Popa & Weaver
Spring 2019

CS 161
Computer Security

Midterm 2

PRINT your name: Liao, Ran
(last) (first)

I am aware of the Berkeley Campus Code of Student Conduct and acknowledge that academic misconduct will be reported to the Center for Student Conduct and may further result in partial or complete loss of credit.

SIGN your name: ______________________

PRINT your SID: 3034504227

Name of the person sitting to your left: ______________________

Name of the person sitting to your right: BARTELL FEI LIN

You may consult **two** double-sided, handwritten sheet of paper of notes. You may not consult other notes or textbooks. Calculators, computers, and other electronic devices are not permitted.

Bubble every item completely. Avoid using checkmarks or Xs.
If you want to unselect an option, erase it completely and clearly.

For questions with **circular bubbles**, you may select only one choice.

○ Unselected option (completely unfilled)

● Only one selected option (completely filled)

For questions with **square checkboxes**, you may select any number of choices (including none or all).

■ You can select

■ multiple squares (completely filled).

If you think a question is ambiguous, please come up to the front of the exam room to the TAs. If we agree that the question is ambiguous we will add clarifications to the document projected in the exam rooms.

You have 110 minutes. There are 9 questions of varying credit (116 points total).

Do not turn this page until your instructor tells you to do so.

Problem 1 ***True or False*** **(20 points)**

You don't need to provide an explanation to any part of this question.

(a) True or False: TCP sequence numbers prevent replay attacks within the same TLS session. (Assume less than 2^{31} bytes of data has been transferred.)

● True ○ False

(b) True or False: A single message in TLS can be split into many TCP packets, and remains secure.

● True ○ False

(c) True or False: `www.example.com` can set a cookie with the flag `HttpOnly`. Then, this cookie can only be accessed through HTTP, but cannot be accessed through HTTPS.

○ True ● False

(d) True or False: `www.example.com/c` can set a cookie with the flag `path = /a` in a user's browser. Then, when the user visits the site `www.example.com/a/b`, the user's browser will send this cookie.

○ True ● False

(e) True or False: Say that `example.com` uses both DNSSEC and HTTPS. If the DNSSEC KSK of `example.com` is compromised, data received from `https://example.com` is not assured to be confidential.

○ True ● False

(f) True or False: Say that `example.com` uses both DNSSEC and HTTPS. If the TLS private key of `example.com` is leaked, data received from `https://example.com` is not assured to be confidential.

● True ○ False

(g) True or False: Say that a user sends a DNS query asking for `nx.example.com`, but this domain does not exist. One advantage of NSEC3 over NSEC is that NSEC3 hides the domain name that does not exist, *i.e.*, `nx.example.com`.

○ True ● False

(h) TRUE or FALSE: A banking website requires the user to attach their password as a form field in every HTTPS request to the website. If the password is incorrect, the bank ignores the request. Assume that the bank accepts only HTTPS connections. If the user's password is sufficiently high-entropy, this method prevents CSRF.

○ TRUE ● FALSE

(i) TRUE or FALSE: Randomizing the source port used by DNS queries can help prevent on-path attackers from spoofing DNS replies.

○ TRUE ● FALSE

(j) TRUE or FALSE: An on-path attacker who successfully spoofs a DHCP reply can become a man-in-the-middle for all victim traffic to the Internet.

● TRUE ○ FALSE

Problem 2 *Wildcard DNSSEC* (11 points)

In this question, we discuss a variant of DNSSEC that supports wildcards.

We define a *wildcard* domain as a domain that matches many subdomains.

- For example, the wildcard domain `*.google.com` matches *all* domains under `google.com`, including `mail.google.com` and `drive.google.com`. Here, the star * indicates a wildcard.

We define a *non-wildcard* domain as a domain with no wildcard, such as `maps.google.com`.

- **Importantly,** non-wildcard domain records take *priority* over the wildcard domain records.
- For example, if there are two records:

*.google.com	5.6.7.8
mail.google.com	1.2.3.4

then the DNS server should respond `1.2.3.4` as the IP address of `mail.google.com`.

