Raluca Ada Popa
Spring 2018

CS 161 Computer Security

Homework 3

Problem 1 *TLS* (20 points)

Alice wants to communicate with `smallfish.com` using TLS. Assume the browser and server use RSA-based key exchange (not Diffie-Hellman). Each part is independent. Assume that the cipher-suites chosen in the TLS connections **do not** make use of TLS sequence numbers. Assume each potential replay attack involves generating a new TLS connection.

(a) Suppose Alice downloads a buggy version of the FireFox browser that implements TLS incorrectly. The TLS specification says that, during the handshake, the browser should send a random 256-bit number R_B and the server should send a random 256-bit number R_S.[1] Instead of picking R_B randomly, it increments a counter and sends it instead. Which of the following is true?

1. If Alice visits a HTTPS URL, a man-in-the-middle can not replay that HTTP request to the server a second time.
2. If Alice visits the same HTTPS URL twice, when Alice visits that URL the second time a man-in-the-middle can not replay the HTML page that was returned by the server on Alice's first visit.
3. A man-in-the-middle can not compromise Alice's confidentiality (e.g., learn the data she sends over TLS).
4. A man-in-the-middle can not learn the symmetric MAC keys that protect data sent over TLS connections initiated by Alice's browser.
5. None of the above

Solution: 1, 2, 3, 4

Reason: Even if R_B were to repeat, Alice would choose a different pre-master secret each time and thus they'll end up with different symmetric keys for each different connection, so it's not possible for a MITM to replay an HTML page returned on a prior connection.

We didn't ask for a reason, so you get full points simply by indicating the choices.

(b) Alice downloads a patch for her buggy version of FireFox, which fixes the previous problem but introduces another bug. Now, her client generates the pre-master secret based on three items: the current absolute time, the total time Alice's computer has been powered on, and the process-ID of the current FireFox process. Which of

[1] Also recall Discussion 6, Q2 here: http://inst.eecs.berkeley.edu/ cs161/sp18/handouts/dis6.pdf

the following is true?

1. If Alice visits a HTTPS URL, a man-in-the-middle can not replay that HTTP request to the server a second time.
2. If Alice visits the same HTTPS URL twice, when Alice visits that URL the second time a man-in-the-middle can not replay the HTML page that was returned by the server on Alice's first visit.
3. A man-in-the-middle can not compromise Alice's confidentiality (e.g., learn the data she sends over TLS).
4. A man-in-the-middle can not learn the symmetric MAC keys that protect data sent over TLS connections initiated by Alice's browser.
5. None of the above

Solution: 5

Reason: The MITM can recover a low-entropy pre-master secret by brute-forcing it, infer the symmetric keys, and then all security is lost. For example, an attacker can decrypt all the traffic Alice sent over TLS, and then inject it into a new TLS session from Alice (encrypted and MAC'ed with the proper keys for that connection, which can also be recovered in the same way).

Note on replay attacks: In general, we refer to Network replay attacks as a method of attack where an exact copy of the recorded data is replayed at the victim and is not rejected/detected. This exact situation is likely not viable given that the server will generate a new R_S for each connection.

However, in this question, we specifically ask whether the 'HTTP request' or the 'HTML page' can be replayed—this implies the contents of the encrypted payload (regardless of whether the data is re-encrypted). In this context, replay is possible because one can decrypt all of Alice's/the server's traffic, and package it up in new TLS connections.

We didn't ask for a reason, so you get full points simply by indicating the choices.

(c) Alice downloads an updated version of FireFox that is finally correct. With her fixed browser, she visits `https://bluefish.com/`. That server's TLS implementation has a bug: instead of picking R_S randomly, it always sends all zeros. Which of the following is true?

1. If Alice visits the HTTPS URL, a man-in-the-middle can not replay that HTTP request to the server a second time.
2. If Alice visits the same HTTPS URL twice, when Alice visits that URL the second time a man-in-the-middle can not replay the HTML page that was returned by the server on Alice's first visit.

