A decorative graphic on the left side of the slide consisting of two overlapping parallelograms. The front one is blue and the back one is a light green color. They are positioned diagonally, with the blue one in front of the green one.

Secure data processing application

Kelly Lunghamer and Siddharth Sharma



Introduction

- Application is a blog post, with an extra layer security achieved using Hmac and nonce.
 - Hmac protects the data integrity
 - Nonces protects against the replay attacks
- Passwords are protected using salts
- We used industry standard encryption methods, such as AES with PBKDF2 to encrypt the post.
- User can sign up, login, and create blog posts, while no two users can post the same content to protect intellectual property



Phases of security implementation

- Phase 1: Setting up a secure connection
 - RSA, AES 256, SHA256
- Phase 2: Login
 - One-way functions, Oracle
- Phase 3: Post a blog
 - AES with PBKDF2, HMAC

Phase 1: PKI examples

The screenshot shows a Safari browser window with the address bar displaying `www.welivesecurity.com`. The page content includes the **welivesecurity** logo and a headline "Two Brits hack". A security warning dialog is open, indicating an encrypted connection. The dialog shows the following certificate details:

- GlobalSign Root CA**
- GlobalSign CloudSSL CA - SHA256 - G3**
- q.ssl.fastly.net**
- Common Name:** GlobalSign CloudSSL CA - SHA256 - G3
- Serial Number:** 35 CB F6 96 12 15 71 52 F6 37 3B 58
- Version:** 3
- Signature Algorithm:** SHA-256 with RSA Encryption (1.2.840.113549.1.1.1)
- Parameters:** None
- Not Valid Before:** Thursday, June 21, 2018 at 1:24:11 PM Eastern Daylight Time
- Not Valid After:** Sunday, April 14, 2019 at 12:29:44 PM Eastern Daylight Time
- Public Key Info**
 - Algorithm:** RSA Encryption (1.2.840.113549.1.1.1)
 - Parameters:** None
 - Public Key:** 256 bytes : E1 D0 14 35 7C 1C 18 4C ...
 - Exponent:** 65537
 - Key Size:** 2,048 bits
 - Key Usage:** Encrypt, Verify, Wrap, Derive
 - Signature:** 256 bytes : 5F D7 1B A8 92 81 A0 7D ...
- Extension:** Key Usage (2.5.29.15)
- Critical:** YES
- Usage:** Digital Signature, Key Encipherment

The dialog has buttons for "Hide Certificate" and "OK".

Below the dialog, the article text continues: "The breach exposed the pers... ny £77 million". At the bottom, it says "Two young Brits have been jailed for their roles in the breach at the telecommunications company TalkTalk in".

At the very bottom of the image, there is a footer that reads "Hacking Introduction Courses".

Phase 1: PKI examples (cont.)

The screenshot shows a Safari browser window with the address bar displaying `darknet.org.uk`. A security overlay is visible, indicating a secure connection. The certificate details are as follows:

COMODO RSA Certification Authority	
Organization	COMODO CA Limited
Common Name	COMODO RSA Domain Validation Secure Server CA
Locality	Salford
Serial Number	0B 0B 21 2E BE 60 C8 DE CD 5F C2 1B 93 1D 6B F7
Version	3
Signature Algorithm	SHA-256 with RSA Encryption (1.2.840.113549.1.1.1)
Parameters	None
Not Valid Before	Monday, February 15, 2016 at 7:00:00 PM Eastern Standard Time
Not Valid After	Friday, February 15, 2019 at 6:59:59 PM Eastern Standard Time
Public Key Info	
Algorithm	RSA Encryption (1.2.840.113549.1.1.1)
Parameters	None
Public Key	256 bytes : CE B7 D2 6F E4 CA 0E CC ...
Exponent	65537
Key Size	2,048 bits
Key Usage	Encrypt, Verify, Wrap, Derive
Signature	256 bytes : 30 4C 26 A0 84 3A 44 02 ...
Extension	Key Usage (2.5.29.15)
Critical	YES

Buttons: [Hide Certificate](#) [OK](#)

The background article, titled "Secure Erasure Methods", discusses the importance of secure data deletion. It mentions that simply deleting a file does not always remove it from the disk, as the data remains recoverable. It also notes that secure erasure is critical for protecting sensitive information, especially in environments where data is recycled or reused.