(a) In classical DNSSEC, if a user asks for the IP address of `mail.google.com`, and there is a *non*-wildcard record, the DNS server will return:

1. the IP address of `mail.google.com`.
2. the signature of the record containing the IP address.

◇ **Question**: What is the type of the record (*e.g.*, A, NS) that contains the information above?

1. the IP address of `mail.google.com`...
   - ... is in a record of the type A (write the record type).
2. the signature of the record above...
   - ... is in a record of the type RRSIG (write the record type).

RR SiG

(b) We now modify the DNSSEC protocol to support wildcards, as follows:

- Consider a user who asks the IP address of the domain `abc.google.com`. There is only a wildcard record that matches `abc.google.com`, as follows:

  `*.google.com 5.6.7.8,`

- The server will return a record that consists of:

  `(abc.google.com, 5.6.7.8),`

  and *a signature* over this record.

However, this design is not good because it involves *online signing*; that is, the server cannot precompute the signature.

◇ **Question:** List one drawback of having online signing in DNSSEC. (write less than 10 words)

The server may be vulnerable to DOS attack.

(c) To remove the online signing, we can have the DNS server instead do the following:

- Return the wildcard record, which consists of:

```
(*.google.com, 5.6.7.8),
```

- Return a signature over the record above.

A client who asks `abc.google.com` will receive this response. The client will believe that:

- No non-wildcard record matches the query.
- Only the wildcard captures this domain.

⋄ **Question:** Is this design secure? If yes, explain why. If not, explain how it could be made secure without requiring online signing (max 15 words).

○ Yes, it's secure ● No, it's insecure

Return the previous existing subdomain and the next existing subdomain and the a signature over them

Problem 3 *Online Banking* **(16 points)**

In an online banking system, each customer has a unique username and a secret numerical PIN. To access the banking website, a customer logs in to this web system using their username and PIN.

Suppose the login script uses the following PHP code:

```
$user = escape_sql($_GET['username']);
$pin = $_GET['PIN'];
$query = "SELECT * FROM Users WHERE user = '$user' AND pin = '$pin'";
$results = $db->executeQuery(query);
if ($results->numRows != 1) {/* login fails */}
else {/* login succeeds as $user */}
```

Here, the `escape_sql` function escapes all quotes, dashes, and semicolons. You may assume that SQL injection cannot be performed from any input that has been sanitized with `escape_sql`.

(a) Mallory obtains the source code of the login script and notices that it is vulnerable to SQL injection.

◇ **Question:** Describe *what input* Mallory should use (*e.g.*, `$_GET['username']`, `$_GET['PIN']`) to exploit this vulnerability in order to drop the table `Users`.

`$_GET['username']` = X

`$_GET['PIN']` = '; drop table Users; --

(b) Mallory knows a rich user whose username is "`alice`".

◇ **Question:** Explain *what input* Mallory should use for SQL injection in order to log in to the online banking system as Alice, *without* knowing Alice's PIN.

`$_GET['username']` = Alice

`$_GET['PIN']` = '; SELECT * FROM User WHERE user='Alice'; --

(c) The bank decides to fix the SQL injection bug by constraining the PIN that the user enters to be an integer, as follows.

- Change the HTML of the login page:
  - From:

```
<form action="/login.php" method="POST">
<p>Username: <input type="text" name="username" /></p>
<p>PIN: <input type="text" name="PIN" /></p>
<p><input type="submit" value="Login" /></p>
</form>
```

– To:

```
<form action="/login.php" method="POST">
<p>Username: <input type="text" name="username" /></p>
<p>PIN: <input type="number" name="PIN" /></p>
<p><input type="submit" value="Login" /></p>
</form>
```

- The <input> element with "type=number" will be treated differently by the web browser. The web browser will prevent the user from entering non-numerical data into this input field.
- *No* change to the PHP script.

◇ TRUE or FALSE: Does this fix prevent the SQL injection in parts (a) and (b)?

