Paxson
Spring 2017

CS 161
Computer Security

Discussion ∞

Additional Pseudo-Section Material

Instructions. We will break into groups to discuss the following questions. Please think of as many solutions as you can. Be original! Maybe you will come up with something no one has thought of yet. Be prepared to talk about your solutions with the rest of the section.

This material is from past CS161 sections and is intended to aid in studying the final material presented in the Spring 2017 offering which was not covered by section, homeworks, or projects. You are *not* responsible for any additional specifics present in these materials but not present in the corresponding lecture materials.

Question 1 ***Worm Spread*** **(10 min)**

(a) In class we have seen that typical network worms propagate using scanning. Can you think of other ways to spread a worm?

(b) The typical virus exploits a benign application to execute its own (malicious) code. Exploiting real world applications is getting tougher every year because of the mitigations for buffer overflows that we discussed. Can you think of a way that a virus would not require an exploit to achieve code execution?

Question 2 ***Botnet C&C*** **(10 min)**

Consider the use of Twitter for botnet command-and-control. Assume a simplified version of Twitter that works as follows: (1) users register accounts, which requires solving a CAPTCHA; (2) once registered, users can post (many) short messages, termed *tweets*; (3) user A can *follow* user B so that A receives copies of B's tweets; (4) user B can tell when user A has decided to follow user B; (5) from the Twitter home page, anyone can view a small random sample (0.1%) of recent tweets.

(a) Sketch how a botmaster could structure a botnet to make use of Twitter for C&C. Be clear in what actions the different parties (individual bots, botmaster) take. Assume that there is no worry of defensive countermeasures.

(b) Briefly describe a method that Twitter could use to detect botnets using this C&C scheme.

(c) How well will this detection method for Twitter method work?

(d) Briefly discuss a revised design that the botmaster could employ to resist this detection by Twitter.

Question 3 ***Botnet Command and Control*** **(7 min)**

(a) Consider a botnet that uses HTTP for its command and control channel. A bot contacts `http://foobar.com` to get the latest commands from the botmaster. How could law enforcement take over this botnet?

(b) The botmaster switches to HTTPS to prevent the above attack from law enforcement. Now when a bot makes a request over TLS, it checks that the server responds with a certificate signed by the botmaster's own CA. Will this protect the botnet from law enforcement?

(c) To further defend against law enforcement, the botmaster changes the bot code so that instead of hard-coding in `foobar.com`, the bot has dozens of domains hard-coded in. The bot will try a bunch of names in the list until it finds one that is registered and has a certificate signed by the botmaster. Now what can law enforcement do to take down the botnet?

(d) How can you improve the above scheme to make it even more resilient to attack by law enforcement?

(e) Can you think of a scheme whereby the botmaster can push out updates and commands to a very large botnet *without* using DNS, and that works without the bots having *any* information about the location of other bots or elements of the C&C infrastructure? You can assume that the bots have wired into them the public key for the botmaster's CA.

(f) Discuss the pros and cons (from the botmaster's point of view) of each of the following botnet topologies:

- Star
- Hierarchical
- Peer-to-peer

Question 4 ***Another Use for Hash Functions*** **(15 min)**

The traditional Unix system for password authentication works more or less like the following. When a user u initially chooses a password p, a random string s (referred to as the "salt") is selected (and kept secret) and the value $r = H(p \mid\mid s)$ is computed, where H is a cryptographic hash function. The tuple (u, r, s) is then added to the file `/etc/passwd`. When some user later attempts to log in by typing a username u' and password p', the system looks for a matching entry (u', r', s) in `/etc/passwd` and checks that $H(p' \mid\mid s) = r'$.

(a) In this system, what do you suppose the purpose of the hash function H is? Why not just store (u, p) directly in `/etc/passwd` without computing any hashes? Is there an advantage in terms of security?

(b) What do you suppose the purpose of the "salt" s is? Why not just compute $r = H(p)$ and store (u, r) in `/etc/passwd`?

Question 5 ***El Gamal and Chosen Ciphertext Attacks*** **(9 min)**

The lecture notes explain El Gamal encryption as follows. The public parameters are a large prime p and an integer g such that $1 < g < p - 1$; these values are known to everyone. To generate a key, Bob chooses a random value b (satisfying $0 \leq b \leq p-2$) and computes $B = g^b \bmod p$. Bob's public key is B, and his private key is b. If Alice has a message m (in the range $1 \ldots p-1$) for Bob that she wants to encrypt, she picks a random value r (in the range $0 \ldots p-2$) and forms the ciphertext $(g^r \bmod p, m \cdot B^r \bmod p)$. To decrypt a ciphertext (R, S), Bob computes $R^{-b} \cdot S \bmod p = m$.

(a) Suppose you intercept two ciphertexts (R_1, S_1) and (R_2, S_2) that Alice has encrypted for Bob. Assume they are encryptions of some unknown messages m_1 and m_2, and that you have Bob's public key (but not his private key). Show how you can construct a ciphertext which is a valid El Gamal encryption of the message $m_1 \cdot m_2 \bmod p$.

(b) Show how the above property of El Gamal leads to a chosen ciphertext attack. That is, assume you are given an El Gamal public key B and a ciphertext (R, S) which is an encryption of some unknown message m and that you are furthermore given access to an oracle that will decrypt any ciphertext other than (R, S). Based on these things, compute m.

Question 6 *Side Channels* (7 min)

A side channel is a channel that leaks information due to the physical implementation. It's a *side* channel in the sense that it is not a theoretical weakness in a system, but rather an effect of its physical implementation. Side channels do not involve two cooperating parties; they instead are used by a single party to extract information they are not meant to have.

(a) Consider implementing the RSA cryptography algorithm. The typical way is to go through the 'key' bit by bit. The pseudo-code looks something like this:

```
foreach (bit in key) {
  if (bit) {
    // do multiplication and all hard work if bit is 1
  }
  // do other simpler stuff that you need to do regardless
}
```

Recall the cable box with a tamper resistant private key inside it that Prof. Paxson talked about in the lecture. Can you imagine a side-channel attack on the above implementation to find the private key? HINT: Can you do something with a multimeter?