Annual ADFSL Conference on Digital Forensics, Security and Law

2011
Proceedings

May 25th, 2:00 PM

MAC OS X Forensics: Password Discovery

David Primeaux
Virginia Commonwealth University, Richmond, Virginia, dprimeau@vcu.edu

Robert Dahlberg
Virginia Commonwealth University, Richmond, Virginia, dahlbergra@vcu.edu

Kamnab Keo
Virginia Commonwealth University, Richmond, Virginia, kkeo@vcu.edu

Stephen Larson
Virginia Commonwealth University, Richmond, Virginia, stephen.larson@sru.edu

B. Pennell
Virginia Commonwealth University, Richmond, Virginia, pennellbd@vcu.edu

Presenter Information
David Primeaux, Robert Dahlberg, Kamnab Keo, Stephen Larson, B. Pennell, and K. Sherman

MAC OS X FORENSICS: PASSWORD DISCOVERY[1]

David Primeaux (dprimeau@vcu.edu),2
Robert Dahlberg (dahlbergra@vcu.edu),2
Kamnab Keo (kkeo@vcu.edu),3
Stephen Larson (larsonsp@vcu.edu),3
B. Pennell (pennellbd@vcu.edu),2
K. Sherman (shermankc@vcu.edu)2

Virginia Commonwealth University
401 West Main Street
P.O. Box 843019
Richmond, Virginia 23284-3019

ABSTRACT

OS X provides a password-rich environment in which passwords protect OS X resources and perhaps many other resources accessed through OS X. Every password an investigator discovers in an OS X environment has the potential for use in discovering other such passwords, and any discovered passwords may also be useful in other aspects of an investigation, not directly related to the OS X environment. This research advises the use of multiple attack vectors in approaching the password problem in an OS X system, including the more generally applicable non-OS X-specific techniques such as social engineering or well-known password cracking techniques such as *John the Ripper* or other versions of dictionary attacks and Rainbow table attacks. In some successful approaches the components of the attack vector will use more OS X specific techniques such as those described here: application-provided password revealing functions, a Javascript attack, an "Evil Website" attack, system file scavenging, exploitation of the keychain, and an OS X install disk attack.

Keywords: OS X, password, password discovery, social engineering, sleepimage, keychain,

1. BACKGROUND

Passwords are of forensic value because while they may be helpful in protecting the interests of computing users who have benign intent, passwords can also obstruct a forensic investigation. A solution to this *password problem*, from the point of view of the forensic investigator, is the discovery of the password (or set of passwords) obstructing an investigation. Password discovery involves locating the password and, as necessary, decrypting that password.

There is no simple, direct solution to all instances of the password problem in OS X. However, in a forensic investigation of an OS X system, the investigator may benefit from the facts that he or she can make use of standard (that is, non-OS X-specific) password attack techniques, and that the typical user may have little or no knowledge of, or control over, the location of some passwords used in the system. Furthermore, when the investigator cannot readily locate a potentially useful password, or when such a password is located but encrypted or obscured (often by means of a password hash function), the investigator may benefit from understanding typical human behaviors that often affect

[1] This research is partially funded by ManTech.

[2] School of Engineering, Computer Science Department.

[3] School of Business, Information Systems Department.

[3] School of Business, Information Systems Department.

password use.

Independent research conducted by Sophos Labs revealed that 33% of the respondents used the same password for every website and 48% used a few different passwords for every web site [1]. Therefore, if an investigator is able to discover one password, it is highly likely that the investigator will see this password again, re-employed by the user for some other purpose. Another survey conducted by Sophos targeted 500 business PC users and revealed that 72% of the respondents used weak (easily discovered) passwords. The respondents had a tendency to use passwords such as their girlfriend's name, favorite football team, or pet's name [2].

Some salient information (such as a girlfriend's name) related to passwords formed in this manner may be known through other aspects of an investigation such as the questioning of witnesses, or through use of known social engineering techniques; and, with increasing likelihood, the user may have published this information on public social networking sites such as *Facebook* and *MySpace*.

