WPI

CS 4404 - Network Security

Mission 1 (25 points)
Launched: Monday, Oct. 24, 2022
Due: Thursday, Nov. 10, 2022 at 11:59:00pm

Mission 1: Indecision 2022: Securing Shueworld's Elections

Shueworld, a distant planet near the star Rigel (the right foot of the star constellation Orion), has a planet-wide government. Every 3 years, the citizens of Shueworld vote on who will be the Sneaker of the House, which is the most powerful position in the government. Naturally, multiple factions vie for this honor and hope that their chosen representative will be selected Sneaker by the populace.

For arcane reasons, the Sneaker's election is broken up into individual precincts, called "laces." After doing some sole-searching, each citizen votes at a local lace. The laces then tabulate their votes and determine who won the lace. The candidate with the most laces wins. In the event the laces result in a tie, the two candidates both serve in a united government, known as a "knot." Unfortunately, most knots either unravel or become extremely difficult to separate in future elections.

In previous years, the citizens of Shueworld voted at their local lace by making a shoe print in a column of mud associated with their chosen candidate. Due to logistical considerations (most notably, rainy days and the growing fashion trend of oversized clown shoes) and privacy concerns, the citizens will now vote using computers available at each lace and some may choose to vote using their phones through the lace's wireless network. The lace computers are naturally connected to each other, the local lace tabulator system, the tabulators at other laces, and to the greater Shueworld wide web (SWW).

Figure 1: Shueworld clearly lacks the budget for a graphic designer for their logo....

While most citizens have embraced the move to computerized elections (with the notable exception of the Galosh faction, which embraced mud-related voting thinks online voting stinks), many are concerned about the security of the election and making sure the results are correct. In particular, they do not want the wrong Sneaker candidate to get the Boot, which is malodorous footwear that the loser must wear until the next election to signify de-feet.

Mission Objectives and Parameters

The objective of this mission is for students to learn about the technologies that are likely to be used for election infrastructure. The students should learn about the communication dependencies, election assets, actors, and security goals. The students should study the tools and techniques that are used to violate these security goals and methods that can best ensure the goals are met.

Since the Shueworld government did not specify the details of how the computerized system would work, students may make reasonable assumptions about these details or consider a range of options. Students may wish to examine the infrastructure used in computerized elections on Earth as a potential model for how Shueworld may end up creating their infrastructure. Students may treat the isolated Zoo lab network (available via the web interface on the secnet.cs.wpi.edu system) as a mock copy of the SWW.

This mission is designed so that different students teams may explore different aspects of the election security problem. Students are encouraged to communicate with the instructor about their plans to get feedback. Students should be careful that they center their learning on network security, rather than other subfields (e.g., software security, webware, programming languages). As they consult the remainder of the project specification, students will want to ensure their project can meaningfully cover each of the following phases.

An Example Project Structure

While some students may embrace the open-ended nature of the mission, others may prefer a bit more structured mission. The following description is an example structure that students may choose to use if they wish. The decision on whether to use the example or not should have little impact on the project grade. The grading criteria in the rubric is designed around learning outcomes, creativity and initiative. Students can demonstrate such concept mastery using the example or self-developed plans.

Consider the following three components:

- **A Voter Web Interface:** How will voters complete their ballots? How will they identify themselves? How will they verify their ballots were completed and/or recorded correctly? How will the Shueworld government ensure only eligible voters vote and that they only vote once? Where do ballots go when they are submitted? What might Mittens Romulan (the ruler of a nearby hostile planet) do to meddle with Shueworld's web infrastructure? What outcomes would Mittens be happy with?

  Importantly, web infrastructure is pretty easy to throw together. But, that infrastructure has specific security considerations. If we use PHP for this, what do we need to worry about? What about if we use Node.js? Or a Python server? What are the best practices for securing that infrastructure? What about the underlying server infrastructure (e.g., Apache or nginx)? How do we secure communication between clients and the web servers? How about between the web servers and any central databases? Are there vulnerability scanners for this infrastructure? How can you test that it is setup right? How could reasonable people do it wrong? How could that make them vulnerable to attack? How would an attacker exploit that vulnerability? How would you remediate it and show it is solved?

