Ethical Hacking And Password Cracking: A Pattern For Individualized Security Exercises

Robin Snyder
Savannah State University
College of Business Administration
2nd line of address
912-356-2716
snyderr@savstate.edu

ABSTRACT
A design pattern provides a type of template or meta-template for solving design problems. This paper provides and discusses a pattern for creating individualized learning exercises. The domain of application is security education. The specific examples presented, that have been realized in practice in an educational setting, are password cracking/recovery and SHA-1 file hash. Additional learning exercises based on this pattern will be added in the future. Some implementation details and a discussion of the role of the SecureS software system, created by the author, are included.

Categories and Subject Descriptors
C.2.0 [**Computer-Communication Networks**] -- *security and protection*; D.4.6 [**Operating Systems**]: Security and Protection -- *cryptographic controls*; E.3 [**Data Encryption**]

General Terms
Security

Keywords
Patterns for security exercises, password cracking

1. INTRODUCTION
This paper provides and discusses the design and implementation of a pattern for creating customized web-based learning exercises for students in the area of security education. The specific examples discussed are password cracking/recovery and SHA-1 file hash. First, a quick review of hash and password concepts is covered in the context of the author's "SecureS" software system.

When teaching, for example, password cracking/recovery techniques, it must be stressed to the students that the techniques are for educational purposes and should not be used to attack systems. Password cracking/recovery should be used ethically. That is, to find existing vulnerabilities, recover forgotten passwords, etc.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
InfoSecCD Conference'06, September 22-23, 2006, Kennesaw, GA, USA. Copyright 2006 ACM 1-59593-437-5/00/0006…$5.00.

In ethical hacking, for example, a network administrator might use the encrypted password file and a "cracking" program to determine who has not picked a good password. The human part is in letting the users whose passwords have been cracked/recovered that they should pick another password (e.g., by having their passwords expire). Some systems run a cracker program on a user's selected password and will not let the user pick a password that is easily cracked. This happened to the author in 1988 it the IBM R&D Center. It took several attempts to get an acceptable password. The password "qwerty" did not work. But another "pattern" on the keyboard did work. Apparently that pattern was not forbidden.

2. SECURES
The author has developed software entitled "SecureS" that makes it easy for students to do various security-related exercises.

The "SecureS" system can be downloaded by students as a self-extracting executable file that installs itself along with the needed files. In this manner, the teacher can be more sure that the students have the needed software. To keep things simple, the installation directory is "`C:\FACULTY\SNYDERR`". In some cases, a GUI (Graphical User Interface) is provided in addition to the command line interface. Over time, the author will add more features to the system. Currently, the following functionality, which changes each time the course is taught, is provided.

Under "TCP/IP", features are provided for accessing the `msinfo32.exe` program, the `msconfig.exe` program, "Sam Spade", local computer info (IP address, user name, etc.), "WhoIs" functionality, "Ping" access, access to several scanner utilities (NetBrute, NMap, SNScan) and their web sites, and to the default command line interface.

Under "Crypto", hashing methods are provided for MD-4, MD-5, and SHA-1, and interfaces to "John the Ripper" (detailed below), `sha1sum.exe` (detailed below), steganography using `jphide.exe` and `jpseek.exe` (currently incomplete), and digital signatures (currently incomplete). The incomplete features will be added in a manner similar to the exercises detailed below.

Under "Time", interfaces are provided to the default Windows TimeDate control panel applet, to the NIST (National Institute of Standards and Technology) public domain time program, and the Windows `telnet.exe` program (for direct access to the daytime port 13).

When a command window is opened, a message informs the student what has happened (the first time) and how to close the command window.

The following hash and password cracking/recovery exercises are needed by the student for the learning exercise pattern to be described. These are now covered.

3. HASHING

A hash, or digest, function converts a value into another value such that, given the second value, it is not easy to determine the first value. For example, here are some hashes of the name **"Savannah"** expressed in hexadecimal.

```
hash value
---- --------
none Savannah
MD4  b08f57ef4e1609cf3cf0b00d1fa4a41f
MD5  cef52e4251b1b52e1da23854efbb0849
SHA1 c31ca1b500b60069c3255a5e1607f5f7f2d01020
```

Hashes are often used to store password information. Rather than storing the password, the hash of the password is stored. If the user typed password, after hashing, matches the previously saved password hash, it is assumed, with very high probability, that the user typed the correct password.