We also expect that several passwords for the same user will show a tendency to have similar characteristics with regard to meaning-to-the-user, length and complexity. Knowing this type of behavior is useful because it gives the investigator insight into trends related to the user's password management. For example, if the investigator discovers two or three passwords that are derived from animal names, when mounting a dictionary attack on an encrypted password that investigator may benefit from loading a dictionary list that contained variations of animal names.

Additional information is available regarding the impact of human and social behaviors on password formation. Recently, for example, phpbb.com was hacked and the passwords of 20,000 users were published. Robert Graham wrote an application to analyze these passwords and found the following trends [3]:

16% of passwords matched a person's first name.
14% of passwords were patterns on the keyboard, such as 1234 or qwerty or asdf
4% were variations of the word password such as passw0rd or password1
5% were passwords referenced to pop culture, such as pokemon, ironman
4% appear to be references of things nearby, such as Samsung, Packard or apple
3% were swear words, the F-word was very popular
3% were "don't care" words like whatever or blahblah
1% were sports related, team names or sports

Table 1 gives the rate of use for the top 20 passwords for these 20,000 users.

Percentage of use followed by password			
3.03% "123456"	0.59% "12345678"	0.36% "trustno1"	0.30% "hello"
2.13% "password"	0.58% "letmein"	0.33% "dragon"	0.30% "monkey"
1.45% "phpbb"	0.53% "1234"	0.31% "abc123"	0.28% "master"
0.91% "qwerty"	0.50% "test"	0.31% "123456789"	0.22% "killer"
0.82% "12345"	0.43% "123"	0.31% "111111"	0.22% "123123"

Table 1. Percentage of shared password use in 20,000 phpbb.com users.

Note that for each password shown, between 660 users (3.03%) and 44 users (.22%) used the same password.

The passwords shown in Table 1 may demonstrate the potential value of a social engineering

perspective in the search for user passwords. Using a social engineering perspective, the investigator would take into account a particular work environment or hobby or other sort of personal interest known to be associated with the subject of the investigation. In the case of these users, for example, some of the commonly used passwords may be indicative of the sorts of users attracted to phpbb.com. Among such passwords are: "trustno1", "letmein", "dragon", "master", and "killer". For other sorts of investigation, on the other hand, it may be interesting to use a "reverse" social engineering attack: if the passwords can be hacked for a given population of users, some passwords may directly or indirectly reveal information useful to the investigation.

A social engineering password attack may not provide a directly usable password, but may still be helpful to the investigator with access to some other means of attack. In this regard, information stored in a user's browser history or bookmarks may be helpful as a means of advancing a social engineering attack, as may discoveries related to the user's behaviors on his or her frequently visited websites. For instance, if a user has visited a gaming site and has a character on that site – the name of that character may provide some insight into the patterns the user has employed in constructing passwords. For example, if a social engineering attack were to reveal that the user often uses a pattern of password structure that contains, in part, the name of an animal (*lionxyz*, for example, or *yrtmonkey*), this knowledge may prove useful in decreasing the time required to mount a successful dictionary attack, described below.

In some investigations, it is possible that known secure applications or remote systems, accessed by the user, have password strength restrictions such as a requirement that a password contain a combination of characters and numbers. An investigator mounting a social engineering attack in this environment should look for numbers that may be significant to the user, such as birthdays, graduation years, anniversaries, and so forth. Or a password constraint may require the password be changed every 30 days. In these cases, it is likely that the user's password will follow some pattern such as some combination of a base structure modified by some addition reflecting a month (for example, Betty01 for January's password, Betty02 for February's password).

A dictionary attack is a non-OS X-specific password discovery method applicable, as are other standard techniques to OS X password discovery. A dictionary attack is a technique for defeating a cipher or authentication mechanism by trying to determine its decryption key or passphrase by searching likely possibilities. A dictionary attack uses a modified brute-force technique of successively trying all the strings in an extensive list of strings. We say the technique is modified brute-force, because a true brute force attack (not further discussed in this paper) would search all possible combinations of characters, whereas a dictionary attack typically limits its list to variations of strings that have some meaning. Such attacks often succeed because many people have a tendency to choose passwords that are short, single words found in a dictionary or simple words appended with a numerical character. In the example provided in the paragraph above, the examiner might benefit from trying a dictionary loaded with strings that represent variations of animal names.