- **A Database Server:** Where will the ballots go? Will that server be publicly accessible; why or why not? What protocols do they use? Are they vulnerable? If so, can they be secured? What might Mittens try to mess with the database?

- **A Tabulator API:** How will the Shueworld government access the database to get the results? What risks does this introduce? How can they be minimized?

With just these three pieces, we have a significant amount of attack space and goals for both attackers and defenders to consider. We do not need to build out complete election infrastructure, but by building things in practice, we get the lower-level "nuts and bolts" view needed to speak reasonably about security. It is fine to argue "we need to ensure confidentiality" at the high level. But to show mastery of the concept, we need to go into detail with an implementation and talk about the intricacies and design decisions of enabling that. Talking about the mechanisms one built to authenticate a web server from a client's perspective shows greater depth of knowledge.

With this starting point, we need to brainstorm and figure out the full combination of things we must consider. We should go through the security goals (hint: "CIAA" is the common acronym), the defender's trusted computing base (the "trusted" things that might not be "trustworthy"), the attacker's techniques (hint: "alteration" is one), the defender's control techniques (e.g., "deflection"), the actors, and assets. In essence, by reviewing the topics covered in the first couple lectures, we have a good place to start. We can then think about how to achieve these with our specific software selections and tools (e.g., searching for articles involving data confidentiality in PHP web servers).

Note that while the "CIAA" security goals are a good starting point, they may manifest as higher-level, application-specific security goals. For example, the "confidentiality" goal could present as "my spouse cannot discover who I voted for" and "authenticity" may manifest as "I have proof my vote was recorded correctly." But, we may also have compound security goals that are application-specific like "I have proof my vote was recorded correctly, but my spouse cannot use that proof to discover who I voted for." This latter case might be reformulated into security goals such as preventing vote-buying schemes (where an individual demands proof of a vote cast a particular way as a prerequisite to providing financial compensation).

Once we have all this, we can tackle the remaining phases.

Reconnaissance Phase

Students must determine the security goals of each part of an election. For each election component, the students must 1) determine the consequences of the security goal not being met, 2) propose ways that an attacker could undermine each security goal, and 3) propose countermeasures that would minimize the risk that a goal would not be met. Students should ensure they have identified at least one network-based attack vector for use in the next two phases.

In the reconnaissance phase, students should consider a broad range of attackers, motives, and strategies. The attackers may have the desire to ensure a given Sneaker is elected, to cause chaos, or to undermine the populace's faith in the election outcome. After all, a Sneaker that stinks is as bad as no Sneaker at all.

Infrastructure Building Phase

Using the virtual machines on the isolated computer network in the Zoo lab, the students should construct the relevant portion of the election infrastructure and a system from which the adversary will launch his or her attack. Students should focus on assembling only the minimal election infrastructure needed to demonstrate the attack and its vulnerabilities. For example, if the system in which a person casts a ballot is needed, a simple HTML ballot with a few choices and a minimal backend form processor may suffice. An elaborately built system is unlikely to be necessary and this may divert student time from efforts that offer a more significant learning (and grading) impact.

Attack Phase

Students should select at least one network-based attack from the reconnaissance phase and consider how to implement it. The attack must have a significant impact on the election. For logistical reasons surrounding shared infrastructure, the chosen attack must not be a resource exhaustion attack (e.g., anything that saturates the network, memory, or computational resources of the involved systems).

Students should avoid trivial attacks that exploit application vulnerabilities that the students themselves introduce. For example, both a "vote stuffing" attack in which the voting system does not attempt to prevent duplicate votes or an SQL injection attack in which the students create an application vulnerable to SQL injection are considered trivial. Part of the scoring for the attack phase is realism, ambition, and degree of learning.

Students must provide the necessary details of the configuration setup, the attack itself, and results needed for a reviewer to know the attack is real, to replicate the experiment, and to validate the results in the Mission Debriefing Report. The attack should have measurable, quantifiable outcomes and an indication of the resources needed to launch the attack. Consider the Mission Debriefing Report the team's only opportunity to convince the current Sneaker of the House that there is a legitimate concern in the election with evidence that allows the Sneaker to hire consultants to verify the team's work.