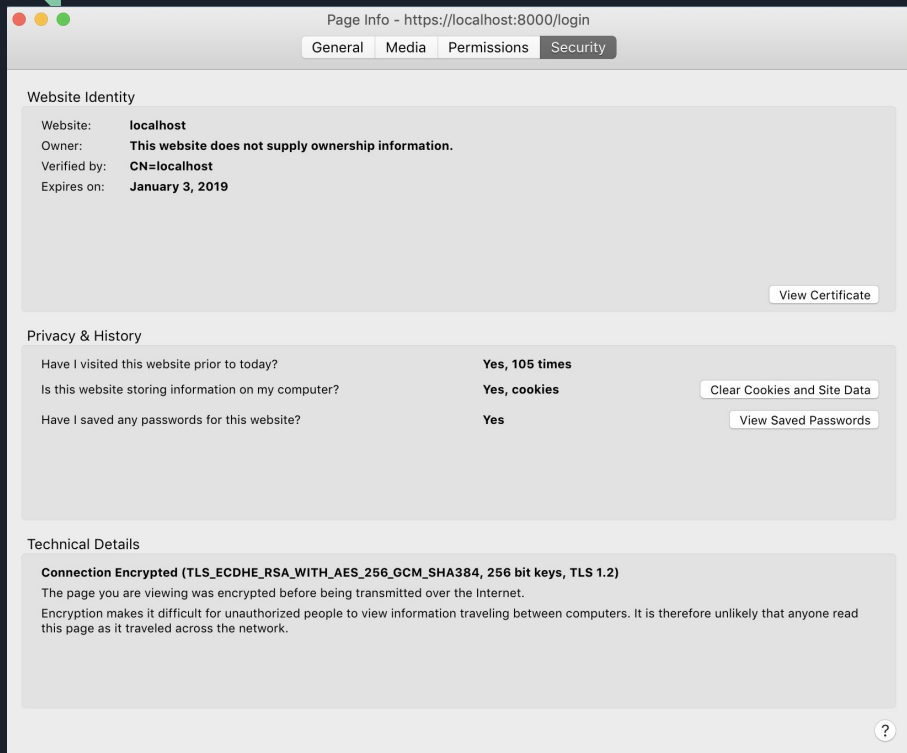
At the bottom of the page, there is a section titled "DISCO AVAILABLE" and a link to "Pseudorandom data, 1 P".



