https://doi.org/10.48009/2_iis_2006_33-37

PROCESSOR TYPE AND ITS RELATIONSHIP TO PERFORMANCE IN THE APPLICATION OF DISTRIBUTED PROCESSING TO DETERMINE VULNERABILITES IN PASSWORD FILES

Paul Safonov, St. Cloud State University, MN, USA, safonov@stcloudstate.edu
Dennis Guster, St. Cloud State University, MN, USA, guster@stcloudstate.edu
Renat Sultanov, St. Cloud State University, MN, USA, sultanov@stcloudstate.edu
Dmitri Podkorytov, Kurgan State University, Russia, podkorytov@mail.ru

ABSTRACT

The recent explosion in hacking has necessitated that security managers understand how hackers are able to break passwords and operationally have superior versions of the tools used by hackers so that they can run proactive tests on the vulnerability of their resources. A resource that is especially vulnerable is the password file even though it is encrypted. This paper builds on a modular approach to offer improvements to a commonly used password attack tool called "John the Ripper" [7] (hereinafter called "John") and evaluates its performance on three different types of processors: Sparc II, Intel and Sparc III. The results revealed that the Sparc III offers 3 to 4 times the performance of a Sparc II and that an Intel processor with a similar clock speed running assembler was close in performance to a Sparc III. Through distributed processing and improved software logic, we were able to reduce time to solve and increase number of passwords guessed over the "John" program running on a single machine. These results, if implemented properly, could offer security managers a competitive advantage over hackers in identifying vulnerable passwords quicker than the hackers can exploit them.

Keywords: Computer Security, Distributed Processing, Password Protection, Encryption, De-Encryption, Security Management

PURPOSE

Unauthorized attacks on information systems continue to grow. In fact, the Computer Crime Research Organization reports that hacker attacks grew 37% in the first quarter of 2004 [8]. There are many different type of attack methods used, but perhaps one of the most dangerous has to do with compromising passwords that secure user accounts The Computer Emergency Response Team (CERT) has issued numerous alerts about the methodologies hackers use and stated that one of the core vulnerabilities is a host's password file [4]. Although CERT suggests numerous ways to protect the file, one cannot assume that it might not be compromised or an encrypted password will not be viewed by a hacker when sent across a network Therefore, if data integrity is to be maintained, a defense/monitoring strategy needs to be implemented.

The primary defensive measure used to mask information from hackers is encryption. This methodology certainly makes it more difficult for hackers to attack computer systems, but is not foolproof. The key to blocking the hackers at the gates is a sound password policy that is effectively implemented. The implementation part is not always easy because end users often feel they must have passwords that are easily remembered [5]. There is often a conflict between ease of memorization and security. That conflict is easier to understand if one looks at the common methods used to break encrypted passwords. One method, the brute force attack relies on trying every possible combination of characters. Over time (at least in theory) you'll find the correct combination of characters; however, it may take a large number of iterations of passwords to find the right one. In this methodology there is not social engineering to help find a short cut. Thus, no matter what the password is, if it is long enough it may be effectively unbreakable (at least within an effective time period). Therefore, hacking tools have often focused on dictionary attacks. According to Arbelaez [1], in most cases, a dictionary attack is the most efficient way to retrieve a lost password. Dictionary attacks use a list of words (a "dictionary") and verify each word as a possible password. He further states that most successful dictionary attacks use custom-made dictionaries (which include every word that could be meaningful to the person who encrypted the document – his name, his birth date, his spouse's name, his company's name, his favorite team's name, and so on). The dictionary method makes it easy to pursue social engineering as part of the password breaking strategy and often it is the added speed of using this methodology that makes breaking passwords in a timely manner practical. Even with the dictionary method and the help of social engineering, Arbelaez [1], recognizes that the