Another standard password-related attack is a Rainbow table attack. A Rainbow table is a lookup table that may sometimes prove useful to the investigator trying to recover a cleartext password from a password hash. A hash is a mathematical technique that takes one input and maps it to another in such a way that it is often very difficult to determine the initial input from the result of applying the hash function. A password hash is the result of inputting a password into a hash function. As we would expect, for security reasons, OS X stores only the password hash (and not the cleartext password itself). We would also expect the selection of an appropriate hash function so that determining the password itself from the password hash would be challenging. The Rainbow attack is similar to a dictionary attack, except the list it uses contains the hashes (or portions of the hashes) generated by introducing possible password into a known hash function, such as OS X's password hash function. If the resulting hash is found to match a password hash, the Rainbow attack can use a simple lookup table to find the password that generated the hash.

2. PASSWORDS

In this section, we outline what can be password protected in OS X, and what are its possible locations and forms for stored passwords.

In addition to providing a means by which a user can password protect individual files and folders, OS X employs password protection (or permits the use of password protection) within several contexts including user login, privileged access as root, *FileVault*, keychain (a means of creating and managing passwords, typically accessed through the *Keychain Access* utility), networking and webpage logins, and other applications.

Password-related information is found in locations associated with utilities such as *Keychain Access* and with applications such as the Firefox browser, as well as in the OS X user-specific password hash file /var/db/shadow/hash (the user-specificity of the file in OS X contrasts with the more usual situation in other UNIX-like or UNIX-related systems in which a single such file holds password hashes for all users). Generally, the actual location of a password is controlled by the operating system, by an operating system related utility, or by a third party application. However, it may be useful for the investigator to be aware that requirements for construction of more complex passwords increases the likelihood of the user having recorded clear text copies of passwords (or of mnemonics associated with them); these user copies of passwords may be stored on the system being investigated (for example on a *Stickies* note), or elsewhere (such as a scrap of paper attached to the system monitor). In any case, as will be discussed below, some passwords in an OS X system will be stored in encrypted form, while others may be stored in a non-encrypted ("clear text") form.

2.1 Additional technologies to assist in OS X password discovery

Given the password characteristics just described, a successful methodology for password discovery is likely both to employ multiple techniques and to leverage any resulting positive results. Several available technologies can be useful in the process of password discovery. These include password-cracking utilities (such as the well-known *John the Ripper*), the use of native password revealing functions in applications that store passwords, the OS X password reset tool, and java scripts. The generally applicable password cracking approaches that use techniques such as dictionary attacks and Rainbow Table attacks should be well known and are very briefly described in Appendix 5; other technologies, perhaps not so widely known, are discussed here. These include: application-provided password revealing functions, a Javascript attack, an "Evil Website" attack, system file scavenging, *Keychain Access* attack, and an OS X install disk attack.

Application-provided password revealing functions. Because human maintenance of passwords is becoming increasingly burdensome, it is expected that many users may elect to have passwords remembered for them. The *Keychain Access* utility in OS X has a *show password function* that has the capability of showing many system-remembered user passwords (and sometimes also relevant, of showing many user usernames). Although use of the password revealing function in *Keychain Access* does require access to the user's keychain password, if that password can be discovered it will permit the investigator to access to all passwords stored in *Keychain* (An "Always allow" authorization option existing in the password revealing function may give the impression that the user could unintentionally provide access to this function for all time; however, our tests show that this is not the case). *Keychain Access* is further discussed later in this section.

Passwords may be more readily available to the investigator through the password revealing function available in at least some versions of the Firefox browser (such as version 3.6). In order to have clear text access to any passwords the user has asked to be stored for browser use, the investigator need only select the following sequence of options once Firefox has started: <Firefox>, <Preferences>, <Security>, <Saved Passwords>, <Show Passwords>. The investigator will be asked whether to

confirm the <Show Passwords> selection and will be permitted to do so without further authentication. A user may prevent the <Show Passwords> option from working as described, in the following manner. For increased security, Firefox permits the user to set up a master password within the <Security> option. If this is done, authentication with that master password will be required to use the <Show Passwords> function. We hypothesize that a high percentage of Firefox users are unaware of this security precaution and do not implement a master password for Firefox. This exploit, and the possible use of a master password to thwart it, is available in some form on at least some other Mozilla 5.0 derived browsers such as SeaMonkey 2.0.4.