Phase 1: Certification generation

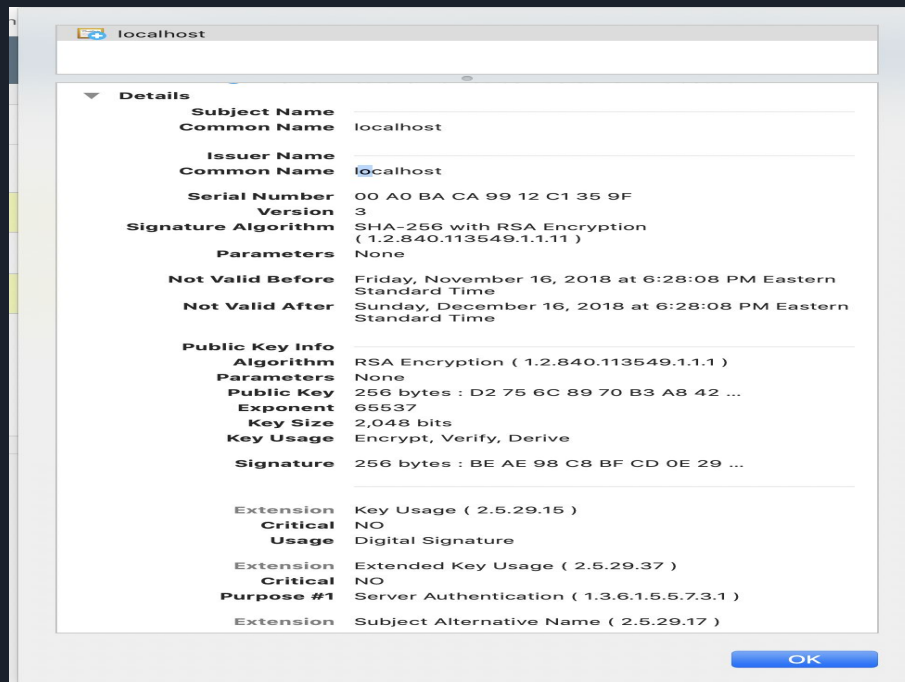
```
openssl req -x509 -out localhost.crt -keyout localhost.key -newkey rsa:2048 -nodes -sha256  
-subj '/CN=localhost' -extensions EXT -config <(\ printf  
"[dn]\nCN=localhost\n[req]\ndistinguished_name =  
dn\n[EXT]\nsubjectAltName=DNS:localhost\nkeyUsage=digitalSignature\nextendedKeyUsage  
=serverAuth")
```

Phase 1: PKI Certificate



- Cipher suite:
 - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA_384
 - ECDHE guarantees perfect forward secrecy
 - key exchange - RSA 2048
 - encryption - AES 256 **with GCM**
 - hashing - SHA256

Phase 1: PKI Certificate (cont.)

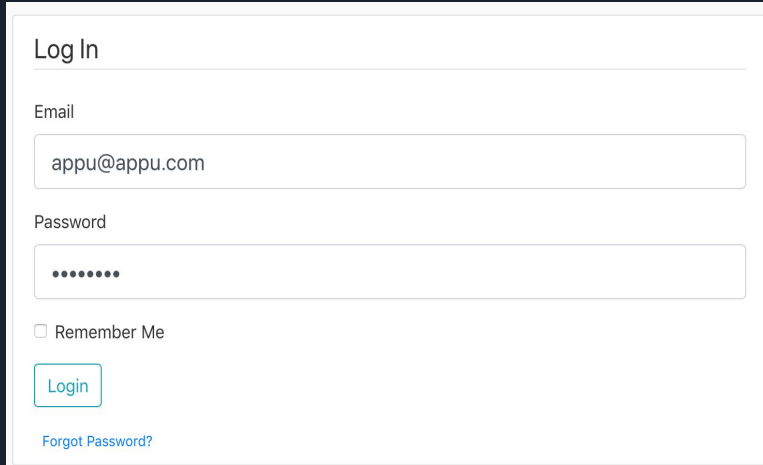


The screenshot shows a Windows Certificate Import Wizard window titled "localhost". The "Details" tab is selected, displaying the following information:

Details	
Subject Name	localhost
Common Name	localhost
Issuer Name	localhost
Common Name	localhost
Serial Number	00 A0 BA CA 99 12 C1 35 9F
Version	3
Signature Algorithm	SHA-256 with RSA Encryption (1.2.840.113549.1.1.11)
Parameters	None
Not Valid Before	Friday, November 16, 2018 at 6:28:08 PM Eastern Standard Time
Not Valid After	Sunday, December 16, 2018 at 6:28:08 PM Eastern Standard Time
Public Key Info	
Algorithm	RSA Encryption (1.2.840.113549.1.1.1)
Parameters	None
Public Key	256 bytes : D2 75 6C 89 70 B3 A8 42 ...
Exponent	65537
Key Size	2,048 bits
Key Usage	Encrypt, Verify, Derive
Signature	256 bytes : BE AE 98 C8 BF CD 0E 29 ...
Extensions	
Extension	Key Usage (2.5.29.15)
Critical	NO
Usage	Digital Signature
Extension	Extended Key Usage (2.5.29.37)
Critical	NO
Purpose #1	Server Authentication (1.3.6.1.5.5.7.3.1)
Extension	Subject Alternative Name (2.5.29.17)

