"Using An Enhanced Dictionary to Facilitate Auditing Techniques Related to Brute Force SSH and FTP Attacks"

Ryan McDougall
Computer Networking Department
St. Cloud State University
Saint Cloud, MN 56301
E-mail: mcry0802@stcloudstate.edu

Nicole Gillespie
English Department
St. Cloud State University
Saint Cloud, MN 56301
E-mail: gini0604@stcloudstate.edu

Dr. Dennis Guster
IS Department
St. Cloud State University
Saint Cloud, MN 56301
E-mail: dguster@stcloudstate.edu

Abstract

This paper analyzes the efficiency of security auditing on network user accounts. More specifically, the paper focuses on the drawbacks of traditional network account username and password creation, namely "dictionaries." Most auditing methods will employ dictionaries, which in this case are lists of common user names and passwords, and use them to try and gain access by guessing the account credentials. When these dictionaries are created, they are based on a criterion that weakens passwords. As an alternative, the authors consider a new way to create a network dictionary, focusing on two important elements: default user names and common user name schemes. A user name dictionary emphasizing user name accuracy based on common schemes, used in conjunction with a large password dictionary, reduces the chances of a security breach. The security put in place creates account lockouts, banning the IP addresses of attackers. This auditing method lowers the effectiveness of attacks.

1 Introduction

With computer crime becoming an increasing concern every year, the need for security services grows as well. To put this into perspective, an annual study conducted by IC3 [14] shows in 2010 alone, there were 303,809 complaints received via their web services. A steady increase is displayed in these particular crimes since the year 2000, which had 16,838 complaints. With that in mind, security professionals in the industry know they have to be proactive when it comes to the safety of the users and systems they protect. One way to achieve this is through security auditing. This paper will focus on account auditing, which is only a small portion of things a security professional can audit, in the context of using a brute force method to facilitate attacks over network protocols such as SSH or FTP.

2 Literature Review

There certainly is no doubt that unauthorized attacks on information systems continue to grow. In fact, B. Keefe revealed a survey by the Computer Crime Research Organization, which reports that hacker attacks grew 37% in the first quarter of 2004 [8]. Of course, there are many different types of attack methods used, but the most dangerous is related to compromising passwords that secure user accounts. Over the years, the Computer Emergency Response Team (CERT) has issued numerous alerts about the methodologies hackers use, and asserts one of the core vulnerabilities is a host's password file [4]. The methodology used certainly is dynamic in nature, but the recent advent of distributed processing has changed the landscape; the attacks may come from multiple processors. Although CERT suggests numerous ways to protect the password file, it is still vulnerable and one cannot assume that it might not be compromised. Furthermore, one should not assume an encrypted password unable to be compromised by a hacker when it travels across the network. It is therefore critical that if data integrity is to be maintained, a defense/monitoring strategy should be implemented.

A common OOPs (object oriented programming) principle is encapsulation. Hence, a form of encapsulation, encryption is used as the primary defensive measure to mask information from hackers. This form of encapsulation makes it more difficult for hackers to attack computer systems, but is not without fault, particularly since hackers started to effectively use distributed computing as part of their attack scenario. Sound policy is critical in blocking the hackers at the gates (which is typically a firewall). Policy implementation is often complicated, because end users tend to want passwords that are easily remembered [5]. Often this results in a conflict between ease of memorization and resulting vulnerabilities. The resulting conflict is much easier to understand if the common methods used to break encrypted passwords are analyzed.

One such methodology, the brute force attack is based on trying every possible combination of characters. The assumption is that given enough time, the correct combination of characters will be determined. The problem with this method is it may take a huge number of iterations to determine the correct one. This methodology does not

rely on any social engineering principles to reduce calculation time. Therefore, within this scenario, assuming adequate length of the password regardless of its content, it may be effectively unbreakable (at least within a reasonable time frame). In an effort to speed things up, hackers have adopted tools that have reduced the randomization principle by focusing on dictionary attacks.

According to Arbelaez [1], in most cases, a dictionary attack is the most efficient way to retrieve a lost password. Specifically, dictionary attacks use a list of words (a "dictionary"), and then verify each word as a possible password. He further asserts most successful dictionary attacks can use custom-made dictionaries. These custom dictionaries include every word that could be meaningful to the person who encrypted the document – name, birth date, spouse's name, company's name, favorite team's name, and so on. This dictionary method fosters the use of social engineering as part of the password breaking strategy, and quite often it is the added speed this method provides that allows hackers to break passwords in an effective time frame. While the dictionary method and social engineering can speed the process up, Arbelaez [1] recognizes that breaking passwords is still a daunting task, and if it is to be accomplished in a timely manner, it will require the benefits of distributed processing.

