Case 1: Cookie Authentication Scheme

Case 2: Encryption of Credit Card Numbers

Conclusion
OUTLINE

**Case 1: Cookie Authentication Scheme**

**Case 2: Encryption of Credit Card Numbers**

**Conclusion**
A cookie authentication code (`auth_code`) is updated whenever the server changes a client’s cookie.

`auth_code` is generated as follows:

\[
\text{auth\_code} = \text{sha512}(\text{secret\_key} + \text{cookie})
\]

Once received by the client, `auth_code` is verified by the server as follows:

\[
\text{auth\_code} == \text{sha512}(\text{secret\_key} + \text{cookie})
\]

The attacker needs to know the `secret\_key` in order to generate a valid `auth\_code`, right?
**BEFORE LOGIN**

Client's cookie:

```
last_visit=961433797 auth_code=3a141c3d13...
```

Where:

```
auth_code = sha512(secret-key + "last_visit=961433797")
```
**After Login**

Client's cookie:

last_visit=961433797 user=foo auth_code=...

Where:

auth_code = sha512(secret-key + "last_visit=961433797 user=foo")
MD5, SHA-0, SHA-1 and SHA-2 families of hash functions all suffers from the length-extension bug:

Let

\[ H = \text{hash}(m_1 + m_2 + m_3) \]

Then

\[ \text{hash}(m_1 + m_2 + m_3 + m_4) = \text{hash}(H + m_4) \]

Assuming padding is corrected for.
Add `user=admin` to the cookie (before login) and update the `auth_code` as follows:

```python
auth_code = sha512(auth_code + "user=admin")
```

Assuming padding is corrected for, the attacker is logged in as `admin` (if user exists).
Do not use insecure hash constructions for authentication schemes, use a secure MAC function (such as HMAC-SHA512) instead.
OUTLINE

Case 1: Cookie Authentication Scheme

Case 2: Encryption of Credit Card Numbers

Conclusion
A webshop stores credit card numbers encrypted in a database.
The encryption is done using the RC4 stream cipher with an über strong key.
Without the key, an attacker is unable to get a hold of the credit card numbers, right? It depends...
Good Things To Know About RC4

- A keystream is generated that is XORed with the plaintext resulting in the ciphertext.
- The keystream is independent of the plaintext, it’s derived from the key.
- A keystream must never be used more than once (true for most stream ciphers).
**Known-Plaintext Attack**

- Lets assume that
  - the attacker creates a profile on the webshop and submits a (bogus) card number (known-plaintext).
  - the attacker has access to the encrypted credit card numbers (ciphertext).
- The attacker can then mount a known-plaintext attack against the system.
Demo of Known-Plaintext Attack
Lesson Learned

Do not use the same keystream more than once when encrypting data using a stream cipher.
Case 1: Cookie Authentication Scheme

Case 2: Encryption of Credit Card Numbers

Conclusion
GENERAL ADVICE

- Do not develop your own crypto schemes.
- If you do, use high-level crypto APIs such as Keyczar and cryptlib.
Questions/comments?