Homework 3
CS 642: Information Security

November 10, 2011

This homework assignment covers topics in cryptography. You *may* work with a partner. It is due **November 22, 2011** by midnight local time.

The deliverable should be a nicely formated hw3.txt or hw3.pdf file, in addition to a file ciphertext.txt for problem 2. If you attempt the extra credit, then include a zip file with all the dependencies (except the target perl script) required to run it.

Each part of each problem is worth 2 points, so you have the opportunity to get partial credit for each. The extra credit portion is 2 points of extra credit. Make answers as concise (you will not get credit for rambling and long answer that happens to contain some correct portions). Hand in the homework using the handin program or email the TA.

1 Problem 1

A colleague has built a password hashing mechanism. It applies SHA-256 to a string of the form "username,password,salt" where salt is a randomly chosen value. For example, the stored value for username "user" and password "12345" and salt "999999" is

0x873b8b6a77af4bb6cee4cae09eaa81b27556c7cd9786e754a169114b6d3674d5

For example, the Perl code to generate this is:

```
#!/bin/usr/perl

use Digest::SHA qw(sha256_hex);

printf sha256_hex( "username,12345,999999") ;
```

or in one line:

```
perl -e 'use Digest::SHA qw(sha256_hex); print sha256_hex("username,12345,999999");'
```

The same process was used to generate the hash

0x37448ba7de7f5b4396697edaeddcd7bc840964e6ce82016915b830a91d69eb2f

for user "ristenpart" and salt "134153169".

1. Recover the password used to generate the second hash above. Hint: The password consists only of numbers.
2. Give a pseudocode description of your algorithm and the worst case running time for it.
3. Discuss the merits of your colleague's proposal. Suggest how your attack might be made intractable.

2 Problem 2

In this problem we'll investigate a poor cryptographic encryption mechanism. It is described by the code in `badencrypt.pl` and `baddecrypt.pl` on the homework website. Roughly, it implements a MAC-then-Encrypt scheme.

1. Give a description of the encryption and decryption algorithms in the Perl scripts.
2. How are the keys handled by the scripts? What are the implications for security?
3. Ciphertext unforgeability is the notion that, even given legitimately encrypted example ciphertexts, an attacker should not be able to come up with new ciphertexts that decrypt properly. Let `badencrypt2.pl` and `baddecrypt2.pl` be identical to `badencrypt.pl`, `baddecrypt.pl` except that a different value for `$key` is used. After running `perl badencrypt2.pl` on a message, then the following ciphertext was produced

   ```
   53616c7465645f5f9c50424afaa7094038462d73f40fd1fbf933bdf8
   d80b602bf79d276302b76dad700738a842d36c2bfc84afe20a2ca230
   ab8dbb95a5e0164a925e97bd8f4cad5d9634f2fbd12e7a59a0b40d1f
   f49f1c71f803dce74a9f4c73
   ```

   (The line breaks are just for readability, it is one contiguous string of hex values.) Provide a ciphertext that differs from this one, but for which `baddecrypt2.pl` returns "Message received!". Put your ciphertext in your main document, and also in a separate text file `ciphertext.txt`. We will test it by diffing it with the ciphertext above and piping it to `paddecrypt2.pl`. Describe why the modification to the ciphertext goes undetected.
4. In a plaintext recovery attack, the attacker attempts to recover the message encrypted within a ciphertext. Describe an algorithm that, given access to `baddecrypt2.pl`, recovers some or all of the plaintext encrypted in the ciphertext above.
5. (Extra credit) Implement your attack in the form of a script (e.g., in Perl, Python, or shell) that takes no inputs and outputs the message. It can assume that it will be run in the same directory as `baddecrypt2.pl`. The script should be runnable on the lab machines, though let us know if you have some need for a different setup.