CS628A: Final Examination Solutions

Computer Systems Security (100 points)

April 30, 2019 (3 Hours)

Student Name _____

Moodle email _____

Roll Number _____

Instructions: Please be brief and to the point in your answers. Don't write a lot of vaguely relevant stuff in the hope that some of it will be close to the right answer; we will take points off for the parts that are not correct. You will have to write your answers in the space provided and the space provided is more than enough to write the correct answer.

Do not open the exam booklet until the instructor announces that the exam has started. You **must put down your pens/pencils** when the instructor says that time is up. **If I find you writing after time is up, or before the exam has started, you will get an automatic 10-point deduction.**

Honor Pledge: Copy the academic integrity pledge below into the space provided and sign your name below it. You **can write** the honor pledge before the exam starts.

I pledge my honor that I have faithfully followed the institute's academic integrity policy during this examination.

Good luck!

_____

_____

_____

_____

_____

DO NOT TURN OVER UNTIL THE EXAM STARTS

Q1. GitHub DDoS v2.0 (10)

The largest confirmed DDoS attack thus far targeted GitHub in February of 2018. This attack generated 1.3 Gbps of traffic toward GitHub servers. It used `memcached` as an amplification vector, specifically the UDP-based protocol supported by `memcached` -- a key-value cache store.

Think of memcached as a key-value store, except that it is entirely stored in RAM and is typically used as a cache for database queries. So instead of directly sending queries to a database, we first check if the query's results are stored in `memcached`. If they are, we return the cached value. If not, we perform the query on the database server and store its result in `memcached` for future use.

a. What does "amplification vector" mean in the context of a DDoS attack? (2)

   It is a means by which the attacker is able to convert a small amount of attacker generated traffic into a larger amount of traffic directed at the victim.

b. `memcached` supports both TCP and UDP. Why did the attackers use UDP? (2)

   The query to `memcached` as well as the response are both single UDP packets. This allows them to spoof source IPs and direct the response to the victim. This would not be possible with TCP where only the SYN+ACK would be received by the victim.

c. Why would `memcached` be an amplification vector? (2)

   It is a key-value store. Keys are small in size while values are large, so it provides a lot of amplification.

d. Explain in one or two lines how the attack works. (3)

   - The attacker finds open `memcached` servers accessible over the internet. Apparently, there are tons of these.
   - They place some key/value pairs with small keys and large values on the server.
   - Then they query the value for each key with a spoof source IP.
   - The victim gets a large response back to them.

   If all 3 points above are mentioned, you get 3 points. Otherwise 2 points.

e. Did the attackers need to spoof their source IP for this attack? Why or why not? If they did spoof their source IP, what would they have set the spoofed value(s) to be? (1)

Yes, otherwise the attack wouldn't work. They set the spoof value to be Github's IP.

Q2. DNS Security (10)

In class, we learned about DNSSEC which uses public key cryptography to secure DNS. It turns out there is another proposal called DNS over TLS (sometimes abbreviated as DoT). DoT is exactly what the name says; in order to resolve a domain name into an IP address using DoT, the client:

1. Establishes a TCP (not UDP) connection to the DNS server.
2. Client and server setup the TLS connection (client hello, server hello, key exchange and so on). As usual the server provides a public key certificate to prove its authenticity. The client is expected to have trusted certificate authorities' public keys "hard coded" in its software.
3. Now the client sends the standard DNS query over the TLS session.
4. Server responds as in DNS over the same TLS session.

Note that only the connection from the client to a recursive nameserver uses DoT.

a. What is a security guarantee offered by DoT that is **not** offered by DNSSEC? (1)

Privacy. DNSSEC doesn't encrypt anything, it only signs records. With DoT, the connection to the nameserver is encrypted so a passive network attacker can't find out which sites are being accessed by a particular user by monitoring DNS.

b. What are **two** security guarantees offered by DNSSEC that are **not** offered by DoT? (2)

- Proof of authenticity – that the IP for a particular domain is in fact the true IP.
- Authenticated denial of existence – if the nameserver says that a particular domain/sub-domain doesn't exist then it really does not exist.

c. Does DNSSEC alone prevent the Kaminsky cache poisoning attack? Why or why not? (1)

Yes, DNSSEC prevents the attacker from announcing bogus IPs for a DNSSEC protected domain.