3. A man-in-the-middle can not compromise Alice's confidentiality (e.g., learn the data she sends over TLS).
4. A man-in-the-middle can not learn the symmetric MAC keys that protect data sent over TLS connections initiated by Alice's browser.
5. None of the above

Solution: 2, 3, 4

Reason: Symmetric keys are derived as a function of the pre-master secret, R_B, and R_S. If the MITM replays the entire handshake to the server, the pre-master secret and R_B will automatically be identical; and due to the bug in the server, R_S will be identical as well, so the same symmetric keys will be derived, and encrypted and MAC'ed data from the prior connection can be replayed a second time. Note that 'replaying the handshake' means re-sending an identical copy of the messages.

We didn't ask for a reason, so you get full points simply by indicating the choices.

Problem 2 *TCP and LAN* (20 points)

Paul has just opened his laptop and is attempting to connect to the internet to visit `www.moneydancemoves.com`.

(a) Eve sees the DHCP Discover message of the laptop and decides to interfere. What kind of DHCP message can she send at this point and what information does it include? Of the information included, **underline** the pieces of information that will allow Eve to exploit Paul's future internet connections. Assuming you do not know the timing of messages delivered in the local network, will this attack always work? Answer in no more than **two sentences**.

> **Solution:** Eve responds with a DHCP Offer Message which includes IP Address, DNS Server, gateway router, and lease time. The DNS Server and gateway router are underlined. The attack will not always work because we cannot be sure if Paul will accept Eve's offer message or the legitimate DHCP server offer message.

(b) Assume instead Eve does not interfere and Paul successfully runs DHCP to connect to the internet via unencrypted HTTP. Mallory is a man-in-the-middle of Paul's laptop and `www.moneydancemoves.com`. Mallory runs a stateless packet filter that checks each of Paul's TCP packets to see if any given packet contains the phrase "send the money". If it does, she injects a RST packet to sever Paul's TCP connection. Will Mallory always know whether Paul asks `www.moneydancemoves.com` to "send the money"?

> **Solution:** No. The phrase may be spread across multiple TCP packets (ex: packet one contains "send ", and packet two contains "the money") and a stateless packet filter cannot remember any prior packets.

(c) Eve and Mallory decide to escalate the situation to their superiors and the NSA gets involved. The NSA can leverage off-path, on-path, and man-in-the-middle attacks against Paul, but they would prefer to use the easiest attack (off-path is easier that on-path, and on-path is easier than man-in-the-middle). For each scenario, which attack will the NSA use (on-path, off-path, or man-in-the-middle)?

(i) The NSA seeks to create a UDP request to the website's server which appears to come from Paul's IP address. The NSA doesn't need to see the reply.

> **Solution:** Off-path attack, since they don't need to see anything about the server.

(ii) The NSA seeks to create a TCP connection to the website's server which appears to be from Paul's IP address. The server uses the current time to generate the initial sequence number. The NSA doesn't need to see the reply.

Solution: Off-path attack, since they can predict the initial sequence number.

(iii) The NSA seeks to create a TCP connection to the website's server which appears to be from Paul's IP address. The server uses a secure RNG to generate the initial sequence number. The NSA doesn't need to see the reply.

Solution: On-path attack, since they can't predict the initial sequence number.

(iv) The NSA seeks to inject content into an existing active TCP connection between Paul and the web server. The NSA knows Paul is paranoid and records his raw traffic, but the NSA does not want Paul to determine that the NSA has modified the traffic.

Solution: A full man-in-the-middle attack, since an on-path attacker can't stop the legitimate reply from the server. If Paul sees both the NSA reply and legitimate server reply, he will know someone is interfering.

Problem 3 ***DNS*** **(20 points)**

Outis is a hacker attempting to sabotage CS161 from a cafe. He sees Raluca walk into the cafe and tell Won that she will upload her draft of midterm 2 to the secret TA website, `www.midterm.ta_secrets.com` after she has worked on it for an hour or two. Outis knows that the local DNS server lies on the cafe network and that it may be possible to interfere with its queries. He cleverly sends Raluca a malicious link titled "IMPORTANT PROJECT UPDATE FROM KEYHAN" that she will click immediately. Clicking the link will redirect her to `www.midterm.ta_secrets.com` and cause her computer to generate one DNS query for the secret TA site. Assume that the following subproblems do not build upon each other (no information is cached from a previous attack, etc).

