HW 2 Security II

1) What issues have prevented the deployment of TLS 1.3 and why has the RFC been redrafted a bunch of times?

Like most things in cryptology when ideas are implemented in practice things tend to go wrong. In this case, when TLS 1.3 was implemented it became clear that it is
incompatible with the the way the internet evolved over time.

Because there are so many devices supporting different levels of security on the internet, browsers needed
to support multiple versions of TLS. Therefore the server could force a connection with an older security protocol if that's all it supported and the client would connect just with a slower startup time. This wasn't a problem until it became clear that an attacker can force a downgrade by pretending the server only supported an older protocol. The problem was exploited by a vulnerability found in an older security protocol SSLv3. Luckily it isn't widely used and support for it has mostly been removed. While this particular attack has been mostly mitigated, it proves that downgrade attacks are a real concern and designers have to prepare for them.

This insecure downgrading was removed and the community felt it was time to release 1.3 again. Unfortunately most servers didn't implement this version negotiation properly and 1.2 servers would disconnect instead of negotiating. This meant that clients would have to downgrade to 1.2 to have the connection. Another manifestation of the downgrade issue. To resolve this the RFC was rewritten again with a new an improved version negotiation that treats 1.3 as a protocol upgrade rather than the default.

Again this seemed to solve the problem, and it did for most clients and servers. Unfortunately there are a lot of other boxes that need to deal with TLS to move packets from client to server. They weren't programmed with these changes in mind or programmed flexibly enough to accomodate these chanes. As a result connections to failed. (Some reasons for the connection failures included 1.3 removing fields that were considered essential in impelementation) 1.3/1.2 in theory aren't the problem. It's the fact that it is incompatible with the way 1.2 was implemented. And so, the RFC was redrafted in a way that 1.3 appeared like 1.2 not only to clients and servers but also middleboxes. This included re introducing fields back into the server hello message among other things. Since then the RFC has been consistently redrafted to support middleboxes.

In the end the reasons the RFC has been re written so many times is twofold. 1) It's been shown that downgrade attacks are real and dangerous. 2) Implementing an upgrade that prevents these downgrade attacks and remains compatible with everything that's on the internet is really hard.

2) Describe the POODLE attack

POODLE is a type of downgrade attack. While most of the internet uses TLS 1.3 1.2 and 1.1 most clients and servers supported the older SSLv3 protocol. It is the predecessor to the TLS protocols and in order to keep new machines compatible with older machines the newer protocols were designed to allow for the older forms of communication.
This wasn't considered a problem because there was no severe issue with the SSLv3 protocol. This is no longer true.

POODLE works by first exploiting this downgrading property and then exploiting a flaw SSLv3's CBC encryption. The flaw is that the block cipher padding
is non deterministic and not covered by the MAC. This means the integrity of the padding cannot be determined during decryption making it an attack vector.

If the attacker swaps blocks in the message they can learn information about the plain text based on weather or not the message is still accepted as if it had the right padding. This leads to a padding oracle attack. If the encrypted text is deemed valid it leaks information about 1 byte of plain text. Working backwards the attacker can then decrypt the entire message.

In detail this works by exploiting the padding scheme. For a block with n bytes of padding the value of the padding bytes is n. Given K is a the last block in the sequence, if the attacker crafts a request that gets accepted it knows that D(k-1) XOR K = 1. This now leaks what the value of the last byte of K-1 is. The attacker can create a last block full of padding which functions as K with known values. The attacker can then work incrementally backwards until the whole message is decrypted. The attack works a byte at a time so there's only 256 values to search. This makes the attack feasible.

The only thing left for the attacker to figure out is how big the actual payload is since the original padding hides the actual length of the payload and the attacker needs control of the last block. The attacker can then send requests of increasing length until a block boundary is crossed and the server responds with an extra encrypted block.

Because this attack is predicated on the SSLv3 protocol in can be mitiagated by not allowing that form of communication. Unfortunately this makes communication with older servers inherently insecure.

So in summary the attack works as follows:
1) The attacker acts as the man in the middle of a normal cookie bearing client server interaction to gain access to the cookie
2) The malicious client manipulates the TLS protocol to downgrade to SSLv3
3) The attacker can send specialized requests to the server that decrypt the cookie byte by byte

3) IPSec History Lecture

a) How did politics contribute to IPsec having an integrity only option (AH)?

There are US export controls on cryptography. There needs to be a license to export confidential technology. Authentication technology is not restricted.

b) What area of expertise was lacking in the in-person IETF meetings on IPsec?

The designers had a somewhat limited cryptographic knowledge.

c) What was the explanation for having sequence numbers in swipe?

