Paxson
Spring 2013

CS 161 Computer Security

Discussion 14

May 1, 2013

Question 1 ***Side and Covert Channels*** **(7 min)**

(a) What is the difference between side channels and covert channels?

Solution: 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 covert channel is a channel that allows information transfer between two cooperating parties that aren't supposed to be able to communicate.

(b) 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?

Solution: The length of time the power is at its peak can give you a clear indication of the bit pattern of the key. Multiplication usually requires more power, and this is noticeable in embedded systems. For example, see the graph on Wikipedia [1].

Question 2 ***Countering Spam*** **(7 min)**

What technical approaches can you think of to (1) block spam (2) detect spam? Is it possible to completely stop spam? Why or why not?

Solution: For blocking spam (i.e., preventing its initial delivery), a standard approach is to create a list of spammy IP addresses and reject all email coming from those IPs. Such an IP "block list" can significantly reduce spam delivery success, but botmasters often counter it by collecting vast numbers of newly compromised machines (bots) with fresh IP addresses.

Another technique deployed today is Domain Block lists: a list of spammy domains (e.g., `canadianpharmacy.com`) is created and any message that includes mention of a listed domain is treated as spam.

Both these approaches suffer from the limitations of using blacklists (versus whitelists) that we discussed earlier in the course.

A common approach for detecting spam based on the content of a message is to use machine learning to classify messages as likely spam and not spam. Note that the base rate of "bad-things" (spam) can sometimes be nearly an order of magnitude larger than for "good-things" (sometimes termed *ham*). This situation differs from the detection scenarios we talked about in class earlier.

Spammers can counter these approaches in multiple ways. If they can acquire access to an "oracle" that tells them the classification of a particular message as spam or ham, they can repeatedly revise their messages until they become classified as ham. For example, the spammer can readily get an oracle for the spam filters employed by GMail, Hotmail etc., by just opening a new account. (Note, though, that these services also emply *counter-intelligence* by which they attempt to determine that one of their users appears to be trying to use their filter as an oracle!)

Another attack is for spammers to increase the false positive rate of the classifier. They can send a large amount of spammy emails with "useful words" in them, so that the classifier learns to match these good words with spam. For example, the spammer can send a large number of clearly spammy messages that have the word `cs161` at the bottom of the mail. The classifier will learn that the word `cs161` is often associated with spam and could start classifying all messages with `cs161` in them as spam. This would increase the false positive rate of the classifier, and may make it useless. The user then might have to just turn off the classifier. (See the paper by Nelson et al. for more details [2].)

Spammers are not known to widely use such an approach, but they *do* use a related technique of larding their messages with hammy words (often encoded so that a human reading the message doesn't actually see them and become distracted by them) in order to boost the apparent "hamminess" of their messages.

Question 3 *Externalities* (10 min)

In lecture, Prof. Paxson introduced the term *externality*. This is a term from economics. It means the cost of decision is borne by people other than those taking the decision. For example, in the case of botnets, the 'costs' include (amongst others) DoS attacks and spam. Arguably, they are caused by the decision of users not using secure/updated systems (the decision), which imposes a cost on others.

(a) Consider all the security issues we have talked about. Which can you phrase in terms of externalities?

Solution:

Many examples are possible. Here's one example: *buffer overflow* still represents a significant problem for software security today. Buffer overflows can come about due to lack of developer education, lack of testing, lack of security audits, or sloppy coding by inexperienced developers,. However, they can also arise because preventing them *costs more* in terms of requiring greater software development effort.

Observe that some creators of software (be it a corporation or just one developer) will skip rigorous testing and audits because the cost of a buffer overflow is nearly always borne by the *user* of the software and not by the developer. (An exception is the case where the flaw is so widely exploited that it leads to bad publicity for the company, a fate that Microsoft particularly suffered with the arrival of the modern worm era and the accompanying press coverage of different outbreaks.) As a result, the developers lack incentives to create software hardened against buffer overflow attacks.

(b) A common solution to externalities is *regulation*. Discuss possible solutions to the externalities you noted above.

Solution:

Regulation can help by transferring the cost of externalities back to the person making the decision. For example, *liability* laws that required software developers to pay for damages caused by insecure software would provide them with major incentives to perform more rigorous testing and security audits.

References

[1] Power attack. https://secure.wikimedia.org/wikipedia/en/wiki/File:Power_attack.png.

[2] Blaine Nelson, Marco Barreno, Fuching Jack Chi, Anthony D. Joseph, Benjamin I. P. Rubinstein, Udam Saini, Charles Sutton, J. D. Tygar, and Kai Xia. *Misleading learners: Co-opting your spam filter*. Springer, 2009. https://www.truststc.org/pubs/723/misleading.learners.pdf.