Question 1 ***Intrusion Detection*** **(15 min)**

FooCorp is deciding which intrusion detection method to employ in a few target scenarios. In each part, consider which of the intrusion detection methods learned in class would be most appropriate, and justify why. Try to be as specific as possible.

(a) FooCorp wants to detect attacks for a specific vulnerability that may exist in some of their web servers.

> **Solution:** FooCorp is probably best off using vulnerability signatures to try to detect attacks for the specific vulnerability. If there are key elements to all attacks on the vulnerability, these can be listened for and detected.

(b) FooCorp is using HTTPS, but all of their services use the same modular web framework. They are interested in detecting any time their servers receive arguments that are suspicious, in real-time.

> **Solution:** Some form of HIDS would probably be most appropriate here. A HIDS can check arguments and detect suspicious ones in real-time, and since the web framework used is modular and extendible, it wouldn't be very difficult to deploy HIDS software for every service.

(c) FooCorp is a diverse company, with a wide variety of different web servers built on top of different web frameworks, offering different services. They wish to detect suspicious arguments for all of their services. Every service uses HTTP and not HTTPS, and FooCorp has a low budget for security, but they want real-time detection.

> **Solution:** A NIDS would probably be best here. A NIDS is endpoint-agnostic, so the variety of servers that FooCorp is using doesn't affect a NIDS. It also only involves writing one piece of software and deploying at one point, so it would be a cheap solution.

(d) FooCorp again has many different web servers built on different web frameworks, but each uses the same logging format. They are using HTTPS, and do not need real-time detection.

Solution: Logfile analysis would probably work best. Since FooCorp does not need to detect attacks in real-time, a HIDS would probably be overkill here, while a NIDS wouldn't be as powerful.

Question 2 ***Detection Tradeoffs*** **(15 min)**

Suppose that S is a network-based intrusion detector that works by passively analyzing individual UDP and TCP packets. Suppose that A is a host-based intrusion detector that is a component of the browser that processes and analyzes individual URLs before they are loaded by the browser. Suppose S has false positive rate S_P and false negative rate S_N, and A has false positive rate A_P and false negative rate A_N.

Your company decides to build a hybrid scheme for detecting malicious URLs. The hybrid scheme works by combining scheme S and scheme A, running both in parallel on the same traffic. The combination could be done in one of two ways. Scheme H_E would generate an alert if for a given network connection either scheme S or scheme A generates an alert. Scheme H_B would generate an alert only if both scheme S and scheme A generate an alert for the same connection. (Assume that there is only one URL in each network connection.)

(a) Assuming that decisions made by S and A are well-modeled as independent processes, and ignoring any concerns regarding evasion, what can you say about the false positives and false negatives of H_B and H_E? In terms of S_P, S_N, A_P, A_N, what are the false positive and false negative rates for H_B and H_E.

Solution: The key insight here is that alarms by H_B will be a subset of the alarms generated by H_E. Since H_B will generate fewer alarms for non-malicious activities, it will have less false positives. On the other hand, because it generates fewer alarms, it might miss more malicious activity, implying more false negatives. The false positive rate for H_E would be $S_P + A_P - S_P A_P$, and for H_B would be $S_P A_P$. Similarly, the false negative rate for H_E would be $S_N A_N$ and for H_B would be $S_N + A_N - S_N A_N$

(b) If deploying the hybrid scheme in a new environment, is one of H_E and H_B clearly better? If not, what environment parameters would help determine whether H_E or H_B is better, and for each parameter p, increasing p favors which hybrid scheme?

Solution: In the absence of more data, particularly the cost of false positive and false negatives, as well as the rate of malicious and non-malicious activity, it is impossible to make any decision.

Increasing the cost of false-positives and the rate of non-malicious activity favors scheme H_B, while increasing the cost of false-negatives, and the rate of malicious activity favors scheme H_E.

Question 3 ***Censorship and Anonymity*** **(10 min)**

(a) You are a resident of the country of Censorshipistan (a former *Eastern Block* state). You suspect that your country is employing an *on-path* censorship device to block content deemed objectionable by the ruling party. How might you detect that you are being censored?

> **Solution:** The key consideration here is that the device is *on-path*, meaning that the device lacks the ability to directly block inbound packets; instead, it needs to inject new packets. We can leverage this behavior to detect censorship behavior. For example, if we observe a RST packet followed closely after by a normal payload packet, then that behavior is highly unusual if both were indeed sent by the same sender (since after sending the RST, the sender should be all done), but often unavoidable for an on-path device that's injecting additional packets but can't prevent the transmission of the original packets. Similarly, observing multiple DNS responses, or SYN+ACK packets in addition to RSTs in response to initial SYNs, also strongly suggest the presence of an on-path injector.
>
> This behavior has been used to study the prevalence of on-path injectors in the wild. See for example the paper http://www.icir.org/vern/papers/reset-injection.ndss09.pdf if you're curious about a study based on this technique.
>
> A completely different approach can be used if the censor does not take care to vary the properties of the injected packets. For example, if all injected RST's had the same TTL or TCP "advertised window", you could filter out such packets in order to ignore them. Studies of the Great Firewall have found such behavior in some circumstances.

