Nick Weaver
Fall 2018

CS 161
Computer Security

Discussion 12

November 26, 2018

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?

> **Solution:** This question is quite open-ended and has multiple solutions. One solution is to post links to friends via email/Twitter/Facebook. Another is to read logs or other files on the local machine to find other likely victims. A third is to use a search engine ("outsourcing" the scanning). A fourth is to contact a server whose job it is to track other servers (for example, some online games have "meta-servers" that you can contact to find a list of servers available for playing the game).

(b) Bitcoin (and most other cryptocurrencies) use a peer-to-peer gossip network to communicate. In a gossip network, each node has a list of peers. Whenever the node receives a message, it "gossips" the message to all of its peers. This process repeats recursively until the message reaches the entire network–typically within seconds. Why would a memory safety bug in the Bitcoin client's networking code be so deadly?

> **Solution:** An attacker could create a worm, infect everyone's clients and steal everyone's monies.

(c) 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?

> **Solution:** The virus could fool a user to download "critical updates" through social engineering. The *Koobface* worm spread using this technique. Once a user's computer is infected with Koobface, it would post a link on the user's friends' walls. When clicked, the linked page would ask the user to download and install an "update" for their Adobe Flash Player. If the "update" is installed, the computer is now infected with Koobface.

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.

Solution: A simple approach is that the botmaster registers two Twitter accounts, A and B. (This requires only solving two CAPTCHAs, trivial to do by hand.) A is used to send commands, and B for receiving commands. The bot malware includes within it the credentials for the B account. New bots then access the B account to read the tweets sent by the A account, which encode the instructions to the bots.

An alternative approach would be to create a new account per bot. Doing so requires some method for solving all of the required CAPTCHAs; for example, by purchasing solutions from a solving service available from the underground economy.

A third approach would be for bots to directly use the account of any user on an infected machine who already has an account.

A fourth approach would be for the botmaster to register a single account and use it to generate 1000s of identical tweets for each command they want to send to their bots. Each bot visits the Twitter home page and examines the random sample of tweets there to look for instructions.

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

Solution: For the first scheme, Twitter could look for access to the same account from many different IP addresses.

For the second scheme, Twitter could look for accounts whose followers all only follow that account.

For the third scheme, Twitter could filter out tweets from their random sample if the tweet is identical to a previous tweet.

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

Solution: The first scheme likely works well—it would be unusual for a legitimate account to be accessed from 1,000s of different IP addresses.

The second scheme might run into trouble for certain types of users, say celebrities, who have many followers who only use Twitter to follow the celebrity.

The third scheme should work well—there's not much value in having an identical tweet show up in the random sample that's made public.

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

Solution: For the first defense, the botmaster will need to shift to using one of the other two approaches given in part (a) above.

For the second defense, the botmaster could have the bots follow some other randomly selected users in order to look more normal.

For the third defense, the botmaster could add some minor variation to their repeated tweets so that Twitter doesn't view them as identical.

Question 3 ***Censorship and Anonymity*** **(10 min)**

(a) You are a resident of the country of Censorshipistan (a former *Eastern Block* state). You suspect that your country is employing an *on-path* censorship device to block content deemed objectionable by the ruling party. How might you detect that you are being censored?

> **Solution:** The key consideration here is that the device is *on-path*, meaning that the device lacks the ability to directly block inbound packets; instead, it needs to inject new packets. We can leverage this behavior to detect censorship behavior. For example, if we observe a RST packet followed closely after by a normal payload packet, then that behavior is highly unusual if both were indeed sent by the same sender (since after sending the RST, the sender should be all done), but often unavoidable for an on-path device that's injecting additional packets but can't prevent the transmission of the original packets. Similarly, observing multiple DNS responses, or SYN+ACK packets in addition to RSTs in response to initial SYNs, also strongly suggest the presence of an on-path injector.
>
> This behavior has been used to study the prevalence of on-path injectors in the wild. See for example the paper http://www.icir.org/vern/papers/reset-injection.ndss09.pdf if you're curious about a study based on this technique.
>
> A completely different approach can be used if the censor does not take care to vary the properties of the injected packets. For example, if all injected RST's had the same TTL or TCP "advertised window", you could filter out such packets in order to ignore them. Studies of the Great Firewall have found such behavior in some circumstances.

(b) After determining that you are indeed being censored, you decide to evade the censorship using the Tor anonymity software.[1] How might the government of Censorshipistan detect your Tor usage, and block it?

> **Solution:**
>
> The government might attempt to detect usage of Tor via one of several possible ways: (1) observe downloading of the software; (2) observe access to the servers that supply information about the Tor nodes (necessary for the client's Tor software to set up the onion path; see below); (3) observe access to those nodes themselves (first step of the onion path); or (4) look for some peculiarity of Tor traffic that occurs dynamically as Tor's being used. This latter could be based on detecting the certificates Tor uses for TLS connections, or distinct timing patterns in its message transmissions.

[1] https://www.torproject.org/

These approaches all have corresponding blocking approaches:

- Block the users' ability to download the software, either by using DNS reply injection when users try to access www.torproject.org, or by blocking the IP address(s) that that server uses.
- Block the users' ability to obtain a list of Tor nodes. The "onion routing" approach used by Tor requires that clients obtain a list of all of the Tor network's nodes so that the client can decide which hops through the network its traffic will take. Clients do so by accessing one of several (mirrored) "directory servers". The censor can block access to these similar to blocking access to the software itself.
- Block the users' access to the Tor nodes. The censor can act like a Tor client and download the list of nodes from the directory, then install the corresponding IP addresses in a blacklist.
- Upon dynamically detecting use of Tor, disrupt such connections as they occur, for example by injecting TCP RSTs.

Question 4 ***Trusting Trust*** **(5 min)**

The title of the Ken Thompson talk that Nick mentioned in class was "Reflections on Trusting Trust."

(a) Think of your daily computer usage, and list down the corporations that you *have* to trust to keep your private data private. Remember that trust is transitive: if you trust EvilCorp, you also have to trust corporations that EvilCorp relies on.

Solution:

This is again an open-ended question. One of the examples we talked about in class was your phone conversation on your iPhone. You have to trust the carriers and the maker of the phone. In addition, you have to trust the corporation that assembled the phone (Apple only designs the iPhone). You have to trust Apple too because it is Apple's software on the phone. In addition, you have to trust the manufacturer of the speaker, the microphone, the camera, the GPS, and so on too.

Interestingly, one of the conclusions that Ken Thompson mentions in his lecture is that you shouldn't trust him or companies that hire him. It's illuminating to go through the list of things you have to abandon if you don't trust Thompson: it gives a clear insight on the massive influence he has had on our daily lives. His Wikipedia page (http://en.wikipedia.org/wiki/Ken_Thompson) is a good place to start.