OK

Phase 2: Login



Log In

Email

appu@appu.com

Password

••••••••

☐ Remember Me

Login

[Forgot Password?](#)

- bcrypt is a one-way function that creates a hash of the password
- It also uses a salt that is randomly generated
- Given a hash and the user-inputted password, an oracle can verify a successful login at the server



Phase 2: Code using bcrypt

```
form = RegistrationForm()
if form.validate_on_submit():
    hashed_password = bcrypt.generate_password_hash(form.password.data).decode('utf-8')
    user = User(username=form.username.data, email=form.email.data, password=hashed_password)
    db.session.add(user)
    db.session.commit()
    flash('Your account has been created! You are now able to log in', 'success')
    return redirect(url_for('login'))
return render_template('register.html', title='Register', form=form)
```



Phase 2: Code using bcrypt (cont.)

```
user = User.query.filter_by(email=form.email.data).first()
if user and bcrypt.check_password_hash(user.password, form.password.data):
    login_user(user, remember=form.remember.data)
    next_page = request.args.get('next')
    return redirect(next_page) if next_page else redirect(url_for('home'))
```



Phase 3: Blog post

- All blog posts' title and content are encrypted on the client side.
- AES keys never travel over the internet, hence are not prone to any kind an attack.
- We are appending our own SHA256 Hmac with our data, to add an extra layer of security to protect against unauthorized data manipulation.
- We are using a nonce to protect against replay attacks.

New Post

Title

Key

Generate Key

Content

Post

Phase 3: Protecting IP & Nonces

```
replay_token = generate_iv()
hmac = Replay(text=replay_token, used=0)
db.session.add(hmac)
db.session.commit()
return render_template('create_post.html', title='New Post',
                        form=form, legend='New Post', replay_token=replay_token)
```

```
title = hash_sha256(data['title'])
```

```
res = Hmac.query.filter_by(title=title).all()
if res:
    return '1'
```

```
content = hash_sha256(data['content'])
res = Hmac.query.filter_by(content=content).all()
if res:
    return '2'
```

Definition:

[flaskblog/routes.py:295](#)



Security considerations

- Confidentiality
 - The connection is encrypted using AES, protecting confidentiality
 - ECDHE also provides perfect forward secrecy and prevents the compromise of a long-term secret key from affecting the confidentiality of past conversations
 - Our login procedure provides authenticity and confidentiality
 - Blog post keys on the client side, so they cannot be stolen over the network
- Integrity attack detection
 - SHA256 is used for HMAC, which protect data integrity and authenticity
 - All blog posts to the server append an HMAC and which is validate by the server
 - Users who are not logged on or registered cannot post to the blog site
- Efficiency
 - Used 2048 instead of 4096 to avoid extra CPU consumption without compromise to security
 - bcrypt is efficient, because it uses Blowfish - which provides a quick encryption rate
 - All the encryption/decryption of the blog posts are conducted on the client side increase efficiency



Potential attacks/adversaries

- If an adversary is eavesdropping, they can get the encrypted data but could only brute force; we are using AES 256 with GCM (Galois counter mode)
 - GCM is widely adopted because of its efficiency and performance; it also provides data authenticity and confidentiality
 - AES proves protection against eavesdropping
- Chosen message attacks and chosen ciphertext attacks
 - AES with GCM will protect us attack these attacks.
- Protection against replay attacks are provided using nonces
- There is a potential for brute force, but it's not feasible considering time
 - We also used PBKDF2 for blog posts, which prevents brute forcing



References

1. https://www.ibm.com/support/knowledgecenter/bg/SSKTMJ_9.0.1/admin/conf_port_enc_adv_r.html
2. <https://en.wikipedia.org/wiki/PBKDF2>