Parallel processing evolved from supercomputers and was not made available to the masses at first. However, in the last ten years numerous effective parallel processing applications were implemented on distributed clusters using standard PCs [12]. This clustering method has been used with great success in breaking passwords. In fact, Business Wire [2] reported in 2000 that at Comdex in Las Vegas, Turbolinux Inc. demonstrated the potential of distributed processing by "cracking" five character NT passwords in less than a minute, using an application freely available on the Internet. Just like any form of technology, new attack methods are constantly evolving, which means that new defensive methods also need to be updated accordingly. For example, when the Data Encryption Standard (DES) family of algorithms was first devised, it took 96 days using trial and error encryption busting programs to break DES encrypted code [9]. However, just three years later using the same "brute force" logic, it was broken in 22 hours [11]. The significant performance improvement can be directly tied to the use of distributed processing. In order to realize this vast improvement, a specially designed supercomputer front end and 100,000 PCs directly connected to the Internet were used to provide processing power. This example clearly defines the vulnerability that can result when distributed processing is harnessed in password breaking schemes. Therefore, it is clear that security personnel not only need to be able to defeat attacks coming from a hacker's single computer, but also attacks coming from an array of distributed logically linked computers [10, 13].

Such a degree of sophistication on the part of the hackers behooves security personnel to proactively prevent successful attacks. To do so, they must be prepared to periodically test the integrity of their password files by applying commonly used attack methods to their systems. Of course, the password file on any host is a very popular target of hackers. If a hacker is able to obtain a copy of this file, it can be expected they will use tools that employ a "brute force" attack to determine if there are any weak passwords within the

file. A list of well-known weak passwords is readily available as part of the dictionaries, and they can often be compromised from their encapsulated form in just seconds. A few examples of passwords considered weak includes *letmein*, *password*, a password the same as the account name, or, in reality, any word in the dictionary. There are also variants, such as p@ssw0rd, that an end user might think clever yet is still an easy break for a hacker because they are in the hacking dictionary.

For enlightened system administrator/security officers, it is common to apply the tools used by hackers to ascertain if weak passwords are present in their password files. If so identified, then they have a chance to proactively modify the passwords long before any damage results. Furthermore, due to the availability of distributed computing, it is crucial that this proactive approach be applied not just to weak passwords, but all passwords. In fact, any information using many of today's common algorithms are in danger of being compromised in less than a day. Specifically, this need has been recognized by Castillo [3] who states, "many system administrators actively run password hacking routines on high powered servers to attempt to break weak passwords thus minimizing the probability of hackers breaking into under secured accounts."

Given this scenario, the question then becomes, "What is the most effective and practical way to employ distributed computing to gain a competitive time advantage over hackers." This paper attempts to expand on a model devised by Guster, Safonov, Hall and Podkorytov [6], which is based on a commonly used password hacking tool. This tool was modified for distributed processing. Coined John the Ripper [7], this tool has been deployed for a number of years and is widely used by hackers. The model [6] proved quite effective in terms of speed of breaking and number of passwords broken. In one test in particular, the model reduced the time to break selected components from a password file from one year on a single processor down to just 11 days on 32 processors.

3 Model Description

When considering an account audit, one needs to consider the components of an account, namely the user name and the password. Generally a user name is going to identify the user or group of users that are going to access services via the account. The password is a basic, but essential, security control put in place to protect the account and entire network from potentially being compromised. Most auditing methods will employ dictionaries, which in this case are lists of common user names and passwords, and use them to try and gain access by guessing the account credentials. When these dictionaries are created, they are based on a criterion that weakens passwords. These criteria, as suggested by a Wikipedia article [16] on password strength, include:

- Default passwords provided by system vendors: e.g password, default, admin...
- Dictionary words: e.g common words without special characters that can be guessed easily
- Simple obfuscation: Words with very little special characters ex.

p@ssword
- Words with numbers appended: ex. password1234
- Doubled words: ex. passwordpassword
- Common keyboard sequences: ex. qwerty, qwerty1234
- Identifiers(User names): ex. jsmith
- Identifying information: license plate numbers, phone numbers, birth dates etc.

While keeping these criteria in mind, we need to consider what the typical user might choose for a password. In a study done by Imperva [15], a data security firm, in which 32 million passwords were analyzed after a security breach at Rockyou.com, a list was compiled of the top 10 most commonly used passwords breached. The passwords are as follows: *123456, 12345, 123456789, password, iloveyou, princess, rockyou, 1234567, 12345678, abc123*. Considering the enormity of the data set these were pulled from, it is safe to assume that these types of passwords are typically used for accounts in other places as well. Considering this assumption, when researching the methods used for creating dictionaries in account auditing methods, there was a tendency for more emphasis on password accuracy. Yet, with common passwords most likely being short and simple as found in the Imperva study, there should be more emphasis on user name dictionary accuracy instead.