d. Does DoT alone prevent the Kaminsky cache poisoning attack? Why or why not? (2)

No, DoT doesn't provide this guarantee. It doesn't provide any authenticity of domain names to IP address mappings.

e. Does DoT alone prevent the DNS rebinding attack? Why or why not? (2)

No, in DNS rebinding it is the attacker's nameserver that is announcing a bad IP address for its own domain. There is no way to prevent this using either DNSSEC or DoT.

f. We learned in class that DNS servers are often used as an amplification vector in DDoS attacks. Does DNSSEC prevent this? How about DoT? (2)

No, DNSSEC still uses UDP if the response packet is small enough. So it can't prevent source IP spoofing/amplification attacks.

DoT requires a TCP connection, and here a spoofed source IP can't be used to establish a TCP connection as the handshake won't be complete. So it is harder to use DoT for amplification of a DDoS attack.

Q3. Aladeen-in-the-middle (10)

Admiral-General Aladeen, the dictator of the fictional republic of Wadiya, would like to monitor all his citizens' Facebook and Gmail messages. He decides to do this by MITM'ing all traffic between his citizens' computers and Facebook/Google. He asks his council of technologists to come up with ways of doing this. They've come up with the following proposals. Your job is to explain how one would get around each of the proposals listed below.

Note that all ISPs in Wadiya detect and block VPNs and Tor so these two options are ruled out. One thing that might help you is that not everyone is being MITM'd all the time. The military of Wadiya has limited computing/intercepting power, so they attack a randomly chosen subset of Internet users in any given hour. After the end of each hour, a new randomly chosen subset is attacked.

a. Tamir suggests that the military should force the default DNS servers used by all ISPs in Wadiya to be servers under military control. These DNS servers will give out the IP addresses of a fake Google/Facebook server whenever a DNS query is sent to them, and the fake "Google" server will be used to eavesdrop on users. Give two ways through which a user (not Facebook/Google) can defeat this proposal. (2)

   - Users can manually set their DNS to Google DNS, which will not be affected by the MITM.
   - Users must only use HTTPS. This way certificate validation will fail on the fake sites.

b. Omar says that fake DNS servers are trivial to defeat and instead suggests BGP route hijacking. He wants to announce bogus routes to Google/Facebook IPs for the users being currently attacked. These IPs serve fake Google/Facebook websites over HTTP. What is one way in which Google/Facebook can mitigate this attack? How could a user protect themselves against this attack? (2)

   - Google can use the HTTP Strict Transport Security (HSTS) header to ensure that browsers will not connect to HTTP versions of the website.
   - Users can make sure to always use HTTPS (for example by using the HTTPS Everywhere addon).

c. Tamir comes back to Aladeen with the information that he has managed to bribe someone in the Tunisian Root Certificate Authority to issue fake certificates for *.google.com and *.facebook.com. He wants to MITM all Google and Facebook traffic using these fake certificates now. How would a user protect themselves against this attack? How would Facebook/Google mitigate this attack? (2)

   - Users would manually remove all untrustworthy CAs from their browser.
   - Google/Facebook could use public key pinning (HPKP) to advertise correct public keys for their certificates. The keys are trusted on first use.
   - Google/Facebook could use Certificate transparency (CT) which requires all certificates issued by a CA to be part of a public log.

- Further, with both certificate transparency and public key pinning, if browsers see an invalid certificate, they will report this information to the web server (i.e., Google/Facebook).

d. After some chats with his tech team, Omar says that none of these MITM options are ever going to work. Instead, Aladeen must ensure that all laptops sold in Wadiya must come pre-installed with a browser add-on that ostensibly provides push notifications of important government announcements to users. Why would he be suggesting this? (2)

Add-ons (if given the permissions by the user) can access all browser data across all websites. They are perfect technique for eavesdropping on all of a user's actions.

e. A separate thing that Aladeen wants to do is log all webpages visited by each of his citizens. Can Aladeen's underlings collect this information even if a user is careful to only use sites served over HTTPS? What information can and cannot be logged in this scenario? (2)

Things that can be logged by a passive network attacker:

- IP address being visited (in most cases, this means the domain can also be inferred in conjunction with DNS traffic observation)
- Port number being connected to
- Number of packets and the size of each encrypted packet

Things that are not visible to a passive network attacker:

- The path within the website that is being visited. (E.g., Aladeen may see that google.com is being visited, but not that www.google.com/search?q=how%20to%20kill%20a%20dictator is being visited.)
- The HTTP request headers, cookies, POST parameters
- HTTP response headers, cookies, response data

Note that with modern extensions like encrypted SNI (server name indication) used in conjunction with DNS over TLS, it may not be possible to view the domain name. If you write this as your answer with the correct explanation, you will get full credit. But since we didn't cover these technologies in class, you will get full credit even if you write the answer given above.