They were included "to protect against replay" attacks.

d) What benefit did sequence numbers end up providing IP sec?

They were used to prevent malicious retransmission of packets.

e) Skip was proposed by who? and why wasn't it chosen?

Skip was proposed by Sun, and it wasn't chosen because it didn't allow for a change in Diffie Helman parameters. Sun had recently had a problem with choosing bad DH parameters.

4) The chrome browser and usability

a) What are the steps a user must perform to see the contents of stored cookies?

settings -> advanced_settings -> privacy and security -> content settings -> cookies -> see all cookies and site data -> click on particular cookie to see content

b) How would you re design this so the user can 1) easily see the contents of cookies and 2) reduce the number of steps to get to the point where the cookies are

To address concern #2 I would put a link to the view cookies page in the dropdown where the settings link is.
To address concern #1 I think they actually do an ok job at this, but it's all stored in local storage.

Making it accessible via the default file system viewer would be a nice touch. Also if there were an alert with all the cookie data any time a website added something to persistent local storage I think that would make the data easier to digest. One at a time rather than a giant block.

c) What are handlers and why are they there? Why are they dangerous?

There are many different types of links on the internet with different types of protocols. The most common protocol http/https is and should be supported by the browser. Links that return types of information that the browser can't understand like email links, calendar links with other unusual protocols need to be handled by other applications. Not the browser. As such chrome allows users to add handlers that automatically open these other applications for these other types of links. The default is to allow this behavior because most people just want to click and open the content. They don't really care what opens it.
On the web whenever a email address is listed usually it contains a mailto: protocol link. When a user clicks on this usually a new email screen appears with the links associated email address in the to section. What's special about this is that the new email screen is in the users email client not the browser. If one were to disable the handlers this interaction could not take place.

The risk here is that handlers provide links a way to get outside the sandbox of the browser. Usually activity in the browser is restricted to the browser. Now attackers have a way to exploit this handler feature to further exploit another application. Users might not want to enable handlers to prevent malicious activity from escaping the browser.

5) How do requirements placed on passwords contribute to the insecurity of the password? -- UNSURE

Users will always do the bare minimum effort required. As such the passwords minimum requirements need to be secure. However poor usability often means poor security, and users will find workarounds if security measures are too inconvenient. When users find workarounds the whole system's security suffers.

Some requirements that might cause problems.
1) What can be a password
Often times password requirements include things like Length, special characters, numbers etc. This essentially provides a blueprint to an attacker as to what passwords to brute force. They don't need to search all possibilities. Just all possibilities that fit the minimum password requirements

2) How often passwords change
People tend to not want to change passwords, and requiring frequent passwords changes can lead to insecure passwords. Generally when required to change people make some small modification to their old password to make the new one. This often leads to poor passwords like

some word followed by a number that changes. This allows attackers to guess passwords in a way that keeps these transformations in mind and maybe break in. In one study attackers hacked 17% of accounts this way.
https://www.cs.unc.edu/~reiter/papers/2010/CCS.pdf

3) Overly complex passwords
If the password requirements are generally overly complex most users don't want to remember them. Instead they tend to write these things down on a post-it or other note. When passwords are written down they are vulnerable to being stolen by the note being taken or a photograph being taken of the plaintext.

4) Allowing usernames/common words to be part of the password
Often times attackers can brute force passwords by using dictionary attacks. This is a way of shrinking the search space. Rather than trying to guess all possibilities just guessing common word permutations etc provides a way to feasibly brute force passwords. If the password requirements allow especially common passwords like "password" or variations of the username it makes the system more vulnerable.

5) Password reset procedure
While not entirely related to the question, a system with a vulnerable password reset feature is just as vulnerable as a system with a vulnerable password. Often times passwords can be reset using commonly known information. If an attacker can use Google to break the security questions they can break into the account.

6) Find an article describing failure rates or issues with facial recognition as a method of authentication?

On the vulnerability of face recognition systems towards morphed face attacks authored by: Ulrich Scherhag et al.
www.researchgate.net/publication/317245046_On_the_vulnerability_of_face_recognition_systems_towards_morphed_face_attacks

This particular paper deals with attacks in which someone is presenting a form of identification in which they control the image on the ID. The verification system is then processing if the photo presented looks like the person presenting the photo. The particular attack they deal with is the "morphed face attack". The scenario they use as an example is immigration.