3.1 Dictionary Creation

To achieve a more accurate dictionary of user names, we need to consider a couple of things: default user names and common user name schemes. These are important, because they will highly contribute to the accuracy of our dictionary. When considering default user names, it is best to consider default system vendor user names. These usually consist of items such as, *Admin*, *Administrator*, *default*, or the name of the device or company. For example, some Cisco networking devices have their default user name for console access as Cisco. Also important are the common user name schemes. These would be the common user names one would see at a workplace, school, or even for home/private use. For example, if a person named John Smith worked at a company where he was assigned a user name to access services he needed to complete his daily tasks, he might be given the user name “jsmith.” To effectively combine these into a dictionary list, it is important to come up with a list of commonly used schemes. For example, a few of these may include: first Letter of first name, followed by the last name (jsmith); first two letters of the last name, and first two letters of the first name with numbers appended (smjo1001); and so on.

A method to compile these used both in the generation of a password and user name lists, is to write a computer program that will manipulate lists of common human names to make various user names based on some common schemes, with default user names included. The method of choice used in this paper is just that, a program written in the C++ language. This program takes advantage of the file streaming capabilities in C++ to manipulate two list files, one of common first names and one of common last names.

From there it will piece together the components of the user names and output lists that can be used with various utilities facilitating account auditing. After creating a program like this, it is very simple to add, remove, or change different schemes to accommodate for the user names that might be needed for a particular dictionary list. Here is an example of a list generated with one of the schemes:

This list contains a part of the first name A section with common last names.
Ajohnson
Awilliams
Ajones
Abrown
Adavis
Amiller
Awilson
Amoore
Ataylor
Aanderson
Athomas
Ajackson
Awhite
Aharris
.
.
.
Arogers

3.2 Dictionary Attack Statistics and Application

A dictionary attack, also known as a brute force attack, is a method of repeatedly guessing credentials to a given account to gain access to it. In our case, this method is used to proactively audit accounts for security purposes. The ideal way to carry out an attack like this is to use a utility program, which will read dictionary lists and attempt to ascertain the credentials in those lists. Dictionary attacks can be attempted on just about any protocol or service.

Looking back at the RockYou.com incident, it is important to pay attention to the effectiveness of the dictionary attack that had taken place. Information gathered from the report produced by Imperva, showed that using the top 5,000 passwords would only take one attempt per account to guess 0.9% of the passwords, at a success rate of one success per 111 attempts. From there, the number of successes grew exponentially. At that rate, on a DSL connection with a 55kbps upload rate, the attacker could successfully gain access to one account every second, and could reach 1000 accounts in less than 17 minutes. At this point, it would only take 116 attempts to gain access to 5% of the total accounts. The following is a graph depicting this.

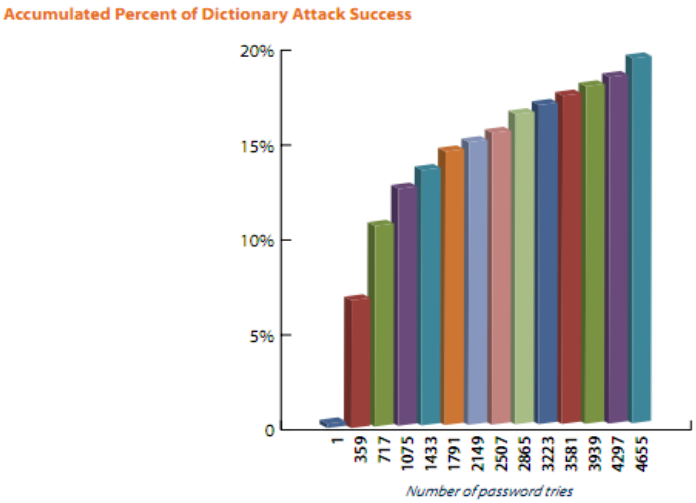

Figure 1

One way to increase the effectiveness of an attack is to use a distributed method while "chunking" the dictionary. A distributed method would require one to use multiple attackers at the same time. Most commonly, in a malicious environment one might see a distributed attack carried out by what is referred to as a "botnet." A botnet is a network of infected computers controlled by software, most times some sort of malware, which can be potentially used to carry out these attacks. When an attack is implemented, an attacker will be given a user name and parameter set, which also can be referred to as a group of passwords, to attempt on multiple victims at one time. Since each attacker is assigned a group of passwords, they can each try their sets of passwords with the same user name. This process can be repeated with each user name in a dictionary and password group. To illustrate this, Figure 2 is provided below.