process of 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. Historically, parallel processing evolved from supercomputers and was not readily available to the masses. In the last ten years numerous effective parallel processing applications have been implemented on distributed clusters using standard PCs [12]. These clusters have been used with success in the process of breaking passwords. In fact, Business Wire reported in 2000 [2] 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. Like any other form of technology, new attack methods are constantly evolving, and new defensive methods need to be devised in a timely manner. When the DES (data encryption standard) family of algorithms was first devised, it took 96 days using trial and error encryption busting programs to break DES encrypted code [9]. Only three years later using the same "brute force" logic it was broken in 22 hours [11]. This vast performance improvement can be attributed to the application of distributed processing. To attain this amazing performance improvement, a specially designed supercomputer front end and 100,000 PCs on the Internet were harnessed to provide processing power. This development makes it 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].

To be prepared to prevent successful attacks, security personnel must periodically test the integrity of their resources by applying commonly used attack methods to their systems. As stated earlier, the password file on any given host is a popular target of hackers. Upon securing a copy of this file, hackers often use tools that employ a "brute force" attack to ascertain if there are any weak passwords present. Weak passwords can frequently be broken from their encrypted form in a matter of seconds. Examples of weak passwords might include *letmein*, *password*, a password the same as the account name or, in reality, any word in the dictionary. It is common practice for security administrators to apply the hacking tools to their password files and if weak passwords are identified, then they have a chance to get the passwords changed before damage occurs.

Because of the availability of distributed computing, it is not only imperative that this proactive approach be applied to weak passwords, but any password or information using many of today's common algorithms is in danger of being compromised in less than a day. In fact, this need has been recognized by others. Castillo [3] states that "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."

The question becomes: what is the most effective and practical way to employ distributed computing to gain a competitive time advantage over hackers. This paper will build on a model devised by Guster, Safonov, Hall and Podkorytov [6], which is based on a commonly used password hacking tool which was modified for distributed processing. This tool, called John the Ripper [7], has been deployed for a number of years and is often one of the first tools employed by a developing hacker. The model proved quite effective in terms of speed of breaking and number of passwords broken. In fact, in one test the model reduced the time to break from one year on a single processor to 11 days on 32 processors. Furthermore, the modifiers used increased the number of passwords broken from 30 with the basic John program to 65 with the modifiers. However, the tests were run primarily on Sun servers. It would be expected that a good share of the attacks would be perpetrated by Intel based machines. Besides the factors of number of nodes and modifiers that proved effective within that model, the type of hardware used may also influence performance.

So, the purpose of this paper will be to build on the basic model suggested in [6] by adding hardware type as a new variable. A series of experiments will be run to ascertain if the Sparc processor contained in Sun servers offers superior performance to the Intel processors contained in standard PCs.

MODEL DESCRIPTION

To determine if hardware type influences the performance of breaking passwords with distributed processing, the multithreaded multiprocessor version of John program devised in [6] was tested at the 12 processor level using three different hardware types. Those hardware types were Intel Pentium, Sun Sparc II and Sun Sparc III. For all three processor types, the tests were performed on a 12 processor array. The software was configured to support two different strategies. The first strategy was to run in parallel the basic John program, which uses a basic dictionary attacked. The second strategy added two modifiers to the basic routine to increase the probability of breaking more passwords. These two modifiers, termed mirror and case, have increased the number of

words broken in previous tests by about 25%. The mirror modifier allows the reverse of each word in the dictionary to be tried as well the regular word. For example, in the word "the" the hash of "the" would be tried as well as the hash of "eht." The addition of the case *modifier* would ensure not just "the" was tried but the following case related variations as well:

The tHe thE The ThE tHE THE.

The combined test used all of the above methods plus reverse case variations as well. The tests were performed on passwords that used the DES (64) encryption method.