○ TRUE ● FALSE

You don't need to provide an explanation.

(d) The bank later did a major re-design of their website. Unfortunately, the new version was vulnerable to a CSRF attack. Mallory notices that she can exploit it by having another user make a GET request like:

/transfer?amt=100&to=Mallory

and the user who makes this GET request will send $100 to Mallory.

For each of the choices below, mark if it defends against CSRF. Assume that aside from the /transfer? endpoint, no other part of the bank is vulnerable to any web attacks.

◇ **Select** 0 to 5 options.

□ Disable JavaScript from executing on the bank's website via content security policy (CSP).

□ Add a new request parameter "from=" to the /transfer? endpoint. If "from" does not match the name of the currently logged in user according to the session cookie, reject the request.

☒ When a user logs in, send them a new cookie called Token, which consists of 128 random digits (different for each user). When a user makes a request to transfer money, the bank uses JavaScript to retrieve the cookie and add it as a query parameter "token=" to the /transfer endpoint. The bank checks that the token query parameter matches what the cookie was originally set to for this user.

☒ When a user logs in, send them a new cookie called Token, the same one above. When the user makes a request to the /transfer? endpoint, the bank checks that cookie sent by the user matches what the cookie was originally set to for this user.

☒ Reject any request to the /transfer? endpoint where the Referer is not the bank's website.

Problem 4 ***The Subtle TLS*** **(10 points)**

This question talks about a ***modified*** RSA TLS protocol. Recall in RSA TLS, the client sends R_b to the server, and then the server replies with R_s to the client. The cipher and integrity keys are generated by putting R_b, R_s, and the premaster secret PS together into a PRNG, like $PRNG(R_b \parallel R_s \parallel PS)$.

For each part of the question, assume an attacker with the following capabilities:

- The attacker is a man-in-the-middle.
- The attacker also controls a website `evil.example.com`, with a valid HTTPS certificate. The user may connect to this site while browsing the Internet.

(a) Let us assume that in RSA TLS, we make the following change. We generate the pre-master secret by $PS = R_b \oplus R_s$, where $\oplus$ is bit-wise XOR.

TRUE or FALSE and Explain: This modified protocol preserves the integrity of RSA TLS.

○ TRUE ● FALSE

◇ **Explain** ***concisely***:

R_b and R_s are not encrypted, therefore the attacker can access them and recompute the PS

(b) Let us assume that in RSA TLS, we make the following change. Instead of providing "$R_b \parallel R_s \parallel PS$" as input to the PRNG, both parties provide "$R_b \oplus R_s \oplus PS$" as the only input to the PRNG. That is, the cipher and integrity keys will depend only on $PRNG(R_b \oplus R_s \oplus PS)$.

TRUE or FALSE and Explain: This preserves the security of TLS against replay attacks.

● TRUE ○ FALSE

◇ **Explain** ***concisely***:

As long as PS is secret The attacker cannot recompute the symmetric encryption key

(c) Let us assume that in RSA TLS, we make the following change. Rather than generating the PS randomly, the client begins with an initial value PS. For each TLS connection the client makes, it simply increments the PS as $PS \leftarrow PS + 1$.

TRUE or FALSE and Explain: This preserves the confidentiality of RSA TLS.

○ TRUE ● FALSE

◇ **Explain** ***concisely***:

If the client visit evil.example.com. The attacker can know the current PS. and thus know all PS in the following TLS connections

Problem 5 *WPA2 Security* **(13 points)**

Recall the WPA2 handshake protocol based on pre-shared keys, as follows:

- The pre-shared key (PSK) is computed from the passphrase (*i.e.*, the Wi-Fi password) and network SSID (*i.e.*, the name of the Wi-Fi network).
- Pairwise transient key (PTK) is determined from the ANonce, SNonce, and the pre-shared key (PSK) using a pseudorandom generator.
- The client derives the encryption and MIC keys from the PTK.

(a) TRUE or FALSE: The WPA2 protocol described above provides forward secrecy.