(a) Suppose Outis owns his own website. He can customize its appearance and see all files uploaded to the site. Explain how Outis can eventually obtain Raluca's midterm 2 draft assuming he is able to successfully spoof a DNS response. Use no more than **two sentences**.

Solution: Outis can trick Raluca into thinking that his website is the TA website by making his website identical in appearance to `www.midterm.ta_secrets.com` and then spoofing the DNS response for the TA website to return his website's IP address. When Raluca navigates to `www.midterm.ta_secrets.com` to upload her midterm she will instead navigate to Outis' website IP address and upload the file to him.

(b) Explain why Outis can spoof a DNS response now even though Raluca will likely upload the valuable midterm file in an hour or two. **Answer in a single sentence**.

Solution: Outis can set the spoofed Resource Record's "Time to Live" to a large number so that even though it poisons the cache now, it can remain cached for several hours (or days) and will be used when Raluca later navigates to the site.

(c) Outis is lazy and wants to use the easiest network attack possible to spoof the response. An off-path attack is easier than an on-path attack, and an on-path attack is easier than a man-in-the-middle attack. Suppose Outis can only generate a single fake response to the cafe's DNS server query (though his response will reliably reach the server first). What flavor of network attack will Outis use (off-path, on-path, or man-in-the-middle)? Justify your answer using no more than **two sentences**.

Solution: Outis will use an on-path attack. If he is only able to generate a single response, he needs to see the DNS request to get the port/transaction ID to generate a malicious reply - however he does not need to prevent the real response from reaching the server since his reply will come first so man-in-the-middle is not necessary.

(d) Outis realizes that due to cafe security upgrades, he can only perform off-path attacks (he cannot see or intercept network traffic). However Outis' friend Kaminsky has told him about a new attack scheme: Outis upgrades his malware to generate k forged DNS responses that will all arrive before the legitimate response. Assume the DNS query randomizes only transaction ID. Give the probability p that Outis will succeed (Outis succeeds if Raluca accepts any of the spoofed responses as valid).

Solution: $p = \frac{k}{2^{16}}$. k attempts to guess the 16 bit transaction identifier.

(e) Outis finishes reading Kaminsky's paper and realizes he can improve his off-path attack. He modifies the link in his message to Raluca ("IMPORTANT PROJECT UPDATE FROM KEYHAN") so that when she clicks it her laptop will generate m requests instead of one. The requests are for URLs: `1.ta_secrets.com, 2.ta_secrets.com, ..., m.ta_secrets.com`. Despite the fact that none of these URLs are valid, Kaminsky has told Outis that if he can correctly spoof the response to a single one of these queries, he will be able to give a malicious IP address to the higher domain *ta_secrets.com* in the "Additional" field of the response and thus by extension control `midterm.ta_secrets.com`[2]! Assuming again that Outis can generate k spoofed responses *per request*, what is the probability, p, that he succeeds?

Solution: k attempts to guess the 16 bit transaction identifier for m different requests.
The probability of success p = 1 - probability that all attempts fail, and the probability that all attempts fail is $(1 - \frac{k}{2^{16}})^m$. Thus probability of success is $p = 1 - (1 - \frac{k}{2^{16}})^m$
Additionally, an acceptable approximation: $p = \frac{k*m}{2^{16}}$.

(f) Raluca sees Outis and suspects he is up to something. She switches her laptop to a network with a DNS server that uses DNSSEC for greater security. The resolver's cache is empty and when she types `www.midterm.ta_secrets.com` into her browser the DNSSEC resolver begins to recursively resolve the name. Suppose the resolver has just received the response from the name server at `ta_secrets.com`. Whose public key will it use to verify the response, and whose public key is contained `inside` the response? Assume `midterm.ta_secrets.com` is configured to be its own zone (with corresponding name server).
NOTE: There is no need to reference ZSK or KSK - they are out of scope of this

[2]For more details on the Kaminsky attack, see the "Dan's Shenanigans" section of this link

class. Please answer with respect to what was taught in lecture.

