Paxson Spring 2013 — CS 161 Computer Security — Homework 4

Due: Monday Apr 29, at 10:00PM

Version 1.1 (Apr 19)

Instructions. Submit your solution electronically *via your class account* by Monday Apr 29, at 10:00PM. You should upload a single file, `HW4.pdf`. Your writeup should include your name, your class account name (e.g., `cs161-xy`), your TA's name, your discussion section, members of your study group (if any; **see below**), and "HW4" prominently on the first page. Use a legible font and clearly label each solution with the problem/subproblem to which it belongs. You *must* submit a PDF file; we will not accept other formats.

You can work on this homework in study groups of up to four people; however, you **must** write up the solutions *on your own*. You must never read or copy the solutions of other students (or from online materials), or co-develop writeups, and you must not share your own solutions with other students. You must explicitly acknowledge everyone who you worked with or who has given you any significant ideas about the homework.

Note: a number of these problems look back to topics addressed earlier in class. Keep in mind that the final exam will be comprehensive across all topics. Also, note that the material for Problem 4 won't be covered until the lecture on Thursday Apr 25.

Version 1.1 changes: in Problem 1, clarified to assume that the particular procedure used by the attacker to dump the user's settings is not known, and therefore there's no signature available to detect it. Clarified that DNS reverse lookups are not uncommon in operational network traffic. In Problem 2, clarified that the attack to consider in the second part of (b) refers to the earlier attack discussed in (a). In Problem 3(b), relaxed the requirement regarding the false positives arising from the alternative architecture.

Problem 1 *Sniffer Detection* (20 points)

As the security officer for your company, your network monitoring has observed a download of a "sniffer" tool. This tool passively eavesdrops on traffic. Whenever it sees a web session, it looks up the hostname associated with the server's IP address.[1] If the sniffer sees a session going to a server in a `*.yahoo.com` domain, it extracts the user's session cookie. It then uses the cookie to create a new connection that automatically logs in as the user and dumps their `*.yahoo.com` settings.[2] Assume that the particular approach

[1] Such *reverse lookups* are done using special DNS resource records, called `PTR` RRs. The term "reverse" is used because the usual functionality we associate with DNS lookups is to map hostnames to IP addresses. Here, it goes the other way, mapping IP addresses to hostnames. Note that such reverse lookups are not particularly uncommon to see in operational network traffic.

[2] Capturing and reusing session cookies is known as *sidejacking* or *session hijacking*.

by which the attacker dumps the settings is not known, and thus there's no signature to employ to recognize the associated network traffic.

You become concerned that one of your employees may have installed a network "tap" somewhere among the hundreds of links inside your building, and will use it to run this tool.

(a) Sketch **two** approaches you could use to detect whether the sniffer is indeed running inside your network.

(b) For each approach, discuss whether an attacker could take additional steps to evade detection by your approach.

Problem 2 ***DNSSEC*** **(20 points)**

DNSSEC (DNS Security Extensions) is designed to prevent network attacks such as DNS record spoofing and cache poisoning. When queried about a record that it possesses, such as when the DNSSEC server for `example.com` is queried about the IP address of `www.example.com`, the DNSSEC server returns with its answer an associated *signature*.

For the following, suppose that a user R (a resolver, in DNS parlance) sends a query Q to a DNSSEC server S, but all of the network traffic between R and S is visible to a network attacker N. The attacker N may send packets to R that appear to originate from S.

(a) Suppose that when queried for names that do not exist, DNSSEC servers such as S simply return "No Such Domain," the same as today's non-DNSSEC servers do. This reply has no associated signature.

Describe a possible attack that N can launch given this situation.

(b) Suppose now that when queried for a name Q that does not exist, S returns a signed statement "Q does not exist."

1. Describe a DoS attack that N can launch given this situation.
2. Describe a circumstance under which N can still launch the attack you sketched in part (a) above; or explain why this attack no longer works.

(c) One approach for addressing the above considerations is to use NSEC Records. As mentioned in class, when using NSEC S can return a signed statement to the effect of "when sorted alphabetically, between the names N_1 and N_2 there are no other names." Then if the name represented by the query Q lexicographically falls between N_1 and N_2, this statement serves to inform R that indeed there's no information associated with the name in Q.

We discussed in class how NSEC has a shortcoming, which is that an attacker can use it to *enumerate* all of the names in the given domain that do indeed exist. To counter this thread, in the April 10 section materials we introduced the NSEC3 Record, which is designed to prevent DNS responses from revealing unnecessary information. NSEC3 uses the lexicographic order of *hashed* names, instead of their

unhashed order. In response to a query without a matching record, NSEC3 returns the hashed names that come just before and just after the hash of the query.

Suppose that the server S has records for `a.example.com`, `b.example.com`, and `c.example.com`, but *not* for `abc.example.com`. In addition, assume that `a.example.com` hashes to `eee60f2e...`, `b.example.com` to `0a8426cb...`, `c.example.com` to `bb6a0821...`, and `abc.example.com` to `b0bb319f...`.