Q4. XSS and CSRF (15)

Consider the following proposal to mitigate XSS attacks. We introduce a new HTTP header "client-side-escape: 1" that will be sent in the HTTP response from the webserver to the browser. If this header is included, then the browser will escape all URLs, get and post parameters before sending them from the client to the server. Your job is to critique the above proposal.

a. Suppose this proposal was implemented and supported by Google Chrome, Firefox and Internet Explorer. Would users of these browsers be prevented from being the victim of XSS attacks? Would they be prevented from launching XSS attacks? Why or why not? (3)

   i. No, even if the proposal worked users would be vulnerable to stored XSS attacks launched by users of other browsers.
   ii. No, XSS can be launched by using more than just URLs/get/post parameters. The AirBnb attack mentioned in class launched XSS by submitting a malicious Origin header. One could also have malicious User-agent headers. (People could use browser addons to modify these headers.)

b. Does this proposal prevent XSS attacks? Why or why not? (2)

   It doesn't prevent anything because attackers can use modified/malicious browsers to launch attacks. Further, even the set of things they are escaping is not enough.

   The lesson here is that it is very difficult to prevent XSS on the client side.

Chrome, IE and Firefox support an attribute "SameSite" attribute for the "Set-Cookie" header. If this attribute is set to "Strict", then the cookie will not be sent along with cross-site requests.

d. Explain why this prevents CSRF attacks. (2)

   CSRF relies on the user's cookie being sent along with an attacker generated request (usually from a different site/domain). So, if cookies are not sent with cross-site requests, CSRF will not occur.

e. What is a scenario where SameSite will not be enough to prevent request forgery? (2)

   Note we are asking about request forgery not cross-site request forgery.

   Intra-site request forgery is still a problem and occurs with websites that host user-generated content. For instance, one reddit user could try to use intra-site request forgery to attack other reddit users. In this case the SameSite protection won't prevent the attack.

f. Is there a usability problem with setting the SameSite header? If so, explain why. (2)
Hint: what would happen if Google Docs used SameSite cookies and I emailed you a link to a document.

Yes, cookies are no longer sent with cross-site requests, so link sharing becomes hard. Suppose I received a link to a google document shared with my google id on my yahoo email account. When I click the link from mail.yahoo.com, the SameSite header prevents the google cookie from being sent along with it. So, I will need to login again to my google account.

g. What are two other ways of preventing CSRF attacks that we learned in about in class? For each method discuss whether they will be effective if an attacker has also found an XSS vulnerability in order to inject JavaScript to the victim website. (4)

- CSRF Tokens: include a hidden field with a secret token that is uniquely generated from the session ID. If a request is not submitted with the correct token, it is rejected on the server side. This is not effective when an XSS vulnerability is present because the attacker can just JS can to read the token and submit it.
- Referrer validation: on the server side check the referrer (or origin header) and make sure that the request comes from a valid domain. This is also vulnerable if the attacker is able use XSS to inject JS, because the malicious JS can send requests with the correct referrer.

Q5. Session Management (10)

a. We learned about three ways of tracking session IDs in class. What are these? (3)

Method 1: Using cookies to store the session ids.

Method 2: Embedding the session id in all URLs.

Method 3: Using a hidden form field in all pages to store the session ids.

b. We also learned that it is often necessary to use a combination of these three methods. For this sub-part of the question, assume that only one of these methods is used. Explain briefly which of methods are vulnerable to CSRF, which are not, and why. (3)

Method 1: Cookies are vulnerable to CSRF because they get auto-submitted with each request to the same domain.

Method 2: Using the URL to store session IDs makes them secure against CSRF because the attacker will not have access to the correct URL for a particular user. However, they may make session hijacking easier.