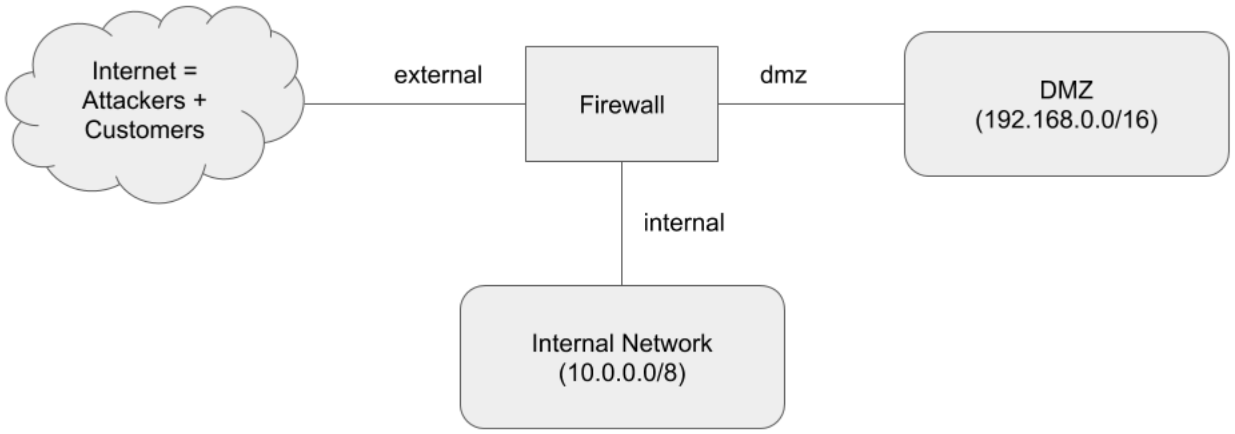
Solution: NOTE: This solution is based on the simplified version of DNSSEC taught in the Sp18 lecture (we assume there is one request and one response from each name server, rather than exchanges of Zone Signing Keys and Key Signing Keys etc.) The RRSig is verified by the public key of `ta_secrets.com`, and the key for `midterm.ta_secrets.com` is contained in the response.

(g) As discussed in lecture, DNSSEC name servers must sign responses to requests that do not exist with a cached record of the two closest domain names (this is a form of "off-line" signing called an NSEC record)[3]. Assume the only other subdomain of `ta_secrets.com` aside from `midterm.ta_secrets.com` is a secret subdomain, `thanks_won_park.ta_secrets.com`. When someone visits `thanks_won_park.ta_secrets.com` they are able to change the grades of the 161 students. How might Outis efficiently discover this secret domain? Explain in **two sentences** or less.

Solution: Outis can efficiently discover the secret domain via domain space enumeration with NSEC records. When he submits requests for domains that do not exist he will receive NSEC records giving him the alphabetically closest valid domains - it should take him at most two queries for nonexistent names above and below `midterm.ta_secrets.com` to see an NSEC record enumerating `thanks_won_park.ta_secrets.com`.

[3]Remember that if a server performs "online" signing (every time a request for a nonexistent name x comes in the server signs a message saying x does not exist) then the server is vulnerable to a DoS attack since signing is computationally expensive.

Problem 4 *Firewalls and DMZ* (20 points)


Internet = Attackers + Customers
external
Firewall
dmz
DMZ (192.168.0.0/16)
internal
Internal Network (10.0.0.0/8)

The figure above shows the network configuration of a company. The company cares very much about the information security, and has created a *demilitarized zone* (DMZ) to isolate the internal network from the public facing services hosted in the DMZ.[4]

You are tasked with writing the firewall rules for this scenario. You need to only worry about the TCP traffic. Here are the requirements:

- Deny all traffic, unless otherwise specified.
- Allow inbound web traffic from the Internet to TCP port `80` and TCP port `443` of a web server at `192.168.1.20`, which sits in the DMZ.
- Allow inbound SSH traffic from the Internet to TCP port `22` of an SSH server `192.168.3.40` in the DMZ.
- Allow all traffic from an SSH server `192.168.5.60` in DMZ to the internal network.
- Allow all outbound connection to the Internet from the internal network, except those of social networking sites with IP address blocks of `8.8.0.0/16`, `2.3.0.0/16`, and `56.78.90.0/24`.
- Allow all connections to the DMZ network from the internal network, except to IP addresses `192.168.250.0/24`.