Helpful hint: It is probably useful for students to learn about tools like `nmap` and the `burp` suite. Those are commonly used by penetration testers to discover what systems run on a network and to evaluate those systems. If those tools are designed to identify a particular vulnerability, it is likely that vulnerability appears often in practice.

Defense Phase

After completing the attack phase, the students must create a proposed defense against the attack. The defense should be effective and realistic for the Shueworld government to deploy. For example, while DNA testing may be useful for ensuring a voter's identity, it is probably unrealistic; it would likely yield significant backlash from the populace and would be too expensive for the government to implement. (It is also a little difficult to implement and demonstrate in the available VMs.) The description of any defense should include a discussion on feasibility and costs: what will the government have to do, what resources are required, and why is it reasonable to assume those resources are available?

Students should demonstrate that the defense successfully mitigates the attack. They must provide all details of the defense, including any tools and configuration changes, along with log evidence demonstrating the attack was blocked. The defense should allow the team to demonstrate significant learning of new knowledge and/or ingenuity. A correct, but trivial, defense may not earn many points according to the grading rubric.

Helpful hint: TLS is probably going to be part of a defense because it provides a wide range of security goals. While installing a package that does all the TLS setup in a single click may go a long way in securing a system, it may not let a student show mastery of the defense. To get away from a trivial solution, students might dig into the TLS process in more detail (e.g., setting up a CA, doing CSRs, installing the results, setting up root CAs in browsers) and explaining what each piece does. Alternatively, students may go the "one click" install route for TLS and focus on a problem that remains even if TLS is present and build a defense for that. The learning outcome goal is to show students have greatly expanded their knowledge by investigating concepts in detail.

Mission Debriefing Report

The Mission Debriefing Report is the team's opportunity to explain the identified security goals and attack vectors in the election infrastructure to the government of Shueworld, including all the results from the reconnaissance phase. The report should then focus on a single network-based attack and demonstrate the negative consequences of not addressing the vulnerability the attack exploits. The report should then provide a feasible solution to defend the infrastructure.

The Mission Debriefing Report should be comprehensive and provide conclusive supporting evidence. It is likely that such a report will be at least five pages of write-up and any figures, followed by an appendix of supporting evidence (e.g., screenshots, output of terminal windows, log file output). In addition to the write-up, the debriefing report should include additional supporting files, such as source code or configuration files. For the report to be scientifically valid, enough detail must be provided that would enable the grader or another member of the class to create and run the experiment **without the need to ask any questions**.

Scoping a Mission: Showing "Independent Learning" and Avoiding "Trivial" Outcomes

When considering the amount of time and effort that should go into each mission, students should consider that each mission is roughly 18% of the final grade in a 4000-level CS class. In other words, students should expect to invest a significant amount of time into the missions to explore concepts and learn.

Teams should note that the setup and assumptions for a mission can greatly affect the learning outcomes associated with the team's activity. As an example, consider the "what you know" factor in this mission (which the instructions preclude students from pursuing). If it were allowed by the instructions, a team could explore the concept in a trivial fashion or in a way that would lead to significant learning. Consider the following scenarios sketches:

Trivial Scenario: At Company X, administrative services are protected by usernames and passwords. Because Company X did not take CS 4404, they did not realize they needed to change their default system credentials, which are the username `root` with password `root`. For our infrastructure phase, we set up a basic login web page and prompt for a username and password. For the attack, we write a script that enters the username `root` and password `root` into the supplied URL. We then supply the login page's URL. We found the attack was successful. For the defense phase, we changed the root password to a random 30 character string. We repeated the attack. We found the attack did not succeed and thus consider our defense to be effective.