Java script attack. While this Java script attack has been tested on FireFox, Internet Explorer, Safari, and Chrome, it has not been tested in all browsers; nor, of course, is it feasible to test its usefulness for every website. This attack reveals the password used as credentials in a website [4]. When this exploit is effective, pasting the following java script into the address bar of the webpage will cause the password to be revealed:

```
(javascript:(function(){var s,F,j,f,i; s = ""; F = document.forms; for(j=0;
j<F.length; ++j) { f = F[j]; for (i=0; i<f.length; ++i) { if
(f[i].type.toLowerCase() == "password") s += f[i].value + "\n"; } } if (s)
alert("Passwords in forms on this page:\n\n" + s); else alert("There are no
passwords in forms on this page.");})();"
```

Evil Website attack. This investigative attack is a modification of the "black hat" technique called the "Man in the Middle Attack." The Evil Website attack is only outlined here, and requires these three elements:

- A forensic copy of the suspect's hardrive that the investigator will use to boot a system and access webpages. For simplification we will call this the *suspect's system*.
- The suspect's system has passwords stored for browser use either through OS X keychain or through the browser's built in password management tool.
- A web server hosting the Evil Website which is actually a dummy forensic website to which the suspect's browser will be made to send passwords.

The basis of this attack is that a browser running on the suspect's system will pass credentials, otherwise hidden from the investigator, in clear text to the Evil Website. The Evil Website web server will be used to capture the user's username and password.

System file scavenging. As determined in related research [5], it is possible to force the content of *active* (and possibly of *inactive*) physical RAM to an OS X *sleepimage* file. Furthermore, information in both *active* and *inactive* virtual memory is stored in the OS X *swapfiles*. The investigator can use a standard hex editor on a forensic copy of these files to scavenge for passwords, either encrypted or in clear text. Our preliminary tests show, for example, that the user system password can be found in clear text in the *sleepimage* file. This specific fact may, however, simply be an artifact of undesirable memory management, but more exploration of this and related issues is required. Nevertheless, this result supports the notion that passwords will be stored at least occasionally in the OS X *sleepimage* and *swapfiles.*

Keychain Access attack. The forensic investigator should become very familiar with the OS X *Keychain Access* utility. Many password-requiring applications that run in OS X are designed to be

"keychain aware." This means that such applications can use *Keychain Access* to manage their credentials. As mentioned above in discussion of password revealing functions, *Keychain Access* does require the user to authenticate in order to reveal in plain text stored passwords. But by default OS X sets the user's required *Keychain Access* password to his or her system password. This design decision may be a direct reflection of Apple's desire to have Macs and the OS X operating system perceived as easy to use and requiring little user intervention to perform daily computing task. But, as seen here, it also can be a cause of concern with respect to their security, especially given the power provided by knowledge of the *Keychain Access* password; the forensic investigator should seek to exploit that security concern to the fullest extent possible. In particular, if the investigator can acquire a suspect's system password, it is highly likely that he or she will be able to use this utility will unlock many additional doors.

As expected, the OS X user does have the more secure option of changing the *Keychain Access* password but seems to be discouraged from doing so by information provided through the *Keychain Access* <Help>:

> You can change the password for your keychain at any time. However, if you want your default keychain to be unlocked automatically when you log in, make sure your keychain password is the same as your Mac OS X login password for your account.
>
> If your Mac OS X login password is not the same as your default keychain password, you'll be asked for the password whenever an application needs access to your keychain and your keychain is locked [6].

For a number of reasons, including: the user not wanting to be asked for a password each time a "keychain aware" applications needs access to the keychain; the default setting for the *Keychain Access* password as the user's system password; the perception that Apple provides a more secure computing environment than Windows does; and the expected relative obscurity of *Keychain Access* to many users --- we are confident that many OS X users will keep their *Keychain Access* password the same as their login password.