You need to consider three separate sets of firewall rules, one for each of the interfaces (see the figure above). The interfaces are labeled as `external`, `internal` and `dmz`. The internal network is `10.0.0.0/8`; the DMZ network is at `192.168.0.0/16`; everything else is outside. Rules always apply to the packets *arriving* at an interface. For example, rules associated with the `dmz` interface apply to packets sent from the `192.168.0.0/16` network.

Use a format similar to the lecture, but annotated with the interface slightly differently. For example:

[4]Read a bit about DMZ here: https://en.wikipedia.org/wiki/DMZ_(computing)

`internal allow tcp 10.1.2.3:* -> 8.8.8.0/24:80 if ACK set`
specifies that on the firewall's `internal` interface, TCP packets arriving with a source address of `10.1.2.3` and any source port, destined for port `80` of any address in the `/24` block `8.8.8.0`, should be allowed providing they have the `ACK` bit set. Another example, that drops all the traffic from the internal network to the DMZ network, will be written as: `internal drop tcp *:* -> 192.168.0.0/16:*`

For your solution, list a separate ruleset for each interface. You should aim to keep the rules as simple (minimal) as possible. Keep in mind that the rules are applied in sequence, thus the order in which rules are processed matters. (*Hint:* You may want to put something like `<interface> drop tcp *:* -> *:*` as the very last rule of your list depending on the interface; this merely indicates that the default action is `deny`.)

Solution:

External interface:

```
external allow tcp *:* -> 192.168.1.20:80          #  1. webserver access
external allow tcp *:* -> 192.168.1.20:443         #  2. webserver access
external allow tcp *:* -> 192.168.3.40:22          #  3. SSH server access
external allow tcp *:* -> 10.0.0.0/8:* if ACK set  #  4. established conns.
external drop tcp *:* -> *:*                       #  5. default deny
```

DMZ interface:

```
dmz allow tcp 192.168.5.60:* -> 10.0.0.0/8:*       #  6. access from SSH serv
dmz allow tcp *:* -> *:* if ACK set                #  7. established conns.
dmz drop tcp *:* -> *:*                            #  8. default deny
```

Internal interface:

```
internal drop tcp *:* -> 8.8.0.0/16:*              #  9. deny social-network
internal drop tcp *:* -> 2.3.0.0/16:*              # 10. deny social-network
internal drop tcp *:* -> 56.78.90.0/24:*           # 11. deny social-network
internal drop tcp *:* -> 192.168.250.0/24:*        # 12. deny part of DMZ
internal allow tcp 10.0.0.0/8:* -> *:*             # 13. allow outbound+DMZ
internal drop tcp *:* -> *:*                       # 14. default deny
```

We didn't ask for a reason, so you get full points simply by indicating the correct rule-sets. The last part after a # is a comment, which isn't needed and is only used for explanation below.

Reasoning: Here's a way to reason about it; you are required to satisfy 6 criteria. There are various mental models to write the rules; my favorite is to start from the most specific to the least specific. Note that the rules are evaluated in order.

- Requirement 2: "Allow inbound web traffic from the Internet to TCP port `80` and TCP port `443` of a web server at `192.168.1.20`, which sits in the DMZ."

  Rule 1 and 2 take care of all incoming packets from the outside world. Rule

7 ensures that the response packets of the TCP connection can make it back. Note that the "if ACK set" specifies that this applies only to those packets that have the ACK flag set, i.e. that are part of an ongoing connection. Rule 8 ensures that no new connections can be established from the DMZ side (other than the exceptions in Rule 6), because a new connection only has a SYN flag set (not ACK).

Note that rule 7 has to go before Rule 8; similarly Rule 6 should be specified before both 7 and 8.

- Requirement 3: "Allow inbound SSH traffic from the Internet to TCP port `22` of an SSH server `192.168.3.40` in the DMZ."

  Similar reasoning as above; relevant rules are 3, 7, 8.