RUNNING THE EXPERIMENTS

To get some idea of the performance potential of this logic, an attempt was made to bust a sample password file on each of the three architectures. To make it easier to complete the tests in a timely manner, all passwords in the initial test series started with the letter "a" or were a mirrored word that started with the letter "a." This strategy meant that only the "A/a" portion of the dictionary needed to be searched, which drastically reduced the size of the dictionary file. These passwords were obtained from several discontinued Linux hosts. A total of 84 passwords were in the file. Therefore, the dictionary used only needed to include "a" words. To provide a flavor of the type of passwords contained in this file, the following are the first fifteen entries:

Ashby Ashby123 Aspen aspen AsPeN nepsa accomplished aCComplishED arthritis SitirhtrA aroma4321 aroma? AMora almost Al?most

The basic logic of the module code was designed so that the code could be split on a varying number of clustered machines. As stated earlier in this case, the number of nodes was set to 12. Four experimental runs were carried out within each architecture. First, the basic John program was run without modifiers. Second, the case modifier was added. Then the case modifier was removed and the mirror modifier added. Last, both modifiers were added to the basic John program. The results of the experiments are depicted in Tables 1-3. Table 1 provides the results for the Sparc II processors. Note that the number of passwords guessed column is cumulative for rows 2-4. In other words, for the case experiments, 23 of the broken passwords were broken by the basic John program and 19 additional passwords were broken by adding the case modifier.

Table 1. Sparc II 500 MHz Processors

	Min Time	Max Time	Number Guessed	Min Cps	Max Cps
No filters	00:00:00	00:00:01	23	17,284	23,820
Case	09:43:03	10:13:58	42	48,250	51,859
Mirror	00:00:01	00:00:01	25	33,686	45,284
Combined	12:47:47	18:21:38	59	53,614	58,832

The basic John program ran very quickly; actually it took less than one second to complete the run. Twenty three of the 84 (or 27%) of the passwords were broken by this basic attack. A maximum transfer rate of 23,820 characters per second was reached. This relatively low value when compared to the other experiments can be attributed to the fact that the encrypted password only needs to be compared to a single hash of each word in the dictionary. In other cases, when the modifiers are employed, multiple hashes of each dictionary word need to be devised. Adding the case modifier greatly increased the run time from less than a second to ~10 hours. This is because every combination of upper and lower case hash for a given dictionary word need to be tried against the encrypted password. However, the number of password broken increased from 23 to 42 (or exactly 50%) of all passwords in the file. The number of character per second more than doubled from ~23,000 to ~52,000. The mirror modifier experiment ran very quickly (less than 1 second). It required only two hashed to be generated for each dictionary word and hence its transfer rate fell in between the other two experiments. The combined test of course took the longest at up to ~18 hours on some nodes. The variation can be explained by the length of the words in the sample relegated to any given node. In other words, the "A/a" portion of the dictionary was broken into 12 equal parts in terms of number of words, but the size of those words was not

controlled for in trying to obtain like size samples. Fifty nine passwords (or 70%) of the total were broken by the combined test. The combined experiment as expected reached the highest transfer at ~58,000 cps. The results for the Intel processors are below:

Table 2. Intel 400-800 MHz Processors

	Min Time	Max Time	Number Guessed	Min Cps	Max Cps
No filters	00:00:01	00:00:01	23	17,284	23,884
Case	04:24:21	09:03:08	42	28,250	57,944
Mirror	00:00:01	00:00:01	25	23,200	23,840
Combined	04:53:47	22:23:16	59	12,570	91,519

There are a number of similarities between Table 1 and Table 2. First, because the same program was run on each, the number of passwords broken is the same. Second, the maximum completion times pretty much follow the same pattern. However, there is a much higher degree of variance within both time and transfer rate. This can be explained by the fact that not all processor in the 12 processor array used ran at the same clock speed. In fact, the speed varied from 400 to 800 MHz. This can create a major speed penalty if one views the time to solve as the slowest time of any given node in the array. In this case it is ~22 hours in the combined experiment—about 5 times longer than the minimum observed. In looking at the minimum times and maximum cps observed, it would appear that the Intel processors in the array with the clock speeds in the 800 MHz range out perform the Sparc II processors. The results for the Sparc III processors are displayed in Table 3.

