Paxson
Spring 2011

CS 161
Computer Security

Discussion 12

April 27, 2011

Question 1 ***Botnets*** **(7 min)**

We discussed in class how botnets pose a major threat on the Internet today. Discuss possible measures that we could employ to secure the Internet from the threat of botnets. Try to organize the concrete ideas from your discussion into higher-level themes. You might find it useful to think of these in terms of prevention, detection, and remediation.

Solution:

1. Angle #1: Detection / Cleanup

   Detecting infection of individual bots can help against the botnet threat. Standard tools like anti-virus, intrusion detection, and firewalls can help either block initial infections or detect their presence. Some tools use a *behavioral* approach and attempt to detect that a bot is engaging in C&C, rather than the infection itself.

   Cleanup tools today vary in their efficacy. A well-written bot can often subvert your computer in such a way that nothing short of a clean install suffices to purge it of infection.

   Unfortunately, the owners of the compromised machine often have little incentive to go through this trouble. Often, being a bot does not really pose a problem for them. Frequently the cost of being a bot (e.g., sending spam or launching DDoS attacks) is borne by others on the network. We can imagine that the current situation could change drasticall if users become legally liable for being part of a botnet.

   Microsoft cleans up millions of infections each month using their *Malicious Software Removal Tool* (MSRT), which runs on all modern versions of windows. Microsoft pushes out new signatures to the tool; when it spots executables matching the signatures, it attempts to remove them. The use of MSRT has presented enough of a barrier for malware writers that today botmasters often release updates to their bots just before the next MSRT signature update, so that the bots change their signature and avoid removal.

2. Angle #2: Take down the C&C systems / Go after the botmasters

   These actions can have very major disruptive effects on botnets, but are not easy to pursue. Faced with the possibility of takedown, botmasters continue to

innovate to make their C&C designs more resilient. Pursuing the botmasters themselves is highly challenging due to the easy of maintaining anonymity on the Internet. Furthermore, there is not a strong legal framework with sufficent international jurisdiction, and thus prosecuting people who commit cybercrimes in other countries is often very difficult. That said, recently there have been some successes.

3. Angle #3: Prevention

   Bots require installing new executables or modifying existing ones. One way to counter this is to employ a whitelist of "allowed applications" that the OS enforces. Doing so has significant costs in terms of lost flexibility, however. But note that the wildly popular iOS platform (iPhone) follows exactly this model. It is not clear whether this can be made workable for home computing systems, but for business systems this is sometimes a viable approach.

   Operating System support for some of the principles we talked in class—*least privilege*, *privilege separation*, and/or capabilities—can help. But this approach can be highly expensive in terms of requiring redesign of the OS and numerous applications.

Question 2 ***Countering Spam*** **(6 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 numberes 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 at the beginning of the class.

The common approach for detecting spam bsed on the content of a message is to use is machine learning to classify messages as likely spam and not spam. Note that

the base rate of "bad-things" (spam) is 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 [1].)

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*** **(6 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 rep-

resents 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] 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.