○ TRUE ● FALSE

You don't need to provide an explanation.

(b) Alice connected her computer to a Wi-Fi access point and accessed a few websites. An eavesdropper recorded the ANonce, SNonce, and all other messages in Alice's connection.

With the information collected above, the eavesdropper can brute-force the passphrase. We consider that at this moment, the eavesdropper guesses that P might be the correct passphrase.

◇ **Question:** How can the eavesdropper confirm whether or not P is the correct passphrase *without* connecting to the Wi-Fi access point? (answer concisely)

He can recompute the encryption and MIC keys according to the procedure described above. The he can try to use these keys to decrypt and verify messages

(c) A man-in-the-middle attacker knows the PSK of a Wi-Fi access point. Alice's laptop has been connected to this access point and is visiting `https://www.chase.com/`. Alice visited this website and logged in by providing her password.

◇ TRUE or FALSE: A man-in-the-middle attacker can see Alice's banking password8.

○ TRUE ● FALSE

◇ **Explain *concisely*:** TLS will provide confidentiality

(d) In WPA2 Enterprise the device authenticates to an authentication server over TLS to generate a unique PSK for the user before the WPA2 handshake. Yet, in the real world, users usually accept any certificate blindly since there is no notion of "name" (unlike for a web server) which the client can automatically validate.

◇ TRUE or FALSE: In this case, the protocol is secure against a man-in-the-middle attacker.

○ TRUE ● FALSE

◇ TRUE or FALSE: In this case, the protocol is secure against a passive eavesdropper.

● TRUE ○ FALSE

You don't need to provide an explanation.

Problem 6 *DNS* **(14 points)**

(a) Injecting spoofed packets as an off-path attacker in TCP is much harder than in UDP. Even though TCP has a higher security guarantee, DNS often does not use TCP because TCP has a much higher *latency*.

⋄ **Question:** Compared with UDP, what is the *chief* reason why using TCP for DNS has a higher latency? (answer within 10 words)

TCP needs 3-way handshake to build a valid connection

(b) Alice wants to access Berkeley's diversity advancement project DARE, dare.berkeley.edu. Her laptop connects to a wireless access point (AP).

Alice worries that a hacker attacks the DNS protocol when her laptop is looking for the IP address of dare.berkeley.edu. Assume that DNSSEC is not in use.

⋄ **Question:** Which of the following can attack the DNS protocol and have Alice's browser obtain an incorrect IP address for DARE? (Select 0 to 8 options.)

■ The laptop's operating system.

■ The laptop's network interface controller.

■ The wireless access point.

■ An on-path attacker on the local network.

■ The local DNS resolver of the network.

■ The root DNS servers.

■ berkeley.edu's DNS nameservers.

■ An on-path attacker between the local DNS resolver and the rest of the Internet.

(c) Now assume that berkeley.edu implements DNSSEC and Alice's recursive resolver (but not her client) validates DNSSEC.

⋄ **Question:** Which of the following can attack the DNS protocol and have Alice's browser obtain an incorrect IP address for DARE? (Select 0 to 8 options.)

■ The laptop's operating system.

■ The laptop's network interface controller.

■ The wireless access point.

■ An on-path attacker on the local network.

■ The local DNS resolver of the network.

■ The root DNS servers.

□ berkeley.edu's DNS nameservers. X

□ An on-path attacker between the local DNS resolver and the rest of the Internet. X

(d) An attacker wants to poison the local DNS resolver's cache using the Kaminsky attack. We assume that the resolver does not use source port randomization, so the attacker will likely succeed.

In the Kaminsky attack, the attacker asks the resolver for a *non-existing* subdomain of UC Berkeley, *e.g.*, `stanford.berkeley.edu`, instead of asking for an *existing* domain like `dare.berkeley.edu`.

◇ **Question:** What is the advantage of asking for a non-existent domain compared to asking for an existing domain? (answer within 10 words)

Make sure the resolver can not cache the correct IP address. So, the attacker can attack multiple times until he succeed.