- Requirement 4: "Allow all traffic from an SSH server `192.168.5.60` in DMZ to the internal network."

  Rule 6 ensures that the any connection from the SSH server to the internal network is allowed. Note that in this particular situation, this SSH server acts as a client, thus the source port can be arbitrary; specifying source port 22 is incorrect. The reverse direction is handled by Rule 13.

- Requirement 5: "Allow all outbound connection to the Internet from the internal network, except those of social networking sites with IP address blocks of `8.8.0.0/16`, `2.3.0.0/16`, and `56.78.90.0/24`."

  Rule 9, 10, 11 implement the blocking of social network. Rule 13 allows for other connections to be established to the outside world. The reverse direction blocking is handled by Rule 4 and 5.

- Requirement 6: "Allow all connections to the DMZ network from the internal network, except to IP addresses `192.168.250.0/24`."

  Rule 12 takes care of the exceptions. Rule 13 ensures that the connection can be initiated. Rule 7 ensures that those initiated connections can be completed.

- Requirement 1: "Deny all traffic, unless otherwise specified."

  This is taken care of by Rule 5, 8 and 14.

Problem 5 *DDoS* (20 points)

Note that we talk about real world attacks in this question; we request you to revisit the 'Ethics' section in the online course policies before starting this question.

Let us talk about Distributed Denial of Service attacks.[5] Recently, `github.com` came under a massive DDoS attack at the rate of 1.3Tbps, which you should read about online.[6]. Just days ago, another unnamed service provider came under an even bigger attack, going as high as 1.7Tbps.[7] Using open `memcached` servers, the attackers were able to send bogus traffic to their victims and effectively *amplify* their bandwidth by a factor of as high as 51,000.[8]

The purpose of such DDoS attacks is to fill up the network bandwidth of a victim web-service with bogus traffic. When the victim has a finite network bandwidth, an attacker can prevent legitimate traffic from reaching the victim's web-service. A victim of a DDoS attack is quite helpless; using a firewall or IDS does not help free the upstream bandwidth.

The attack: DDoS attacks could be either at the network level (UDP floods, TCP SYNs), or at the application level (HTTP GET/POST); the `memcached` based amplification attack falls in the network level category. Let us calculate the potential size of a DDoS attack an adversary can launch using these `memcached` servers by *amplification. Before starting, make sure you have skimmed through the posts in the footnotes.*

(a) Imagine an attacker: a bored teenager with a modest Internet connection of 10Mbps (megabits/second). The attacker finds 10,000 open `memcached` servers on the Internet, all of which have very good bandwidth. He also discovers a standard query of size 25 bytes that all the servers respond to; the response is a fixed size of 750KB (kilobytes). Also note that after these high-profile attacks, all the `memcached` servers started using a rate limit of 10 requests/second per server for a given client IP address.

Given these constraints, what is the size of DDoS that this attacker can launch against `smallfish.com`, the website of a small local game-store that maps to an IP address `1.0.0.1`. Please provide the traffic-volume seen by the victim in units of Tbps (terabit/second), and provide a 2-3 sentence summary of how you achieved this number. Assume that 1 terabit = 10^{12} bits.

[5] Here is a fun little analogy for what it feels like to be under a DDoS attack (a scene from the movie Harry Potter and the Sorcerer's Stone): https://www.youtube.com/watch?v=yQIFkMlDF4M&t=150s

[6] Here is a good summary: https://www.wired.com/story/github-ddos-memcached/

[7] https://arstechnica.com/information-technology/2018/03/us-service-provider-survives-the-biggest-recorded-ddos-in-history/

[8] We do not require you to understand *all* the details, but please skim through the following post about `memcached`'s role in DDoS attacks; we expect you to have at least a high-level understanding of the actual attack: https://blog.cloudflare.com/memcrashed-major-amplification-attacks-from-port-11211/.

Solution: 0.3Tbps. The attacker can send up to $(10 * 10^6)/8 = 1.25 * 10^6$ bytes/second; which translates to $1250 * 10^3/25 = 50,000$ requests/second. Distributed equally, this is 5 requests/server (well within the rate-limit). Each request results in a response of size 750KB; so the estimated traffic seen by `smallfish.com` is $(750 * 10^3) * (50 * 10^3) * 8 = 300 * 10^9 = 0.3 * 10^{12}$bps

Note that the bottleneck here is the attacker's own Internet connection.