Method 3: Storing the session ID in a hidden form field makes CSRF very difficult. In fact, this is a lot like the CSRF token idea. The only downside is that long-lived sessions are not possible with this method.

c. Separate from the issue of how session tokens are stored is the problem of how session tokens are generated. Compare the security and implementation practicality of the following methods of generating session tokens. (4)

i. SHA256(user_name || server_secret || login_date)
ii. SHA256(user_name || password)
iii. SHA256(user_name || nonce)

(ii) is terrible because it is vulnerable to brute-force attacks against common passwords.

(iii) is kinda redundant because the nonce has to be generated and stored on the server side for each user. So why not just use the nonce as the session id and save ourselves some computation.

(i) obviates the need for storing any session id on the server side, but will force the user to login at least once every day.

3 points for getting security properties correct + 1 point for realizing that (i) doesn't need stored per-user session state but (iii) does.

Q6. TLS/HTTPS (10)

a. What does certificate transparency achieve? Briefly explain how it works and what attacks it prevents? (3)

There are two parts to certificate transparency. (2 points)

- Every certificate signed by a CA is placed on a public log.
- On the client side, when certificate transparency is enabled, the browser checks whether the certificate appears in this log. If it does, then the certificate is accepted. If it does not, this bad certificate is reported to the web server.

It prevents the rogue CA attack – untrustworthy CAs can't sign certificates for, say, *.google.com or *.yahoo.com. Because if they did, they would have to place these certificates on the public log and everybody would know they are untrustworthy. Or if they didn't place them on the log, when the users accessed these websites, the web browser would report these bad certificates back to the webserver. (1 point)

b. The HTTP response header "Expect-CT" is sent from a webserver to the browser to indicate that the website uses certificate transparency. Why is this header required? What would happen if this header is not sent? (2)

The header is required so that the browser knows that this domain uses certificate transparency. If it were not sent, the browser wouldn't not know that it is supposed to check the certificate logs.

c. What does the HTTP header "Content-Security-Policy:upgrade-insecure-requests" do? Who sends this header to whom and what is an example of an attack that is prevented by the inclusion of this header? (3)

- It converts all http resources (images, javascript etc.) to https links.
- It is sent from the webserver to the client (browser).
- It prevents a network attacker from attack a HTTPS website that uses some HTTP resources (images/JS).

d. A related HTTP header is "Content-Security-Policy:block-all-mixed-content". What does this header do and why would one use it? (2)

It prevents any HTTP resources (images, JS, etc.) from being accessed by a HTTPS site. While upgrade-insecure-requests converts HTTP references to HTTPS, this one just blocks them.

Q7. Web Client Security (10)

a. The window.postMessage function takes two arguments – the message to be posted and the target origin. What checks should be performed by the receiver of the message? What would happen if these checks were not performed? Why is the target origin required? (3)

- The receiver should check that the origin of the message is trusted.
- The sender should send the message to an expected origin.

On the sender side if the message is sent to wrong target origin, there is a possible of confidential information being leaked to the attacker. For example, if Gmail used postMessage to communicate between the main window and a chat window, then if no target origin is specified, there may be a possibility that chat messages are sent to an attacker.

One the receiver side, if the message origin is not checked, the receiver frame may execute some sensitive operation based on attacker input. Returning to the Gmail example, if the chat window didn't check message origin, it may end up showing a message from attacker.com in the chat window.

b. What are HTML ETags? What were they designed for? How can they be (ab)used to track visitors to a site? (3)

ETags are entity tags and they were introduced to help implement more efficient caching in the browser. Think of the ETag as a hash of the content. The server would return an ETag value along with each response. The browser would store the content along with this ETag. If the same site is requested again, the browser send the stored ETag value to the server and the server can respond with new data if the content has changed (by checking the ETag), or just send a response saying that the content has not changed. (1 point)

The ETag ends up acting just like a cookie because it is a unique identifier sent along with requests from the browser to the server. By sending a unique ETag for each user and observing the values that are returned, the ETag can be used to track visitors to a website. (2 points)

c. What is the difference between third-party and first-party cookies? Since all cookies can be used for tracking, why would a privacy conscious person block only third-party cookies? (2)