Problem 7 ***Same-Origin Policy*** **(13 points)**

(a) TRUE or FALSE: Setting "secure" flag on a cookie protects it from a network attacker eavesdropping on an insecure HTTP connection.

● TRUE ○ FALSE

You don't need to provide an explanation.

(b) TRUE or FALSE: After a successful XSS attack, JavaScript can access all cookies set by the website it attacked.

○ TRUE ● FALSE

You don't need to provide an explanation.

(c) Which of these URIs have the same origin as "`http://same.origin.com:80/a.htm`" according to same origin policy? (choose 0 to 4 options)

□ `http://origin.com:80/a.htm` X ■ `http://same.origin.com:80`

■ `http://same.origin.com:80/a.htm/b` □ `ftp://same.origin.com:80` X

(d) If a page loads a JavaScript file from some other site, this JavaScript file takes the origin of...

Choose one option:

● The page that loaded it ○ The site that hosts the JavaScript file

(e) Same-origin policy is very useful in preventing many web attacks. Yet, it also inconveniences for web developers – different domains cannot talk to each other.

◇ **Question:** Provide a *specific* solution for the web developers to *conveniently* enable JavaScript in different domains' webpages to *conveniently* talk to each other. (answer less than 10 words)

Use cookies to communicate with each other

Problem 8 *HTTP TRACE method* **(7 points)**

Web servers can support another type of HTTP requests, TRACE, as follows.

A TRACE request is like a GET request or a POST request. It simply has the web server echo back the HTTP request sent by the client. *Importantly*, this method will echo back the entire request, including all the cookies sent by the client.

Assume that neither web servers nor web browsers set any restriction preventing the use of TRACE requests. In particular, JavaScript can send a TRACE request and receive its response.[1]

(a) Mallory knows that `victim.com` is vulnerable to an XSS attack. Mallory also knows that this website stores a session cookie on the user's browser.

However, Mallory's injected JavaScript is unable to access that session cookie.

⋄ **Question:** What is a likely reason for which Mallory's injected JavaScript code failed to access the session cookie? (answer in less than 10 words)

HTTPOnly flag is set for these cookies.

(b) Explain how Mallory's injected JavaScript can steal the cookie and send the cookie to Mallory's personal website `evil.com`.

You don't need to provide the specific JavaScript code; rather, you only need to provide the high-level ideas. Please arrange your answer in a few steps (no more than three steps).

⋄ **Outline:** Mallory's injected JavaScript, which consists of no more than three steps:

1. Use XSS attack to execute injected JavaScript

2. JavaScript will construct a TRACE request with source address of evil.com

3. The cookie will be sent to server and be echoed back to evil.com

(Answer concisely, put only a single line of text above the dashes, and do not exceed the space.)

[1] These days, browsers now all block JavaScript from sending TRACE requests because of this particular attack.

Problem 9 *Firewalls and DDoS* **(12 points)**

(a) TRUE or FALSE: A stateless firewall can block in-bound **TCP** connections on destination port 80 to devices on the internal network, while still allowing these devices to make out-bound connections using destination port 80.

○ TRUE ● FALSE

(b) TRUE or FALSE: A stateless firewall can block in-bound **UDP** connections on destination port 53 to devices on the internal network, while still allowing these devices to make out-bound connections using destination port 53.

● TRUE ○ FALSE

(c) TRUE or FALSE: SYN flooding attacks can be effectively prevented by rate-limiting the number of TCP connections from a given IP address.

○ TRUE ● FALSE

(d) Consider the following implementation of SYN cookies. During the TCP handshake, the implementation sets the sequence number of the SYN ACK packet to be the first 32 bits of HMAC$_k(t)$, where t is the time rounded to the nearest second and k is a secret key known only by the server.