(b) Imagine the same situation for the attacker above, except that the bored teenager decided to use his school's Internet for his nefarious actions. The school Internet is 100Mbps (megabits/second).

Given the constraints, what is the size of DDoS that this attacker can launch against `smallfish.com`. Please provide the traffic-volume seen by the victim in units of Tbps (terabit/second), and provide a 2-3 sentence summary of how you achieved this number.

Solution: The situation is slightly different here, since the bottleneck is not the attacker's Internet connection anymore; it is the rate-limit put in by the `memcached` servers. At 10 responses/second, each of size 750KB from 10,000 individual servers, `smallfish.com` will see a traffic of 0.6Tbps.

(c) The attacker is still at school, and enjoying his 100 Mbps Internet connectivity. Sufficiently content with destroying the `smallfish.com`'s business, he now wants to go after `largefish.com`. The major difference between `smallfish.com` and `largefish.com` is that the latter is a much larger and heavy traffic website. In order to process the traffic, `largefish.com` uses 10 web-servers with IP addresses in the range `2.0.0.1-2.0.0.10`, and uses a DNS-based load-balancing scheme.[9]

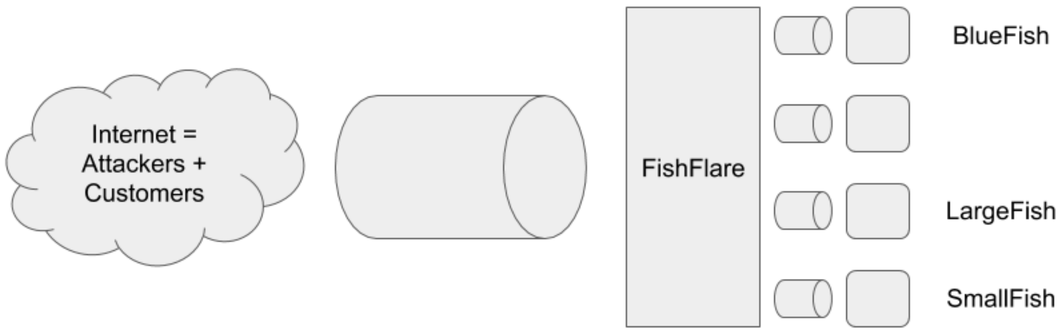
Given the constraints, what is the size of DDoS that this attacker can launch against `largefish.com`. Please provide the *total* traffic-volume seen by the victim `largefish.com` in units of Tbps (terabit/second), and provide a 2-3 sentence summary of how you achieved this number.

Solution:

The attacker can work around the rate-limit imposed by the `memcached` servers by specifying 10 victims instead of 1. However, once again, the attacker is limited by his own bandwidth. The expected traffic seen by the `largefish.com` is 3Tbps.

[9] This is a common load-balancing scheme used widely. In this example, the DNS lookup for `largefish.com` can return a randomly picked IP-address from the 10 choices. A particular client has a 10% chance of ending up with a given server, but it doesn't matter because all the servers host exactly same content. This allows `largefish.com` to scale linearly by just putting more servers.

(d) Let us now look at some of the mitigations of these DDoS attacks. To protect against a potential DDoS of the magnitude we analyzed above, website owners have to acquire significant extra bandwidth (by orders of magnitude), which is just not practical. However, the risk of a *specific* website being under attack at a given time is relative small. This is a perfect fit for an insurance scheme, *i.e.* risk distribution across a number of participants.



In order to provide protection against such large scale attacks, a new company `FishFlare` provides a DoS mitigation service to its customers. This is how it works: a customer, `BlueFish`, points the DNS entry for `bluefish.com` to `FishFlare`'s servers, and tells `FishFlare` about the real IP address that host `bluefish.com`'s content. This way, `FishFlare` sits between the customers of `BlueFish` and actual servers hosting content for `bluefish.com` (see the figure above).[10]

The hope is once `FishFlare` acquires enough customers, it can also acquire large enough bandwidth to withstand very large DDoS attacks with the cost amortized over a large number of customers. Since all the traffic passes through `FishFlare`, it can *scrub*-away the bogus traffic and only send cleaned-up traffic to actual `BlueFish` infrastructure.

