

HW #7: Security Design Principles

CS 392/6813: Computer Security

Fall 2007

Due 11/14/07

[100pts]

Problem 2 [30pts]

If you don't know about them already, learn about the following computer security concepts (say, through wikipedia) and explain which security design principle(s) they exploit or violate:

1. Sandboxes
2. Spam (email, web,..)
3. Pharming
4. Trusted computing
5. Malware
6. Reverse engineering

Problem 1 [35 + 35pts]

A web server implements access control using two mechanisms. The (perl based) pseudo code for the implementation of these two mechanisms, called version 1 and version2, is provided below. These versions are exactly the same as in the last homework.

- Argue whether or not version 1 and version 2 implementations satisfy the eight security design principles that we studied in class. In case a design principle is not applicable, mention it and explain the reason why it is not applicable.

#Pseudocode for Web Server Access Control Application

#version 1

#

Network Setup

#

Setup Socket Listening for Connections

socket_listen();

Bind Socket to port 80

```

# bind_socket(port)
bind_socket(80);

# Start Listening to incoming connections
listen();

# Accept new connection
while listen == true
    accept();

#
# Connected to Client
#

# Ask if user is returning or needs new account.
send_welcome();

    # catch_response will determine what the user's answer is and
ensure it is
    # then send it on to create new user or returning user
    in the format we want (yes or no)
    catch_response();
        new_user();
        returning_user();

#
# Create a new user account
#
# new_user function
$name = request_new_name();

    # Make sure username is 4 or more characters and less than 20
    # input_lenght(min, max, name)
    input_length(8, 20, $name)

    # See if username exists by checking against /etc/.app_passwd file
    user_exists($name)

# Request new name if necessary
$name = request_new_name();

# If name is ok, ask for password
$newpass = request_new_pass();

# Ensure password is 6 or more characters max 30
input_length(6, 30, $newpass);

# If password fails check, ask for new one
$newpass = request_new_pass();

    # If password and name are ok, record them to /etc/app_passwd
    # First need hash of password, passwd_hash calls the unix command
line md5 -s
    # program to create a secure hash

```

```

    $passwd_hash = gen_hash($newpass)

    # Write out username, password and date of creation to
    /etc/app_passwd file

    # Grab Date
    $date = gen_timestamp();

    # add_user(date, username, password hash)
    add_user($date, $name, $passwd_hash);

#
# Authenticate Returning User
#
# returning_user function
$username = request_name();
$pass     = request_passwd();

# First need hash of password, passwd_hash calls the unix command line
md5 -s
# program to create a secure hash
$passwd_hash = gen_hash($pass)

# Compare the given username and password hash to the hash stored in
# /etc/app_passwd file for the given user through the compare_pass()
function
# compare_pass(user, hash)
$valid = compare_pass($username, $passwd_hash);

# Grab Date so we can record last login time
$date = gen_timestamp();

# If the user is authenticated, report last login time and record
current time
if ($valid) {
    $lastlogin = grab_logintime($username);
    record_login_time($username, $date);

    # From here user can procede into system

# else if user fails login, start back over and let them try again
else {
    send_welcome();

-----
-----

#Pseudocode for Web Server Access Control Application
#version 2

#
# Network Setup
#

```

```

# Setup Socket Listening for Connections
socket_listen();

# Bind Socket to port 80
# bind_socket(port)
bind_socket(80);

# Start Listening to incoming connections
listen();

# Accept new connection
while listen == true
    accept();

#
# Connected to Client
#

# If connecting client is running v2 then they should send cert right
away
$cert = grab_cert();

# See if cert is signed by our CA and if signature is valid
# check_cert(certificate) returns true or false
$signed_cert = check_cert($cert);

#
# Proceede with Certificate Authentication
#

# If the cert is signed by our CA we can procede to check cert for
freshness
if($signed_cert)

$cert_date = get_cert_date($cert)

# Grab Date so we can compare to date on cert
$today = gen_timestamp();

# Compare date on cert to today's date to determine freshness of cert
# compare_dates(date, date, time limit in days) returns true or false
$fresh = compare_dates($today, $cert_date, 365)

if (!$fresh) then disconnect("Certificate is no longer valid.")

else continue
$username = extract_user($cert);

# Check name on cert against list in /etc/app_certs
# check_user(username) returns true or false depending on
# if the Name from the certificate is stored in the app_certs file
$valid_user = check_user($username);

```

```
#
# From here user can procede into system
#
if $valid continue...

else disconnect("Certificate does not contain valid Authentication
Name")

#
# Revert to version 1
#

else if !$signed_cert

# If Certificate is not valid, either the cert is bad or the client is
not
# using v2 of software. Lets post a message about where to get info on
v2
# of software and how to get a valid cert from our cert server then
drop back
# to username/password access

# Welcome tells user about upgrade procedures
send_welcome();

#
# Authenticate Returning User
#
# returning_user function
$username = request_name();
$pass     = request_passwd();

# First need hash of password, passwd_hash calls the unix command line
md5 -s
# program to create a secure hash
$passwd_hash = gen_hash($pass)

# Compare the given username and password hash to the hash stored in
# /etc/app_passwd file for the given user through the compare_pass()
function
# compare_pass(user, hash)
$valid = compare_pass($username, $passwd_hash);

# Grab Date so we can record last login time
$date = gen_timestamp();

# If the user is authenticated, report last login time and record
current time
if ($valid) {
```

```
$lastlogin = grab_logintime($username);  
record_login_time($username, $date);
```

```
# From here user can procede into system
```

```
# else if user fails login, start back over at welcome and let them try  
again
```

```
else {  
    send_welcome();
```

```
-----  
-----
```