Hashes are often used to store password information. Rather than storing the password, the hash of the password is stored. If the user typed password, after hashing, matches the previously saved password hash, it is assumed, with very high probability, that the user typed the correct password.

The SHA (Secure Hash Algorithm), version 1 (i.e., SHA1 or SHA-1) is a very popular hash method. Older methods include MD4 and MD5. Hashes can be made of files to insure that a file is not changed (i.e., tampered). The program `sha1sum.exe` is a popular and readily available command line program for determining the SHA1 hash of a file.

The "sha1sum" tab provides an interface to run the `sha1sum.exe` program in order to determine an SHA-1 hash for a given file.

The student can see where the `sha1sum.exe` file is located, by default, or can set the path to another location, if desired. When the "Calculate checksum" button is pressed, a command window is opened and the result displayed.

4. PASSWORDS

A user name with password is often the only gatekeeper that is used to provide information security. Thus, it is important that users use secure passwords. A password cracker/recovery tool works via a brute force method using the encrypted passwords. A brute force method goes through every possibility in an attempt to achieve a goal. Most password cracking/recovery tools go though long lists of possible passwords (i.e., dictionaries) and variations of randomly generated passwords. The hash of each is generated and compared to the known hash. For off-line guessing of passwords, access to the encrypted passwords is required. Lockout can be used to sufficiently slow down online guessing of passwords such that an alarm can be set off and appropriate response taken.

Most systems will add a "salt" value to the typed password when hashing. The "salt" value is put in the database which makes it more computationally difficult to guess passwords if the attacker get access to the encrypted passwords and makes it harder to create databases of the inverse password hash (e.g., devoting a 1TB hard drive array to the inverse table).

Next to asking for the password, the next best way to get a password might be to find the encrypted password file posted on the Internet. Suppose that one does the following Google search.

```
ext:pwd inurl:(service | authors | administrators | users)
```

Here is one such hit that was found on Tue, Aug 23, 2005 (slightly modified).

```
# -FrontPage-
ekendall:bYabcde73NLKo
louisa:5zm9abcdeDFiQ
```

Note the following for FrontPage password files.

- The encrypted password file contains the user name.
- The encrypted password file contains the encrypted user password.
- The encryption algorithm is known, the DES (Data Encryption Standard).

This information potentially provide access to web site authorship.

One security problem here is that the encrypted password file is not itself encrypted. The parsing of "encrypted password file" is, using parentheses, "(encrypted password) file" and not "encrypted (password file)".

Here is another hit that was found on Tue, Aug 23, 2005 (slightly modified).

```
# -FrontPage-
cannon:Qcbaabcde4ub.
```

In this case, the site had removed this information. But the information was still in the Google cache, saved from a previous visit.

The term Googledork appears to be the "slang" term for a person who leaves such information available to Google. Thus, a Google search for "Googledork" would provide additional information and links on the subject.

Note that Google can be used to find search terms that can be used to look for various vulnerabilities on the Internet. To keep such files from being accessed via the Internet one might use the `urlscan.exe` program intended for Microsoft IIS (Internet Information Server) or the `.htaccess` file intended for the Apache web server. In both cases, one would disallow a file with extension `pwd` from being returned by the web server (i.e., being accessible directly from the Internet).

5. PASSWORD RECOVERY

What might be done with an encrypted password file? A password is usually encrypted. But the password can often be recovered. A password cracker is a program that allows you to recover or crack a password.

- Ethical term: password recovery
- Unethical term: password cracking

Be careful which term you use in which context.

One such password cracking program, I mean, recovery tool, is "John the Ripper", at http://www.openwall.com/john/ [as of Tue, Aug 23, 2005]. The "John the Ripper" program is a command line program. You start the program and type commands at the command line.