(b) After determining that you are indeed being censored, you decide to evade the censorship using the Tor anonymity software.[1] How might the government of Censorshipistan detect your Tor usage, and block it?

> **Solution:**
>
> The government might attempt to detect usage of Tor via one of several possible ways: (1) observe downloading of the software; (2) observe access to the servers that supply information about the Tor nodes (necessary for the client's Tor software to set up the onion path; see below); (3) observe access to those nodes themselves (first step of the onion path); or (4) look for some peculiarity of Tor traffic that occurs dynamically as Tor's being used. This latter could be based on detecting the certificates Tor uses for TLS connections, or distinct timing patterns in its message transmissions.

[1] https://www.torproject.org/

These approaches all have corresponding blocking approaches:

- Block the users' ability to download the software, either by using DNS reply injection when users try to access `www.torproject.org`, or by blocking the IP address(s) that that server uses.
- Block the users' ability to obtain a list of Tor nodes. The "onion routing" approach used by Tor requires that clients obtain a list of all of the Tor network's nodes so that the client can decide which hops through the network its traffic will take. Clients do so by accessing one of several (mirrored) "directory servers". The censor can block access to these similar to blocking access to the software itself.
- Block the users' access to the Tor nodes. The censor can act like a Tor client and download the list of nodes from the directory, then install the corresponding IP addresses in a blacklist.
- Upon dynamically detecting use of Tor, disrupt such connections as they occur, for example by injecting TCP RSTs.

Question 4 ***Tracking*** **(10 min)**

(a) Sam is researching which pair of headphones to buy, and he visits a few different blogs outlining the pros and cons of each model. He then visits his favorite social media site, FaceSpace, and notices incredibly targeted ads, which advertise specific headphone models. He goes back to each of the blogs that he visited, and sees that each one had an iframe tag containing an embedded FaceSpace like button for that page. Looking closer, he sees that each iframe source URL is structured as `facespace.com/like_button?id=<ID>`, and that each blog page he visited has a different ID. How did FaceSpace figure out that Sam is interested in purchasing a pair of headphones?

> **Solution:** When requesting the FaceSpace like button on each blog site, Sam's browser sends Sam's FaceSpace cookies, allowing FaceSpace to identify Sam. FaceSpace can identify the page using the page ID passed as a parameter.

(b) Sam figures that he can maintain his privacy from FaceSpace simply by removing any FaceSpace like buttons embedded onto the webpages he views. So he writes a small extension to his browser that removes all FaceSpace like buttons before loading the page. Content, he continues browsing, this time comparing different graphics cards. Unfortunately, when Sam goes back to FaceSpace, his page is filled with graphics card ads. Sam concludes that some of the sites he visted must be cooperating in some way with FaceSpace, but isn't sure about the details. What are some ways that FaceSpace could've figured out that Sam is interested in graphics cards?

> **Solution:** There are a couple ways, and each involves FaceSpace having some relationship with the blog site. One possible example is that the blog site can include a hidden image with source `facespace.com/<blog site>?info=<some relevant info>`. This request will then be sent with Sam's FaceSpace cookies, and FaceSpace can learn arbitrary information that the blog site provides. Another way is that the blog site can embed a script on FaceSpace's behalf, which submits a request to a URI structured as above. The same origin policy guarantees that the blog site cannot see the response, but FaceSpace will still receive the request, along with Sam's FaceSpace cookies.

(c) Sam is now done with FaceSpace and their invasive tracking. He decides to clear all of his cookies and go back to browsing different types of headphones. He reads a blog comparing two headphone models, and clicks a link from that blog to a RedFeed post about cats, only to find another advertisement about headphones on the page. But when Sam reviewed his traffic log, no cookies were sent to RedFeed in his request, and no extra information was passed in the URL. How did RedFeed figure out that Sam is interested in headphones?

Solution: RedFeed received the referrer header, which specified where Sam came from before visiting RedFeed. From there, they could extrapolate that Sam is interested in purchasing headphones.

(d) What are some ways of avoiding web-based tracking? List some pros and cons for each.

Solution: There are a few mitigations for web tracking as follows:

- Private browsing allows you to browse with a new blank set of cookies. This set of cookies persists for as long as the private window is open. Private browsing works well, but it is inconvenient for the user, and can still reveal information if a user logs into websites in a private window or just browses for too long.
- Setting the Do Not Track flag tells web servers that you want to opt out of tracking. Unfortunately, it is entirely up to web servers to honor this or not, so there is no guarantee of privacy.
- Browser extentions (Ghostery): Recognize third-party tracking scripts on web pages, and blocks HTTP requests to those sites. While Ghostery works well to track well-known scripts, new tracking scripts need to be added to their library to be recognized, which may take some time. Additionally, users who opt in have data sent back to Ghostery's servers, which is anonymized and sold to ad companies.