes
mbr_guid	- GUID for user, used as 'g' in cookies
pc	- Postal / zip code
pw	- Password stored as a FreeBSD MD5 hash
pwqa	- Secret Question and Answer
reg	- Registration time
rip	- Registration IP
sid	- SLED ID, used as 'sl' in cookies
st	- State (us only)

The User Database

The user database is accessed by production servers for a variety of reasons. Most servers need to query the database in order to authenticate a user, others may use it to store or retrieve information that is relevant to the service they are providing. There are also some special servers that allow a user to start a session or change details of their account. These login and edit servers need a higher level of access to the database in order to read or write to the 'pw' field.

Having fewer servers being able to access the password field is a sensible precaution and should mitigate the risk to user accounts when a standard production server is compromised. It is however still possible to take control of most accounts with only this access.

An account can be taken control of in the following steps.

1) The mbr_commchannel is used to inform users of changes to their account but can easily be changed to deny the account owner a chance to recover it.

2) The account recovery feature can be used to change the password to anything we desire as long as we know enough information about an account.

3) Fill in the birthday, postal code, country and state for the account. These are all easily known with with regular database access.

*4) The 'pwqa' field storing the secret question and answer is not directly accessible without **FULL** database access but if the 'audit' field is present for a user then it often contains the secret question along with the answer. (definitely an escalation issue)*

5) Pick a new password.

The User Database

```
# Example of accessing the Yahoo! database using PERL with the ydbs
# package.  This code queries the login and id for a given username with
# read-only permission.
#
#!/home/y/bin/perl -w

use ydbs;

my $login;
my $id;
my $yid = "username";
my $u = new ydbUser();
my $acctid = new ydbsAccountID($yid);
my $rc = $u->open($acctid, ydbUser::ro, join("\001", qw(login id)));

die ydbEntity::getErrorString($rc) if $rc != ydbReturn::UDB_SUCCESS;

$login = $u->get("login");
$id    = $u->get("id");
$u->close();
```

Scope

If you have viewed the accompanying video then you will have seen how it was possible to gain access to several example accounts. This begs the question of who is affected and in what way.

Millions of regular users
Businesses
Web Hosts
Yahoo!'s own staff
Federal Agents
Government Officials

1 - The most obvious people affected are regular and business users.

2 - Many users link their domains or web hosting accounts to a Yahoo! Email, this not only effects their security but in many cases that of websites that share the same server.

3 - Yahoo!'s own staff make use of the regular mail service, especially customer care.

4 - A quick web search and recent events ('gov.palin') show that there are federal agents and government officials who make use of Yahoo! mail. Depending on what precautions each has taken this could pose a more serious threat.

Recommendations

This section is directed at Yahoo! staff.

1) The vulnerability used to gain access needs to be patched, but I won't discuss the details of that here.

2) YINST will correctly set restrictive file permissions on the 'tcookie.keydb' file but unfortunately there is still a way that low levels users can retrieve this file. If the package install data is left on the system then a user need only extract it from the corresponding 'installed.tgz' archive.

3) The 'audit' field seems to unnecessarily contain the 'pwqa' field, which in the event of database compromise acts as easy escalation. Perhaps it would be possible to remove it?

4) As mentioned earlier servers such as WAP, Geocities and possibly others do not use very strict cookie checks. It would be better if all servers enforced the same high standard.

Appendix (A)

(Rant)

Certain Yahoo! chat rooms are overrun with people who openly engage in fraud, they sell credit card details, online bank logins and a variety of other crap. By the time customer care could react the person can simply reappear on a new name.

"**uneed_help1:** Westernunion Bug 2009 Available for now with Activation code 250$ can send screenshots to prove i have it " - Exhibit A

Doesn't Yahoo! Have a duty to prevent this kind of illegal activity over it's network?

The problems do not end there. The system is also plagued by the constant presence of porn bots and booters. Two foes who should have been wiped from existence years ago but are still dominant due to the lack of development in the underlying chat infrastructure.

It is difficult to understand how Yahoo! could roll out WAP messenger, web messenger, MSN integration and regular messenger (with it's ever more elaborate features) whilst the infrastructure expected to carry it all is neglected. I believe when the issue has been raised in the past the response was to the effect of "chat is not a priority at this time".

Appendix (A)

(Rant)

On the current system the morons reign supreme they have their booters complete with sites hosting booter communities and there are actually people who buy and sell so called “illegal” Yahoo id's for money. You need only type “Yahoo booter” into a search engine to see how rife these disruptive people are. Perhaps the most annoying thing about all of this is that it need not be the case. Yahoo! could use their resources to remedy these problems but they choose not to.

I'd like to make some suggestions in the hope they are given fair consideration.

1) Enforce the YMSG protocol more strictly at the server-side, if a malformed packet or an unknown type arrives drop it, then drop the user sending it. While this may create more overhead on servers, the overall system performance will eventually benefit without huge volumes of disruptive malformed traffic.

2) Make registering for an account require more verification, currently people can sign up for large volumes of names very quickly. Without some form of prevention it is pointless to tackle the existing problems, bot supplies for example can easily be replenished.

3) Give somebody the power to identify and disable large numbers of names, users will willingly help battle the bot infestations if they know that they will be taken seriously. 20 bot names with the same prefix flooding a room could be dispatched in minutes with the correct mechanism.

Appendix (A)

(Rant)

4) Like a lot of people I do not like advertisements but I am willing to endure a reasonable amount of them for an otherwise free service. I believe billboard style adverts embedded into the CAPTCHA system could be beneficial for the following reasons.

* It would add much needed entropy to a system that is getting easier to beat with modern OCR software.

* Since CAPTCHA is a necessary component for all users even 3rd party client users would see them. I can safely say in one day a large number of CAPTCHAs will be seen by users. If the chat system becomes a renewed source of revenue for Yahoo! then there is more motivation to keep users happy and the system in working order. While i can't speak for everyone I believe this would be an acceptable trade off for a better chat system.

Appendix (B)

(Greetings)

Last but not least, greetings to...

Paradox
Warpboy
Timq
cool_mofo_2
Ice_Dragon

And of course to all the regulars of a certain room that shall remain nameless (you know who you are)

hl1