An actual password cracker would make guesses until the guess hash matches the actual hash. Besides using algorithmic methods, here are some common passwords near the top of the list for "John the Ripper".

```
12345
abc123
password
passwd
123456
newpass
notused
Hockey
internet
Maddock
12345678
newuser
computer
Internet
qwerty
```

If your password is on this list, then that password is not very secure. Here is the result of running "John the Ripper" on the above information that was saved in file `pwd3.dat`. (slightly modified).

```
[4nt] john.exe pwd3.dat
Loaded 1 password (Standard DES [24/32 4K])
6126             (cannon)
guesses: 1   time: 0:00:00:23 (3)   c/s: 67959
trying: 6cco - 63ri
[4nt]
```

It took `23` seconds to crack the password for user `cannon`. The password is `6496`. Note that some passwords may take much longer. It depends on the password used.

When restarted, "John the Ripper" keeps working on unsolved passwords and does not restart the previous work. Here is one way to confirm what work has been done to this point.

```
[4nt] john -show pwd3.dat
cannon:6126


1 password cracked, 0 left
```

If a password is not cracked quickly, one can let the program run longer, or use additional options to provide additional guidance to the program. One should always make your password hard to crack. One can always run a password cracker on your encrypted password to see how hard it is to crack. Or, one might run the password cracker on the encrypted password file for local users to see who does not have a very safe password. But, do not post the encrypted password file on the Internet where an attacker (e.g., using Google) can find it.

The "Ripper" tab provides an interface to run the "John the Ripper" password cracking/recovery program. The following options are provided.

- "Start Command Line"
- "Reset John the Ripper"
- "Run John the Ripper"
- Visit the "John the Ripper" web site.
- "Show command help" for "John the Ripper".
- Show the "Status of John the Ripper"

A text box is provided in which to copy-paste the text of an encrypted FrontPage password file.

6. WEB EXERCISE

The author has developed a web-based educational system for exercises such as password cracking/recovery, SHA-1 hash, etc. This system is now explained, first from the student point of view and then from the teacher point of view.

The student is presented with a simulated login box and must do the following as a typical scenario (different for each student).

- Step 1: Login as user mary
- Step 2: Login as user caitlin
- Step 3: Login as user amanda
- Step 4: Login as user dilbert
- Step 5: Login as user jenna
- Step 6: Answer question and submit work.
- You are student John Smith (smithj1).
- You are on step 1.

In this case, the student is on step "1" of "6" steps. The following encrypted password text is supplied for step "1".

```
# -FrontPage-
mary:Ds81AR.h3maHY
```

The student would then need to use software such as "John the Ripper" to crack/recover the password. Of course, the student could try guessing manually, but that strategy might not work very well. The SecureS system makes it easy for the student to use John the Ripper, and te see how John the Ripper works, should SecureS not be available.

On the simulated login, after recovering the password, the student moves to the next step. After completing a teacher-specified number of steps, the student answers a question and submits the answer to the teacher's on-line submission system. After making a submission, the student can verify their submission in the submission database, and can download any submitted file and see annotations on submitted work from the teacher.

The student receives this advice.

Note that the system administrator requires that users change their userid's and passwords every few days. So, you should complete the entire assignment at one sitting. Otherwise, you may have to start over next time. If you find a password that you cannot crack, you may have better luck next time.

In the Spring of 2006, the author had 6,500 submissions from 75 students in three sections of an upper-level e-commerce class. Needless to say, automated software helps greatly in the processing of such submissions.

7. PHP

The popular web server processing language PHP can be used both to verify the results of John the Ripper and to better see how passwords are encrypted and how a password cracker works. This simple program creates a FrontPage encrypted password file text for all possible passwords of two characters from the set characters "ABC" using salt value "YZ".

```
<?
allowed = "ABC";
max_len = 2;
count = 0;
salt = "YZ";

function permute($pw, $n, $state) {
 global $allowed;
 global $count;
 global $salt;
 if ($n == 0) {
  $hash = crypt($pw, $salt);
  $count++;
  if ($state == 0) {
   echo "<br>";
   echo " user=[user" . $count . "]";
   echo " password=[" . $pw . "]";
   echo " salt=[" . $salt . "]";
   echo " hash=[" . $hash . "]";
   }
  else {
   echo "<br>user" . $count . ":" . $hash;
   }
  }
 else {
  for ($i = 0; $i < strlen($allowed); $i++) {
   permute($pw. substr($allowed,$i,1), $n-1, $state);
   }
  }
 }

echo "<br><br>All passwords and hashes";
echo " from set <b>\"" . $allowed . "\"</b>";
echo " of length <b>" . $max_len . "</b>";
echo " using salt <b>\"" . $salt . "\"</b>:";
echo "<br><tt><b>";
count = 0;
permute("", $max_len, 0);
echo "</b></tt>";
echo "<br><br>Encrypted password file for above.";
echo "<br><tt><b>";
echo "<br># -FrontPage-";
count = 0;
permute("", $max_len, 1);
echo "</b></tt>";
?>
```