OS X install disk attack. The OS X install DVD comes with a built in utility that permits a user to reset nearly any password on the system. Resetting a user's OS X password will not reset the user's *Keychain Access* password or FileVault password. At this time we have not discovered an alternative means of resetting the *Keychain Access* password except through direct use of the *Keychain Access* utility itself, or an alternative means of resetting the FileVault password except through direct use of the OS X *System Preferences* utility. Future research will be required to determine whether some other method, such as the use of a command line interface like that provided by the OS X security to directly access the keychain, might be useful in further exploiting the keychain or breaking into a FileVault-protected system.

In OS X, "root", as in other UNIX-like or UNIX-related systems, is the name the most highly privileged system user, capable of reading, writing, deleting, moving, or otherwise accessing any file or folder in any account on the system. This makes "running as root" a particularly dangerous way to work within such a system. Because of the dangers of running as root, OS X makes access to root privileges a little less obvious than do some other UNIX-like or UNIX-related systems. In particular, by default, the root user is not enabled in OS X. However, by using the OS X install DVD the investigator can enable the root user and/or change the root user's password. This may prove significant in an investigation because it may also lead to information about the password for a suspect's account: in OS X, when root attempts to change another account's password, the password hint associated with that account is revealed. This can give the investigator insight as to what the account's password might be. If the password hint should be something like "high school attended" or "favorite pet's name," then it might be relatively easy to determine that information.

3. CONCLUSIONS AND FUTURE RESEARCH

OS X provides a password-rich environment in which passwords protect OS X resources and perhaps many other resources accessed through OS X. Every password an investigator discovers in an OS X environment has the potential for use in discovering other such passwords. Additionally, the investigator will be aware that these discovered passwords may also be useful in other aspects of an investigation, not directly related to the OS X environment.

There is no direct, unique solution to the password problem in OS X. An effective approach to the problem may require the use of multiple attack vectors. In some successful approaches the components of the attack vector will include the more generally applicable techniques such as social engineering or well-known password cracking techniques such as *John the Ripper* or other versions of dictionary attacks and Rainbow table attacks. And in some successful approaches the components of the attack vector will use more OS X specific techniques. These include: application-provided password revealing functions, a Javascript attack, an "Evil Website" attack, system file scavenging, exploitation of the keychain, and an OS X install disk attack

Our future password discovery research will focus in three directions. We will explore the command line OS X security interface to determine the extent to which this interface can used to provide additional information about the keychain, as well as to provide methods of manipulating the keychain with the potential of permit additional system access. We will also continue our exploration of the OS X *swapfiles* and *sleepimage* file with the specific goals of determining which if any passwords may be routinely stored in these files and whether the location of any such passwords can be predicted. Finally, we intend to explore the impact of enabling Secure Virtual Memory on OS X password discovery.

REFERENCES

1. Cluley, Graham (2010) Graham Cluley's Blog : "Do you use the same password for every website?", http://www.sophos.com/blogs/gc/g/2009/03/10/password-website/, accessed 26 March 2010.
2. Sophos (2010) "Employee password choices put business data at risk, Sophos poll reveals." http://www.sophos.com/pressoffice/news/articles/2006/04/passpoll06.html, accessed 26 March 2010.
3. Graham, Robert (2009) "PHPBB Password Analyses", http://www.darkreading.com/blog/archives/2009/02/phpbb_password.html, accessed 26 March 2010.
4. Davis, Sam (2008) "Reveal a Password in Internet Explorer or Firefox", http://www.blastedthing.com/misc/mag-reveal-a-password-in-internet-explorer-or-firefox/ , accessed 19 April 2010.
5. Primeaux, D.; Dahlberg, R.; Keo, K., Larson, S.; Pennell, B.; and Sherman, K. (2010) "MAC OS X Forensics: Volatile Memory Acquisition." Technical Report VCU-CISS-TR-10-1, Virginia Commonwealth University, Computer Science Department.
6. Apple Inc. (2010) "Mac OS X 10.4 Help, Changing your keychain password" http://docs.info.apple.com/article.html?path=Mac/10.4/en/mh463.html, accessed 23 September 2010.