Paxson
Spring 2017

CS 161 Computer Security — Homework 5

Due: April 28, 2017, at 11:59pm

(Version 1.0: released April 21, 2017)

Instructions. This homework is due April 28, 2017, at 11:59pm. You *must* submit this homework electronically via Gradescope (not by any other method). When submitting to Gradescope, *for each question* your answer should either be a separate file per question, or a single file with each question's answer on a separate page. This assignment must be done on your own.

Problem 1 ***Denial-of-service via email*** **(10 points)**

One day you try to email ten of your friends at Stanford your award-winning recipe for Big Game game-day cookies to their `stanford.edu` addresses. Unfortunately you typed two of their email addresses incorrectly so you received two failed-delivery notification (FDN) messages that include both the original text from the email you sent and copies of the attachment.

(a) Knowing this, how could you launch an amplified denial of service attack against some poor unsuspecting person via email?

(b) How could the developer of the Stanford mail server mitigate this issue?

(c) Sketch another form of email-based DoS attack that works even if servers all employ the fix you sketched in the previous part.

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 is not signed.

Describe an 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. Can N still launch the attack you sketched in part (a)? If so, explain how this attack would work. If not, explain why the attack no longer works.

(c) One approach to address 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 confirm to R that there's no information associated with the name in Q.

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 threat, in the April 17 section we introduced the NSEC3 Record, which is designed to prevent DNS responses from revealing other names in the domain. NSEC3 uses the lexicographic order of *hashed* names, instead of their unhashed order. In response to a query without a matching record, NSEC3 returns a signed statement of the hashed names that come just before and just after the hash of the query.

Suppose that the server S has records for `a.cs161.com`, `b.cs161.com`, and `c.cs161.com`, but *not* for `abc.cs161.com`. In addition, assume that `a.cs161.com` hashes to `dee60f2e...`, `b.cs161.com` to `80a4cb36...`, `c.cs161.com` to `c218f96a...`, and `abc.cs161.com` to `99f3e2ba...`.

If R queries S for `abc.cs161.com`, what will S return in response? Describe how R uses this to validate that `abc.cs161.com` does not exist.

(d) The hashes in NSEC3 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)
```

The name of the hash function, the salt and the number of iterations are all included in an NSEC3 reply (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 that domain, can they determine whether these names exist? If so, sketch how. If not, describe why not.

(e) Why would a domain change their *salt* value 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 could happen if that upper bound did not exist?

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 150 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 20 GB/sec (gigabytes per second) link will cause a false alarm.

(e) What could a worm author do to try to ensure that their worms do not have many fixed-byte sequences?

Problem 4 ***Feedback*** **(0 points)**

Optionally, feel free to include feedback. Whats the single thing we could do to make the class better? Or, what did you find most difficult or confusing from lectures or the rest of class, and what would you like to see explained better?

If you don't want to submit any feedback, just select your last page to use for this question when submitting to Gradescope.