In many countries to get a passport one submits a photo which is then scanned and passed to the passport creation office. In this scenario the morphed face attack takes place as follows. Person A is able to travel and has a clean record while person B has travel restrictions for any number of reasons. Person A submits a photo and gets a legitimate passport. Person B uses the passport. Normally this attack is prevented because person A and B don't look alike both to people and

computers. However in a morphed face attack the photo that person A submitted is a photo containing features of both A and B's face. So the resulting photo looks something like both people. A human will notice that it's not exact, but given some explaining and given people A and B probably look somewhat similar anyway it's easy to understand a human may not catch this. The question is whether or not a facial recognition system will. One way to think of this would be a hash collision. Both people map to the same document.

This paper doesn't implement a new version of this attack or propose this attack but evaluates the effectiveness of this attack with different facial recognition systems. They also put together a new dataset of morphed faces and originals for others to use.

In their evaluation they examine the effect of morphed face attacks on 2 different facial recognition systems. The first was the commercial VeriLook SDK and the other was the opensourced OpenFace. They also ran their experiments with both HP and Ricoh scanned images as well as the original.

The success of this attack in their charts and figures is somewhat ambiguous. They found that using Veriface with the original scanned images, HP scanned images, and RICOH scanned images had a attack success probability of 100%, 99.7%, and 97.3% respectively. With OpenFace SDK they found the chances of the attack succeeding in the same order was 80.1%, 62.3%, 70.8%. While I think that these numbers are a bit inflated they are high enough to show that there is a real probability that morphed face attacks could be successful.

7) What main issues are the papers addressing? Summarize the experiments about what worked and did not work for addressing this issue.

When you amp up security at the expense of usability users develop a callousness to security warnings. These papers suggest solutions to approach this problem and then evaluate the effectiveness of different approaches.

The authors published a set of guidelines for security warnings that formalize the activity that takes place when a dialogue appears. They also offer a useful table of common problems when users ignore warnings and provide published suggested solutions to those problems. The general advice is to keep things simple, clear, concise, and digestible for the user not just the programmer.

The first paper published at SOUPS 2013 from CMU in conjunction with Microsoft deals with this problem as it relates to permission alerts from the browser. Different applications ask to install or update various supporting applications to run and as a security precaution the user needs to approve these actions. These alerts appear often and most users don't check what the alert is asking for. They show examples of poor dialogues and suggest improvements both in the

structure and content to make warnings clear, concise, and understandable.

The authors' approach to this problem was to modify alerts so that they are more visually stimulating or more interactive than the standard pop ups. Some of the changes they made involved highlighting pieces of the popup, small animations, minor text changes and others. They branded these modifications *attractors*. They also differentiate between inhibitve attractors and non inhibitve attractors. Inhibitve attractors are those which prevent the user from accepting or refusing the pop up immediately. They sent participants in a study to 3rd party sites and tested their acceptance of popups with both control sites and a manipulated site with the modified popups. They also presented participants with both benign and suspicious scenarios on the study controlled site. In the suspicious scenarios they tested both installation of downloads from a suspicious publisher and granting suspicious permission privileges to the downloaded software. The results showed that the attractors were rather successful but slowed the users down significantly for those who wanted to disregard the warning. The percentage decrease in suspicious activity approval from the control group varied from 25-85% with most attractors falling somewhere in the middle of that range. ANSI (part of the pop up had text written in ANSI) performed surprisingly well and Type (required user to type something) performed the best across both of the first 2 experiments. Finally they ran a 3rd experiment testing *habituation*. Habituation is the effect of seeing something repeatedly and becoming less sensitive. Participants went through a habituation period, and then shown the dialogue when the attractor. They showed that all 5 inhibitve attractors performed significantly better than the controls while the non inhibitive attractors didn't have the same effect in the face of habituation. In the end they determined that inhibitive attractors provide significant improvement over the default, but this improvement comes at a cost of efficiency.

A year later the authors revisited the habituation problem. The previous experiment tested high levels of habituation not the normal low levels of habituation that come with prolonged exposure over a long period of time. This time they tested the effect of the attractors with high and low level habituation. They used the same attractors as before and designed the experiments in a similar fashion. They measured habituation in two ways, number of times shown the pop up and number of total seconds exposed to the pop up. In the previous experiments they found that only the attractors that introduced a delay in the process had effective results. In this experiment they found that attractors that introduced delay but didn't require the user to interact with the field worked the best in low habituation environments, but failed in high habituation situations. They found that attractors that required user activity were the most resistant to habituation. The authors claim with time people interacted with these attractors efficiently. Again they found that least usable measures, swipe (users must swipe a piece of text) and type,were the most effective, but they maintain that the swipe attractor has low enough overhead that this may be worth considering as an alternative to the current dialogue style.