Security

- Learning Objectives
  - Be able to explain the Heartbleed vulnerability.

- Topics
  - The three A’s of security
  - Security through encryption
  - SSL
  - Heartbleed

Thanks to Sean Cassidy’s Existentialize Blog
Caveat

• Security is serious business.
• Just because we talk about ways to break systems does not mean that we condone doing so.
• Anyone who takes what we learn in this class and uses it to do “bad things” will incur not only any organizational or legal penalties, but also the wrath of Margo™.
• Be a “white hat” not a “black hat.”
Not all Problems are Technical

• Virtual memory is a technical problem.
• File systems are a technical problem.
• Security is both a technical and a social problem.
• Consider:
  • What expectations are reasonable for computer security?
  • What precautions do we take for security in the real world?
  • How do those translate into security properties in the virtual world?
  • In what ways are the virtual world and physical worlds similar with respect to security?
  • In what ways are they different?
Goal of Security Protection

• Avoid both accidental and intentional misuse of a system.
  • Prevent unintended access, modification, or destruction of private information within the system.
  • Prevent information leaking through covert channels.
  • Prevent denial of service (DOS) attacks.

• Accidents
  • Relatively easy to solve if you make the probability of the accidents happening small enough.
  • Someone with super-user privileges accidentally deletes the command interpreter. Nobody else can log in (because login will fail to find the command interpreter) to fix the problem.
  • User forgets to logout; next person has access to that user’s account.
  • Solution: Make it difficult to make serious mistakes.

• Malicious Abuse
  • Very difficult to completely eliminate because you cannot play probabilities. You must get rid of all loopholes.
  • High school hacker breaks password for Bank of America accounting system and transfers billions of dollars to his own account (just for fun!).
  • Someone installs a trojan horse or virus in system software.
The Three A’s of Security

• **Authentication**: Verify the identity of a user

• **Authorization**: Determine if the user is allowed to do something

• **Access enforcement**: prevent users from doing that which they are not authorized to do.
The Three A’s of Security

• **Authentication**: Verify the identity of a user
  • Passwords
  • Biometrics

• **Authorization**: Determine if the user is allowed to do something
  • A mapping from a user identity to a set of credentials
  • Credentials tell you what a user can do.

• **Access enforcement**: prevent users from doing that which they are not authorized to do.
  • Sounds easy: just check credentials before doing anything sensitive.
  • Challenge comes from bugs and oversights
Encryption

• Everyone has heard of encryption, but:
  1. What is it?
  2. How does it work?
  3. Which of the 3 A's does it address?
• What is it?

Plain text

One-way
Cryptographic
function

Cipher text

4/29/14 CS161 Spring 2014
Cryptographic Functions

- Take encryption and decryption keys and transform plain text into cipher text. Function should be one-way (non-reversible).
  - Sometimes encryption and decryption keys are the same; sometimes they are different.
- Private key encryption:
  - Same key is used to encrypt and decrypt.
  - Requires that you keep the key secret, because anyone with the key can decrypt a message.
- Public key encryption:
  - Encryption and decryption keys are different.
  - Each individual has a keypair – either can be used to encrypt and the other will then decrypt.
  - Much more computationally expensive than private key.
More on Public Key Encryption

• Called “public key” because you can publish one of your keys and keep the other secret (private).
• If you encrypt with your private key, anyone can decrypt with your public key.
• If anyone encrypts with your public key, only you can decrypt with your private key.
Public Key Encryption in Action (1)

Any Problem?

Yes! He could decrypt it too!
Public Key Encryption in Action (2)

Any Problem?

Yes! What if the message is really from the bad guy?
Public Key Encryption in Action (3)

1. Only she can open it
2. Only his public key will decrypt it.
SSL: Secure Socket Layer

• Network protocol to transmit private documents.
• Uses public key encryption.
• SSL depends on certificates, which are typically issued by a “certifying authority” (CA).
• A standard certificate binds together:
  • A domain name, server name or hostname AND
  • An organizational identity and location

By creating a small file containing that information signed (encrypted) by the CA.
Viewing Certificates

• Click on the little padlock!
SSL in action (SSL Handshake)

Yo, server, please identify yourself.

Here is my certificate and my public key.

Hmm – looks OK to me.

OK, here is a session key encrypted with your public key.

ACK (in other words, I have the session key)

2-way encrypted communication using session key
Heartbleed

• Bottom line: allows an attacker without any privileged information or credentials to read up to 64 KB of memory from the machine under attack (i.e., the server).

• For example:
  • It can read secret keys
  • It can read user names
  • It can read passwords
  • It can read sensitive documents
  • In other words, it can read anything!
Heartbleed: Heartbeat

- Like many distributed algorithms, SSL has a heartbeat protocol.
  - This is a protocol that machines use to check if other machines are alive.
  - Typically machines send each other periodic messages so the other machine knows that the sender is still alive.
  - The receiver needs to process such heartbeat messages.
  - In SSL, peers send each other a heartbeat packet with random data and the other peer is supposed to respond with the same data.
Heartbleed: Heartache (1)

- SSL record structure

```c
typedef struct ssl3_record_st {
    int type;         /* type of record */
    unsigned int length;    /* How many bytes available */
    unsigned int off;      /* read/write offset into ‘bug’ */
    unsigned char *data;   /* pointer to the record data */
    unsigned char *input;  /* where the decode bytes are */
    unsigned char *comp;   /* only used with decompression — malloc()ed */
    unsigned long epoch;   /* epoch number, needed by DTLS1 */
    unsigned char seq_num[8]; /* sequence number, needed by DTLS1 */
    unsigned int orig_len; /* How many bytes were available before padding was removed? This is used to implement the MAC check in constant time for CDC records. */
} SSL3_RECORD;
```
SSL Record Data/Meta-data

<table>
<thead>
<tr>
<th>type (4)</th>
<th>length (4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>off (4)</td>
<td>data (4)</td>
</tr>
<tr>
<td>input (4)</td>
<td>comp (4)</td>
</tr>
</tbody>
</table>

- epoch (8)
- seq-num (8)

Payload

Metadata

Data
Heartbleed: Heartache (2)

int
dtls1_process_heartbeat(SSL *s)
{
    unsigned char *p = &s->s3->rrec.data[0], *pl;
    unsigned short hbtype;
    unsigned int payload;
    unsigned int padding = 16;

    /* Read type and payload length first */
    hbtype = *p++;
    n2s(p, payload);
    pl = p;

    ...
SSL Record Data/Meta-data

<table>
<thead>
<tr>
<th>type (4)</th>
<th>length (4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>off (4)</td>
<td>data (4)</td>
</tr>
<tr>
<td>input (4)</td>
<td>comp (4)</td>
</tr>
<tr>
<td>epoch (8)</td>
<td></td>
</tr>
<tr>
<td>seq-num (8)</td>
<td></td>
</tr>
</tbody>
</table>

Metadata

Payload

Data
Heartbleed: Heartache (3)

• Later in the same code ...

```
unsigned char *buffer, *bp;
int r;

/* Allocate memory for the response, size is 1 byte for the message
 * type, plus 2 bytes payload length, plus the payload, plus padding
 */
buffer = OPENSSL_malloc(1 + 2 + payload + padding);
bp = buffer;

/* Enter response type, length and copy. */
*bp++ = TLS1_HB_REPONSE;
s2n(payload, bp);
memcpy(bp, pl, payload);
```
Heartbleed: Heartache (4)

• So, where is the memory that OPENSSL_malloc allocated?
• And more importantly, what is near it???