Table 3. Sparc III 1 GHz Processors

	Min time	Max time	Number guessed	Min Cps	Max Cps
No filters	00:00:01	00:00:01	23	17,284	23,778
Case	04:39:41	04:39:41	42	104,426	108,175
Mirror	00:00:01	00:00:01	25	33,622	45,156
Combined	04:44:27	07:04:12	59	140,784	233,874

Again, because it is the same program, the number of passwords guessed is the same. With no modifiers or with just the mirror modifiers, performance is equal across all three processors and the time to solve is consistently less than one second. However, when the case modifier either alone or combined with the mirror modifier is added, the type and number of processors become important factors. For the case and the combined experiments, the Sparc III provides the best performance in regard to maximum time and characters per second. Although the word length is the same and the clock speed is only doubled, the Sparc III consistently out performs the Sparc II by a factor of three to four times in terms of both speed and transfer rate. Although its performance is generally better than the Intel array, the minimum times are in same range. So it would appear that the 800 MHz processor can offer similar performance. This is partially explained by the coding strategy of the Intel versus the Sparc versions of the basic John program. The Intel version is written in assembler code whereas the John version uses a higher level language. Thus, it would appear that an 800 MHz, 32 bit Intel processor running an assembler version is close in performance to a 1 GHz 64 bit Sparc III running a higher level language. At the time these tests were conducted a beta assembler version for the Sparc processors was underdevelopment, but was not stable enough for testing.

CONCLUSIONS AND RECOMMENDATIONS

The results of the tests conducted herein illustrate the effectiveness of distributed processing in speeding up the password breaking process. Furthermore, the software modifiers added significantly to the number

of passwords broken. The fact that the two modifiers employed increased the number of passwords broken from 23 to 59 indicates the additional vulnerability that occurs when a couple of common social engineering techniques are employed to speed up the basic search methodology. To a security officer, it provides a warning that any password variant based on a word in the dictionary is at risk. Furthermore, the security officer might want to monitor the results of these tests to help guide the standards for password generation within his/her organization. However, standards are often difficult to enforce because end-users like passwords they can easily remember. The results herein indicate that if it is a word in the dictionary, no matter if the letter case is varied or the word reversed, it will be broken by this routine. Furthermore, with distributed processing it is done quickly enough to be very dangerous. In the past, policies that required passwords to be changed on a monthly basis could defeat password breaking strategies, but with distributed processing that is no longer the case. Password files that took 1 year to bust on one machine could be broken with this software using 32 Sparc II processors in 11 days (and approximately 4 day on 32 Sparc III processors). Improvement in processor sophistication and array size make it more practical to pursue social engineering techniques through the "modifier" strategy. One would expect that other tricks that end-users might devise, such as reversing syllables, could be defeated by a modifier in this modular design logic. One would expect that over time additional modifiers will be devised both by defenders and hackers. However, each modifier added to computational chain will reduce performance. So if the complex test were rerun with four instead of two modifiers, the time to solve would increase dramatically. Therefore, if the time to break the password file is to remain within practical limits as more modifiers are included, larger processor arrays with more sophisticated processors will be required.

Although these results provide some interesting baseline information, additional work is still needed. Further research needs to not only focus on how this program will scale as additional processors are added (beyond the 12 level), but how to devise a modifier strategy. An information systems manager that doesn't take the threat of hacking seriously is just being naïve. It is a serious threat that requires proactive methods. The program logic offered herein provides the basis of one tool that could be effectively employed in the arsenal of defense against hackers. Furthermore, it is clear that more effective attacks loom over the horizon. To be able to proactively identify vulnerable passwords having a processing array superior to the hackers will be paramount.

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.crime-research.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/crackdes1.htm.