First-party cookies are cookie set by the site visible in the address bar. Third-party cookies are cookies for sites embedded in frames or other resources inside the first-party site. (1 point)

Third-party cookies are bad because they referrer information about the first-party site being visited, so they are especially helpful for tracking users across different websites. This is why privacy-conscious people block third-party cookies. (1 point)

d. According to the Mozilla Developer Network, “two URLs are considered to be the same origin if they have the same protocol, host and port.” Why is the protocol important in this definition? Give an example of an attack that could be carried out if the protocol were not considered in the same origin check. (2)

If the protocol were not considered, a network attacker could serve a bogus HTTP website like http://www.google.com and have this be considered the same origin as https://www.google.com. This would allow the network attacker to steal all of your google account information by serving malicious JavaScript over the http website.

Q8. Developing Secure Applications (10)

When Google Chrome was first introduced about a decade ago, an important innovation in the browser was its use of a multi-process architecture. It had three types of processes:

- The browser process manages tabs, windows, network, user input and display. Note the browser only reads the network input, it does not parse or interpret it.
- The renderer process contained all the logic for parsing and rendering HTML and CSS, executing Javascript, etc. Note that the renderer figures out what to draw on the screen, the actual drawing is done by the browser.
- The plug-in processes executed plugins like Flash, PDF and so on. There was one process for each type of plugin.

a. What is purpose of having each of these types of processes? In other words, write very briefly about what security objective is ensured by each type of process. (3)

   - The browser process handles I/O interaction.
   - The renderer does the parsing of HTTP and CSS, so if there are buffer overflows in the renderer or during JavaScript execution, this does not affect the actual browser process where the content is displayed, or user input is read. This helps mitigate the impact of any vulnerabilities in the handling of untrusted data coming in from the network.
   - Plug-ins are notorious source of vulnerabilities and this architecture ensures that even if Flash/Adobe PDF have a buffer-overflow vulnerability, they are prevented from accessing user data as the data is in a separate process.

b. There is only one browser process but there are many renderer processes. Why are there many renderers and what is one plausible way of dividing work among the renderers? (2)

   We could have on renderer for each origin. This way buffer overflow attacks from one origin cannot steal the data of another origin.

c. The renderer processes are run inside sandboxes. What kind of sandbox would this be? Give an example of a vulnerability that sandboxing aims to mitigate. (2)

   The sandbox prevents the renderer from making most system calls. This is again to mitigate the impact of buffer overflow vulnerabilities in the renderer.

d. What would be an argument against this multi-process design? In other words, why didn't other browsers at the time also have the same type of design? (1)

   Having a lot of processes consumes more memory and involves many more context switches so it slows down both the browser itself as well as other applications executing on the machine.

e. Why do we need to put each plugin in a separate process? What is an example of a vulnerability that would occur if, say Flash and PDF viewers shared the same process? (2)

Suppose I am playing a flash game in one tab and viewing a PDF document containing student marks in another tab, and there is a buffer overflow vulnerability in the flash player. In this scenario, an attacker who manages to exploit flash will also be able to read all student marks. Putting separate plugins in separate processes prevents this.

Q9. Cryptographic Primitives (10)

a. What is bad about using AES in Electronic Code Book (ECB) mode? (1)

It leads to deterministic encryption. Same plaintext has the same ciphertext.

b. Recall that the Cipher Block Chaining (CBC) mode requires an initialization vector (IV). Note the IV is chosen at the time of encryption and then has to be stored/transmitted in plaintext for use during decryption. Therefore, does it make sense to set the IV to be an HMAC of the message so that it doesn't need to be stored/transmitted? Explain your answer. (2)

No, it does not make sense because the receiver would be unable to compute the HMAC. Further any such scheme will lead to deterministic encryption, which is also bad.

c. Why is optional asymmetric encryption padding (OAEP) needed for RSA? (1)

To prevent brute force attacks on small messages. OAEP introduces randomization in the input message before exponentiation.

d. What is the Diffie-Hellman Key Exchange (DHE) used for? (1)

To derive a session key from long-term public keys.

e. Dropbox – the file sharing service – uses de-duplication: if two users store the same file on Dropbox, their servers will store only one copy of the file. How would Dropbox implement de-duplication efficiently? Briefly describe the algorithm Dropbox would use. (2)

- When the file is to be uploaded for the first time, Dropbox client software computes the hash of the file using a cryptographic hashing algorithm (e.g., SHA256).
- It sends this hash to the server to check if a file with the same hash is already present on the server.
- If the hash is present, there's no need to upload. Just store a pointer to the existing file.
- If the hash is not present, upload the file afresh.