Here is the output.

```
All passwords and hashes from set "ABC" of length 2
using salt "YZ":


user=[user1]          password=[AA]          salt=[YZ]
hash=[YZsFm3YrgecZg]
user=[user2]          password=[AB]          salt=[YZ]
hash=[YZCHocMN3PJ7s]
user=[user3]          password=[AC]          salt=[YZ]
hash=[YZON1lNo5Gr2E]
user=[user4]          password=[BA]          salt=[YZ]
hash=[YZulckeKm1B.g]
user=[user5]          password=[BB]          salt=[YZ]
hash=[YZnD/9dO.pUuo]
user=[user6]          password=[BC]          salt=[YZ]
hash=[YZzqpgQD.Z36A]
user=[user7]          password=[CA]          salt=[YZ]
hash=[YZCK6d4a8atkA]
user=[user8]          password=[CB]          salt=[YZ]
hash=[YZWMRULpbn3oI]
user=[user9]          password=[CC]          salt=[YZ]
hash=[YZ1YTDCtmc6tI]


Encrypted password file for above.


# -FrontPage-
user1:YZsFm3YrgecZg
user2:YZCHocMN3PJ7s
user3:YZON1lNo5Gr2E
user4:YZulckeKm1B.g
user5:YZnD/9dO.pUuo
user6:YZzqpgQD.Z36A
user7:YZCK6d4a8atkA
user8:YZWMRULpbn3oI
user9:YZ1YTDCtmc6tI
```

Needless to say, changing the allowed character set to an actual set and the password size to a larger size would create an intractable amout of computation. But, the idea is simple. It only takes time to eventually guess the password. If the password is easily guessed, the password is secure. If the password is not easily guessed, it is more secure.

8. STUDENT VIEWPOINT

The design pattern for each exercise is as follows, from the student point of view.

The student logs into the system. Login is required to access almost every part of the author's web site so that audit trails of

page accesses can be maintained, information security can be maintained, etc.

The student navigates to the proper page and starts the exercise by being provided with the FrontPage text of a user name and encrypted password. The student then uses John the Ripper, usually via the "SecureS" interface, to crack/recover the password. Once cracked/recovered, the student is presented with another password to crack/recover. After 5 successful attempts, the student writes a paragraph to answer a question about the exercise and submits the work. The submission is handled by the author's class management software. Further details are omitted for space reasons.

The SHA-1 hash exercise is similar except that the student is presented with a sequence of files to download and for which to compute the SHA-1 hash. When the proper hash is submitted, the next file is provided. At the end of the sequence, the student writes a paragraph to answer a question about the exercise.

Since each student receives a different sequence of user names and passwords, students cannot copy answers from one another. While students may not mind showing other students their work or answers, or even copying the work of other students, simple copying will not work with this method. Thus, students are encouraged (i.e., forced) to either do their own work or get someone else to do the work for them.

9. TEACHER VIEWPOINT

Here is how it is done from the teacher point of view. Note that due to the experimental nature of this exercise, not all of this is point-and-click. In each case, the exercise is generated for all registered students of the class and for the teacher of the class. The teacher exercise is used by the teacher to demonstrate how the exercise works. On-line written documentation for the students is provided for each exercise on the author's web site.