If the query Q from R is for `abc.example.com`, what will S return in response? Describe how R uses this to validate that `abc.example.com` indeed does not exist.

(d) In more detail, the way the hashes work in NSEC3 is they are computed as a function of the original name plus a *salt* and an *iteration parameter*, as follows:

```
Define H(x) to be the hash of x using the Hash Algorithm selected by
the NSEC3 RR, k to be the number of Iterations, and || to indicate
concatenation.  Then define:

   IH(salt, x, 0) = H(x || salt), and

   IH(salt, x, k) = H(IH(salt, x, k-1) || salt), if k > 0

Then the calculated hash of a name is

   IH(salt, name, iterations)
```

In an NSEC3 reply, the name of the hash function, the salt and the number of iterations are also included (that is, they are *visible* and assumed to be easily known). All replies from a given server use the same salt value and the same number of iterations.

Suppose an attacker has a list of "names of interest," i.e., names for which they want to know whether the given name is in a particular domain. If the attacker can get all of the NSEC3 responses for the particular domain, can they determine whether these names exist? If so, sketch how. If not, describe why not.

(e) What is the purpose of the *salt* in NSEC3 replies?

(f) What is the purpose of the *iteration parameter* in NSEC3 replies?

(g) The specification of NSEC3 also sets an upper bound on the *iteration parameter*. What threat does that protect against?

Problem 3 *Detecting Worms* (20 points)

Assume that you are working for a security company that has to monitor a network link for worm traffic. The link connects a large site with the rest of the Internet, and always has lots of traffic on it.

Your company sells a monitoring box that can scan individual packets for fixed strings at very high speeds.

(a) Suppose that after careful analysis you discover that a particular TCP-based worm that you need to protect against generates traffic that always contains a fixed 4-byte sequence. You program your company's specialized hardware to generate an alarm whenever it detects a packet containing this 4-byte pattern.

Explain how this detector can exhibit false negatives.

(b) Propose an alternative architecture for your company's monitoring box that fixes the problem from part (a). Your alternative should not increase false positives by more than a modest amount. Does your revised approach completely eliminate false negatives? Explain why or why not.

(c) Suppose benign network traffic is uniformly distributed in terms of its content. That is, every payload byte is chosen uniformly at random. Calculate the expected time until a 100 MB/sec (megabytes per second) link will cause a false alarm.

(d) Assume now that the signature length is 8 bytes. Calculate the expected time until a 10 GB/sec (gigabytes per second) link will cause a false alarm.

(e) Explain a technique that worm authors can employ to try to ensure that their worms do not have many fixed-byte sequences.

Problem 4 ***Covert Channels*** **(20 points)**

Consider a highly secured operating system that runs jobs for multiple users, but tries to assure strict isolation between the jobs, i.e., the jobs have no means to communicate with one another. The OS not only prohibits use of shared memory and any other forms of inter-process communication (pipes, sockets, shared memory, signals, use of the `ps` command), but also eliminates shared resources such as a global file descriptor pool.

The OS does, however, provide read-only access to a common file system. This file system does *not* make available "access time" information.

(a) Sketch how two cooperating processes could use the common file system to create a covert channel for communication. You can assume that the OS tries to optimize performance when accessing files by caching recently accessed blocks.