How can `FishFlare` cleanup the `memcached`-based attack on the website of one of its customers? In no more than 2 sentences, describe an efficient strategy that works at line rate and is effective (*i.e.* very few, if any, false positives or false negatives).

Solution: Block UDP traffic with source port 11211; only send HTTP traffic to the customer website. A stateless packet filter should work well.

(e) Imagine that the idea of a DoS mitigation scheme became quite popular, and

[10]Note that this still leaves `BlueFish` vulnerable, since anyone who can figure out the real IP address can launch DDoS directly against `BlueFish` without going through `FishFlare`. This DNS based technique is merely one solution that fits small-scale businesses. The actual techniques used by large corporations are slightly more involved and require a bit nuanced knowledge of how the Internet routing works (specifically BGP), and context (whether a victim can seek help from its ISP in some filtering). You may learn about this more in a networking class; we ignore the details here to get the high-level idea across.

FishFlare is now quite successful. Now they have expanded their business in protection against not just network-level DoS attacks (such as the memcached attack), but also application-level attacks. More specifically, FishFlare acts as an application-level firewall and filters out malicious HTTP requests that attempt to exploit potential vulnerabilities in the web-applications of BlueFish.

FishFlare employs the most advanced analytics schemes to predict whether a particular HTTP request is a malicious request by an attacker, or a benign request by a real human. They are updating their analytics scheme and are deciding between the following:

1. Scheme 1 has a false positive rate of 5% and a false negative rate of 1%
2. Scheme 2 has a false positive rate of 1% and a false negative rate of 5%
3. Scheme 3:
   Run both given schemes in series, one after the other (run requests through scheme 1, and if they are marked benign pass them to scheme 2). The two schemes are not fully independent detectors: the false negative rate decreases to 0.1%, while the false positive rate increases to 5.5%

FishFlare performs a cost analysis and determines that each time an automated HTTP request by an attacker passes through undetected it costs \$100. Each time a benign user request is incorrectly flagged as an attack it costs \$3. The new schemes will not cost anything when a request is correctly categorized as malicious or benign. FishFlare estimates that 2 in every $10,000$ requests are malicious requests. Which detection scheme should FishFlare use? Which factor ends up mattering the most in this scenario, false positive or false negative rates? Show your calculations (Hint: which scheme would we *expect* to be more cost effective?) Please choose only from the 3 options given (do not, for instance, put "no option" or "option 1 in parallel with option 2" etc).

Solution:

1. Scheme 1:
   2 Malicious Requests $*0.01$ False Negative rate $*\$100 = \$2$
   $9,998$ Benign Requests $*0.05$ False Positive rate $*\$3 = \$1,499.70$
   Total Expected Cost for 10,000 requests $= \$1,501.70$
2. Scheme 2:
   2 Malicious Requests $*0.05$ False Negative rate $*\$100 = \$10$
   $9,998$ Benign Requests $*0.01$ False Positive rate $*\$3 = \$299.94$
   Total Expected Cost for 10,000 requests $= \$309.94$
3. Scheme 3:
   2 Malicious Requests $*0.001$ False Negative rate $*\$100 = \$0.20$
   $9,998$ Benign Requests $*0.055$ False Positive rate $*\$3 = \$1,649.67$
   Total Expected Cost for 10,000 requests $= \$1,649.87$

Note that this is the worst solution, despite the fact that chaining the two schemes decreased the false negative rate significantly and only slightly increased the false positive rate. The point here is that throwing a bunch of not-very-good schemes together does not create a good scheme, it can create an even worse one.

Thus Scheme 2 is far less costly, and the false positive rate matters much more. Note that both of these options are actually extremely bad because of the base rate fallacy. A large company can expect way more than 10,000 requests per day, so while a 1% error rate sounds nice, it will actually prove to be costly. In fact, we can expect to spend $200 per 10,000 requests if we don't use any NIDS at all!