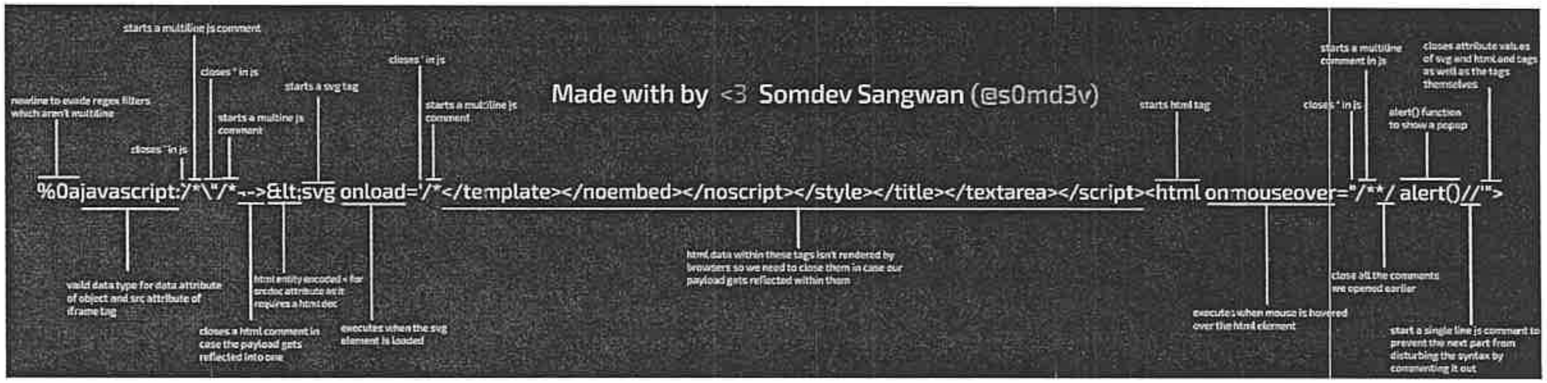
◇ **Question:** Explain why such a design of SYN cookies makes it easier for an off-path attacker to spoof TCP packets.

The time is rounded to second, which means it will not change for 1 second

(e) What additional piece of information could you include in the MAC in order to fix the problem above?

◇ **Answer *concisely*:**

Include time information in milisecond or microsecond



Made with by <3 Somdev Sangwan (@s0md3v)
starts a multiline js comment
closes " in js
starts a svg tag
closes ' in js
newline to evade regex filters which aren't multiline
starts a multine js comment
starts a multiline js comment
closes " in js
starts html tag
starts a multiline comment in js
closes " in js
closes attribute values of svg and html and tags as well as the tags themselves
alert() function to show a popup
valid data type for data attribute of object and src attribute of iframe tag
html entity encoded < for srcdoc attribute as it requires a html doc
closes a html comment in case the payload gets reflected into one
executes when the svg element is loaded
html data within these tags isn't rendered by browsers so we need to close them in case our payload gets reflected within them
executes when mouse is hovered over the html element
close all the comments we opened earlier
start a single line js comment to prevent the next part from disturbing the syntax by commenting it out

```
%0ajavascript:/*\"/*-->&lt;svg onload='/*</template></noembed></noscript></style></title></textarea></script><html onmouseover="/**/ alert()//'">
```