Here are some specific details about the password cracking/recovery exercise. When the passwords and encrypted passwords are generated, each password is checked to insure that it will actually be cracked in a fairly short period of time. An XML (Extensible Markup Language) file containing the user names, passwords, and encrypted passwords, and actual passwords is created. Here is an example fragment of the XML for a two-step sequence (for just the teacher, for example purposes).

```
<?xml version="1.0"?>
<rmsSecures
 class="CISM3300"
 term="Fall 2005">
<account
 type="teacher"
 login="snyderr"
 name="Dr. Robin Snyder">
<challenge index="0" done="0">
# -FrontPage-
mary:Ds81AR.h3maHY
</challenge>
<challenge index="1" done="0"
 name="mary" code="password">
# -FrontPage-
jenna:DsKvHJgx2b7DI
</challenge>
<challenge index="2" done="1"
name="jenna" code="amber6"/>
</account>
</rmsSecures>
```

For the SHA-1 file hash exercise, each file is generated as a text file of about 1,000 random bytes. The hash of each file is determined. The file is stored on the web site to be accessed via a link. The hash information is stored in an XML file in a format similar to the password cracking/recovery exercise. A future improvement will only use legible text and store the text in the XML file, in a manner similar to the password cracking/recovery exercise.

The interface between the student and the XML files is currently done using ASP (Active Server Pages). The ASP page uses POST (i.e., form) parameters and session variables to keep track of the state of progress for the student. For space and complexity reasons, details on this are omitted. The author will be abstracting this pattern for use in other exercises the next time that the course is taught (i.e., for Fall 2006).

10. FEEDBACK

Most students did well on these exercises and indicated that the liked them and found them beneficial. Here is the stem and leaf diagram for the password security exercise.

```
Horizontal histogram for
A1: Asmt#1: Password security (100.0% = 30 points)
23 scores (1 missing scores)

 100.0%+ : 30 30
  80.0%+ : 29 29 27
  60.0%+ : 23 23 23 23 20 20 20 18 18 18 18
  40.0%+ : 16 16 14 14 14 12 12
```

Here is the stem and leaf diagram for the data integrity exercise. The one student who did not complete this exercise was too busy to do the work (job-related reasons).

```
Horizontal histogram for
A2: Asmt#2: Data integrity (100.0% = 30 points)
19 scores (5 missing scores)

 100.0%+ : 30 30 30
  80.0%+ : 29 27 27 27 27 27 25 25 25 25 25
  60.0%+ : 23 23 21 20 20
```

Five students failed to complete this exercise, compared to one student from the first exercise. Those five students scored 12, 14, 14, 16, and 18 on the first exercise. That is, at the bottom of the scoring for the first exercise. All of these students were in the bottom half of the scoring overall at the end of the semester. But, those that did complete this exercise apparantly learned the mechanics from the previous exercise. The distribution of scores arises from the quality of their written answer to the question at

the end of the exercises. But, before that question can be answered, the sequence of learning exercises must be completed.

Additional learning exercises based on the pattern described will be added in the future.

11. SUMMARY

This paper has presented and discussed a pattern for creating individualized learning exercises in the area of security education. The specific examples presented, that have been realized in practice, are password cracking/recovery and SHA-1 file hash.

12. REFERENCES

[1] McClure, S., Scambray, J., & Kurtz, G. *Hacking Exposed.* New York: McGraw-Hill Osborne Media, 2005.

[2] Schneier, B. *Secrets & lies: digital security in a networked world.* New York: John Wiley & Sons, Inc., 2000.

[3] Simpson, M. *Hands-On Ethical Hacking and Network Defense*. Boston, MA: Course Technology, 2005.

[4] Smith, R. *Authentication: From Passwords to Public Keys.* Boston, MA: Addison-Wesley Professional, 2001.

[5] Snyder, R.. IIS security considerations of the FileSystemObject for cooperating and uncooperating users, In ***Proceedings of the 31st Annual Conference of the International Business Schools Computing Association (Daytona Beach, FL***, July 13-16, 2003).

[6] Snyder, R., Proactive approaches to information systems and computer security, In ***Proceedings of the 27th Annual Conference of the Association of Small Computer Users in Education. (Myrtle Beach, SC***, June 12-16, 1994).

[7] Snyder, R., Using ethical hacking to educate users about secure passwords by cracking insecure passwords using readily available software. In ***39th Annual Conference of the Association of Small Computer Users in Education (Myrtlie Beach, SC, June 11-15, 2006)***.