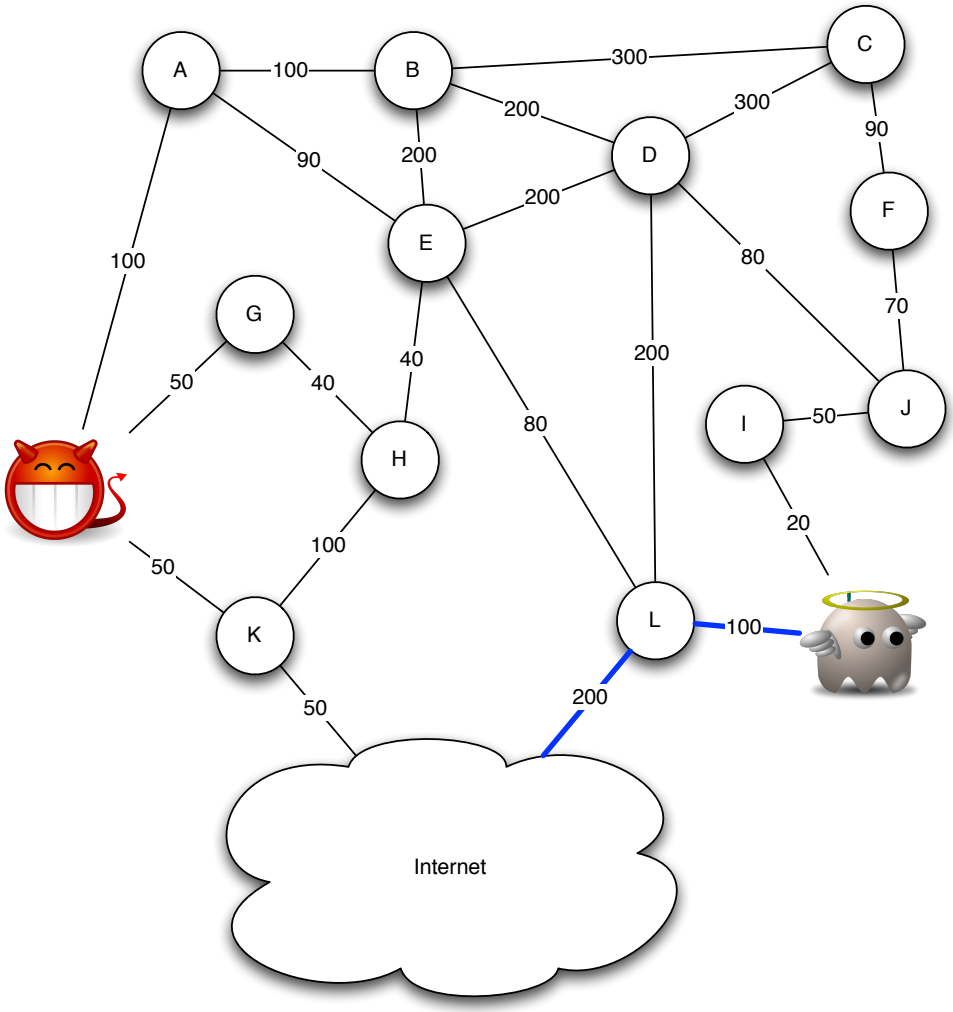
(b) In rough terms, what is the capacity of your covert channel? State any assumptions you make in your estimate.

(c) Suppose the OS eliminates the common read-only system in its entirety, though retains the performance technique of caching recently accessed blocks. Can two cooperating processes still communicate? If so, sketch how, and again estimate in rough terms the capacity of the channel. If not, then explain why communication is no longer possible.

Problem 5 ***Denial-of-Service Attacks*** **(20 points)**

The figure below shows the network of *Billy Bob's ISP*, a small regional ISP that contains only a dozen routers and two upstream connections to the rest of the global Internet. Each alphabetically labeled node corresponds to a router. The links between the nodes represent the maximum capacity in megabytes/sec (MBps). For example, the maximum bandwidth between G and H is 40 MBps. Assume that if an attacker can fill such a link with traffic at that level in either direction, then the link becomes unusable for regular (benign) traffic. In addition, assume that each node runs a full set of Internet services.[3]

The two players of interest are the attacker on the left and the victim on the right.



To route traffic between two nodes, the nodes always choose the shortest path according to the number of hops.[4] The routing is thus independent of the link capacity. An

[3] This situation is quite abstracted from how things work in practice. Topologies are rarely as graph-like as shown here, and routers do not run numerous Internet services.

[4] If there is a "tie" between two paths of the same length, a router resolves it by picking the path for which the next hop comes first alphabetically. For example, to get to node H the attacker will use the path

example routing from B to J would be the path B-D-J. In addition, assume that routing does not transit the "Internet" cloud in the diagram (that is, it won't go into the cloud and back out).

(a) The Internet connection of the victim is highlighted with bold links (also, blue, if viewed in color). Is there any means by which the attacker at the other end of the network can deny service to the victim in terms of cutting the victim off from the global Internet by saturating the link from the victim to router L?

(b) Suppose that the victim's web server infrastructure can process 1,000,000 TCP connection requests per second. Assume that a single connection request (TCP initial SYN) requires 40 bytes to transmit. Can the attacker launch a sufficiently large SYN flood to overwhelm the victim's server? Explain why or why not.

(c) Suppose the victim configures routers I and L to block all incoming TCP packets with only the SYN bit set, but to allow inbound TCP packets with both SYN and ACK set. This denies all inbound connection requests to the victim. Can the attacker launch a SYN flood attack against the victim? If not, is the victim now safe from any type of TCP DoS?

(d) The *Slashdot effect* (also termed "flash crowds") is an unintended DoS attack by a massive number of visitors that click on a popular link in a news article, which causes them to load pages from a web server whose infrastructure lacks sufficient capacity to accommodate the large number of visitors. This popularity overloads the small site by either clogging its available bandwidth, or consuming all of the web server's resources.

After you graduate, your incredibly cool startup, `v.k001.com`, becomes *slashdotted*. The Slashdot article contains a link to `http://v.k001.com/kewlness`, a page on one of your company's lightly-provisioned web servers. Sketch **two** approaches by which you could leverage this incoming traffic to launch a DoS attack against someone else's website.

through G rather than K. Here we have again abstracted the network from how things actually work in the Internet.