Figure 1: An amazing XSS polyglot payload

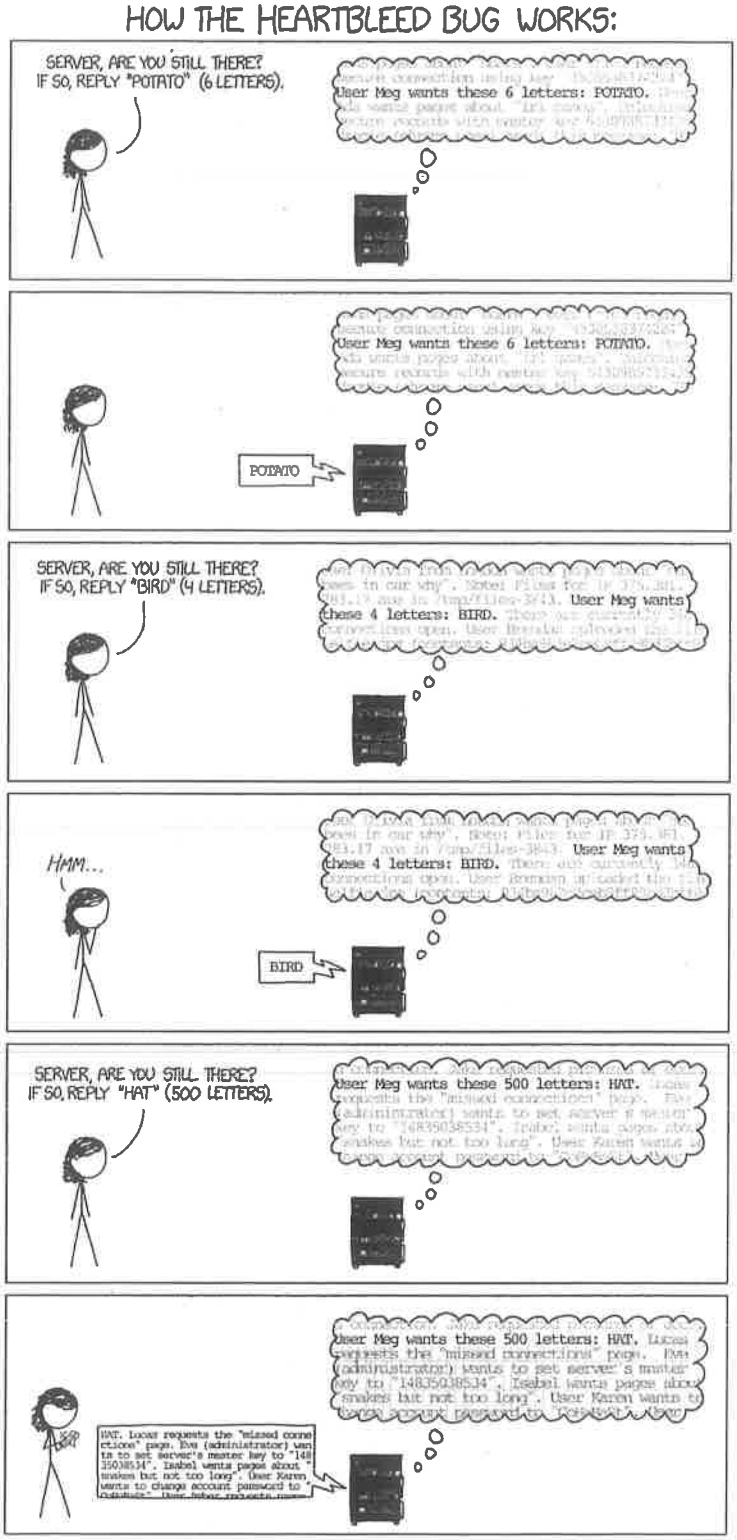

HOW THE HEARTBLEED BUG WORKS:
SERVER, ARE YOU STILL THERE? IF SO, REPLY "POTATO" (6 LETTERS).
User Meg wants these 6 letters: POTATO.
User Meg wants these 6 letters: POTATO.
POTATO
SERVER, ARE YOU STILL THERE? IF SO, REPLY "BIRD" (4 LETTERS).
User Meg wants these 4 letters: BIRD.
HMM...
User Meg wants these 4 letters: BIRD.
BIRD
SERVER, ARE YOU STILL THERE? IF SO, REPLY "HAT" (500 LETTERS).
User Meg wants these 500 letters: HAT.
User Meg wants these 500 letters: HAT. Lucas requests the "missed connections" page. Eve (administrator) wants to set server's master key to "14835038534". Isabel wants pages about "snakes but not too long". User Karen wants to change account password to
HAT. Lucas requests the "missed connections" page. Eve (administrator) wants to set server's master key to "14835038534". Isabel wants pages about "snakes but not too long". User Karen wants to change account password to "

Figure 2: XKCD Explains Heartbleed