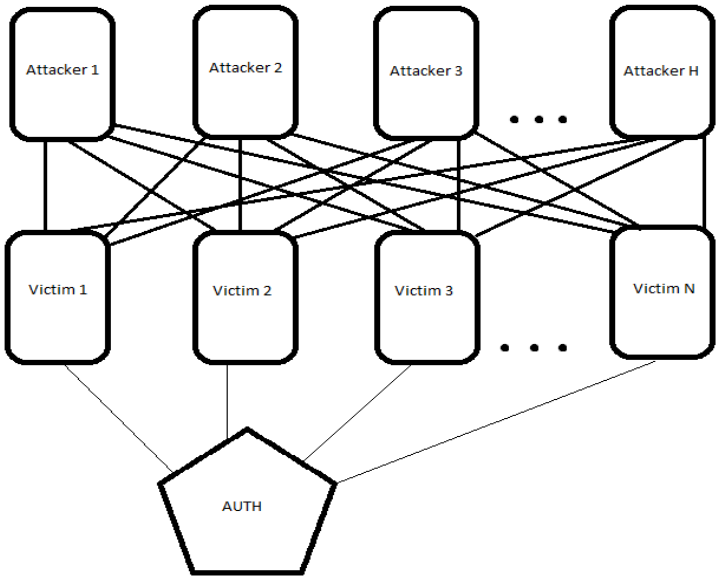

Figure 2: Applying Distributed Processing in the Attack Scenario

Figure 2 above shows how the "victims" are all tied back to the item AUTH. This is an authentication system, which is where the user names and passwords are checked to see if they are valid. In most cases, this will be some sort of directory service such as LDAP (Lightweight Directory Access Protocol) or Windows Active Directory. When attacking a group of victims, usually they are configured in this fashion, especially in scenarios in which SSH or FTP protocols are being used.

Doing an audit in this way has great advantages. If a user name dictionary that emphasizes user name accuracy based on common schemes is used in conjunction with a large password dictionary, the chances of avoiding security controls are reduced. These controls typically are put in place to create account lockouts or ban the IP addresses of attackers. This auditing method lowers the effectiveness of attacks as compared to allocating all resources to one victim. This is because the IP addresses of the attackers will differ, as well as the fact that the user names tend to be more accurate. In fact, the user name may seem to be an attempt from a legitimate source, and therefore appear transparent to the authentication system and any other security controls.

References

[1] Arbelaez, R. (2003). Advanced Office XP Password Recovery v. 2.30. http://www.itsecurity.com/itsecpep/3.

[2] Business Wire (2000). Turbolinux EnFuzion Cracks NT Passwords in a Minute at Comdex, http://www.businesswire.com/webbox/bw.111300/203180468.htm.

[3] Castillo, A. (2002). Research Proposal for Parallel Password Hacker, Computer Science Technical Paper, University of South Carolina, 2002.

[4] CERT. (2002) http://www.cert.org/tech_tips/passwd_file_protection.html.

[5] Conklin, A., Dietrich, G. & Walz, D. (2004). Password-Based Authentication: A System Perspective, *Proceedings of the 37th Hawaii International Conference on System Sciences*.

[6] Guster, D., Safonov, P. Hall, C. & Podkorytov, D. (2004). Business Computer Information Systems Security Against Hacking Attacks: Application of Distributed Processing and Software Modifiers in Defense of Password Files, *Proceeding of the Academy of Business Administration's National Conference*, Las Vegas, NV.

[7] John the Ripper Password Cracker. (2003) http://www.openwall.com/john/.

[8] Keefe, B. (2004). Computer Crime Research Organization News, http://www.crimeresearch.org/news/2003/04/Mess0902.html.

[9] Oakes, C. (1998). RSA: Crack DES in a Day, *Wired News*, December 22.

[10] Pfleeger, C. & Pfleeger, S. (2003) *Security in Computing*, Upper Saddle River, NJ: Prentice-Hall.

[11] RSA Code-Breaking Contest Again Won by Distributed.NET and EFF. (1999).

http://www.eff.org/Privacy/Crypto_misc/DEScracker/HTML19990119_deschallenge3.html.
[12] Rolfe, T. (1994). PVM: An Affordable Parallel Processing Environment, *27th Annual Small College Computing Symposium* (SCCS, 1994), 118-125.
[13] TECS Intelligence Papers: Cracking DES, (1998). http://www.itsecurity.com/papers/c rackdes1.htm.
[14] IC3: Internet Crime Complaint Center, (2010). http://www.ic3.gov/default.aspx.
[15] Imperva, (2011). http://www.imperva.com/docs/WP_Consumer_Password_Worst_Practices.pdf.
[16] Wikipedia , (2011). http://en.wikipedia.org/wiki/Password_strength.