We are using the collision-resistance property of hash functions.

f. What does their use of de-duplication tell you about Dropbox's design in comparison to the Key-Value store you built in your assignment? In particular, are they encrypting data using a key derived from your password? (1)

They are **not** encrypting each user's data using a key derived from the password. At best, everyone's data is encrypted with the same server-side key. At worst, there's no encryption at all.

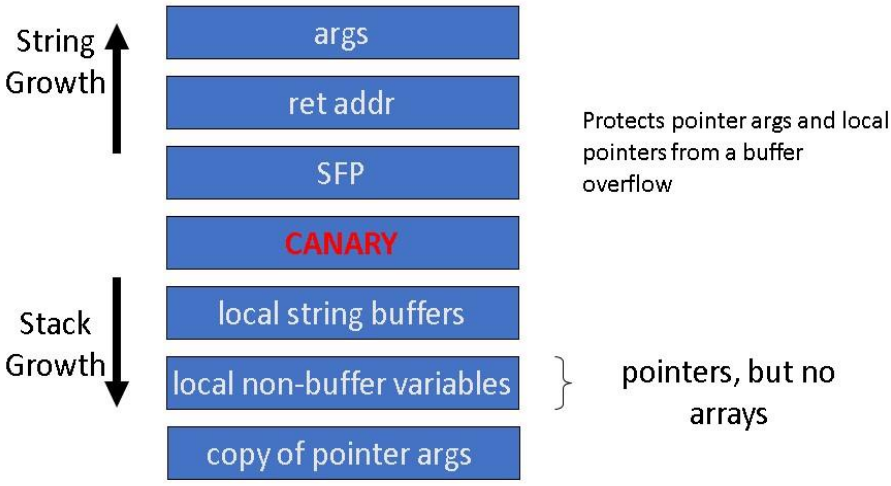
g. Does the Diffie-Helman Key Exchange protect past session keys even if the underlying long-term private key is compromised? What about session keys derived after the private key is compromised but before the compromise is discovered? (2)

It protects past session keys, but it cannot protect session keys derived after the compromise because there is no way to distinguish the honest server from an MITM attack who has access to the private key.

Q10. Control Hijacking (10)

a. Explain the ProPolice stack layout. Justify why it reorders variables and what attacks are prevented due to the ProPolice layout. (3)

ProPolice is GCC modification to defend against Buffer Overflows. It rearranges the stack layout such that a overflowed buffer can't overwrite return address. It also places a random ca`nary between local variables and return address to prevent overwriting return address.



It reorders stack layout and changes the growth of string buffers such that an overflowed buffer can't overwrite return address or even pointer arguments.

Buffer overflows, Ret-to-libc, Return oriented programming all are prevented by ProPolice.

b. What kind of NOP slide would be useful in return-oriented programming attacks? Explain your answer. (2)

There is no utilization of NOP slide in return-oriented programming attacks as we don't need to execute anything on stack.

c. One type of attack that we didn't learn about in class are use-after-frees (UAF). Based on what you have now learned about memory safety vulnerabilities, give a brief description of what UAF vulnerabilities and why they might be exploitable. (4)

During execution program might free dynamically allocated memory. If during later part of execution program is referring to the already freed data, a malicious attacker can inject its own malicious payload onto that freed location (probably using heap spraying). In this way attacker can make victim program dereference a pointer that contains injected malicious content by attacker – this could lead to control hijacking.

We gave two points to those who said that this could lead to an information leakage vulnerable, although that is technically incorrect because there is no "use" after the free here (the "use" must be done by the original program.)

d. What is the purpose of heap spraying? (1)

Even if stack is non-executable chances are high that heap will still be executable. Attacker can use heap spraying in conjunction with buffer overflow to make victim program jump to arbitrary heap location where attacker has 'sprayed' malicious code, which would lead to arbitrary code execution even if systems has non executable stack.

(END OF EXAM)