More Advanced Scenario: At Company Y, administrative services are protected by usernames and passwords. The company has 500 users, each with unique usernames and passwords. The company has an employee search feature on the website where visitor can find information about the employee by entering the user's username or email address, which is common in organizations [citation1]. The organization's username schema is to take the person's first initial and concatenate it with the user's last name to create a username (adding a digit at the end if the username is already in use), which is a common convention [citation2]. We created a list of 500 usernames using a database of common first names and last names. For half the users, we randomly generated passwords. We divided the randomly generated passwords evenly across passwords of length 6, 7, 8, 9, and 10 characters long to test entropy-based tools [citation3]. For the other half of users, we used the [insert details] data set, which contains passwords from a recent breach [citation4]. We randomly picked passwords from that list and set it for each user. For our infrastructure phase, we set up a basic login web page and prompt for a username and password. For the attack, we use two tools: 1) a brute-force password guesser [citation5] and 2) a tool that uses a dictionary of common passwords [citation6]. We used the same common last names database as used in the setup and had a script try each last name, prepended with each of the 26 alphabet characters, in the username search page. We recorded each match. For each match, we then tried both the brute-force and dictionary approach. We found that we were able to recovery X valid usernames and Y valid username/password combinations. From our study, it seemed that short passwords were easily brute-forced, while long passwords were not. But, the dictionary tool seemed effective on the non-random passwords, independent of word length. For the defense, we tried three defenses: 1) we rate-limited the number of queries each person could do for usernames and the number of password guesses per minute (which is recommended by NIST [citation7]), 2) we set the minimum password length to 10 (also recommended by NIST [citation7]), and 3) we prohibited any password containing an English dictionary word of three characters or more, a best practice [citation8]. When we ran the attack, we found [... summary of results...].

A student project exploring the second scenario would earn more points according to the mission rubric, for multiple reasons. The first scenario is indeed realistic, since attacks on default credentials do happen, but it requires the defender to make a silly mistake and the proposed defense is simple. Frankly, the only thing the team would learn is how to setup a web server and login prompt page, how to write a web script, and how to change a password. That seems fairly trivial and would not amount to 18% of a 4000-level CS class.

In contrast, the second scenario is likely to affect more organizations and does not require the defender to make a silly mistake. The second scenario explores common organization practices and password generation techniques from two different angles. It identifies and explores two different password guessing techniques that are used in practice and identifies and uses those tools in practice. The team looked for, and found, a data set of publicly-available leaked passwords. The scenario incorporates the tools into the overall system and collects data about their effectiveness, then analyzes it. The defenses incorporate three different measures that are commonly used by organizations and evaluates the effectiveness in the same setting. In this scenario, the team had to learn about what is done today, what tools exist, how to use those tools, what data sets would be needed for their experiments, how to measure their effectiveness, and how to implement modern defenses. Through its course of actions, that scenario demonstrates significant evidence of independent learning.

Finally, if students have any questions about whether something is "trivial" or a good learning opportunity, they should consult with the course instructor.

Mission Rubric

The mission score can be broken down into five components, each of which are scored independently. The total score is out of 25 points. The following are the components:

1. Reconnaissance: 4 points
2. Infrastructure Building: 5 points
3. Attack: 5 points
4. Defense: 5 points
5. Mission Debriefing: 6 points

When working together, teams are required to evaluate their partners. While no additional credit is awarded for completion of these evaluations, a **3 point penalty** will be assessed to the project grade for any student working in a team who fails to complete a project partner evaluation.

We now describe each of the project components and how they are graded.

Reconnaissance: 4 points

Rubric:

- 4 points: The team has identified a comprehensive set of security goals for the targeted system. Each security goal is described in detail and the impact of not achieving the security goal is clearly stated. The description includes at least one realistic attack vector for each security goal. The countermeasures for each attack vector are likely to be effective and realistic to deploy.
- 3 points: The reconnaissance phase is generally good, but a key security goal may be missing. Alternatively, one or more security goals is not described in sufficient detail or the attack vectors or countermeasures are not fully developed or are unrealistic. This score reflects solid work with relatively minor deficiencies.
- 2 point: The reconnaissance phase omits two or more key security goals. Alternatively, multiple security goals lack detail or attack vectors or countermeasures are poorly formulated or unrealistic. This score reflects work with moderate deficiencies.
- 0 points or 1 point: The reconnaissance phase has significant limitations. The work does not demonstrate a comprehensive review of the problem or the analysis is severely flawed.

Infrastructure Building: 5 points

Rubric:

- 5 points: The infrastructure is realistic and appropriate for the scenario evaluated. The details of the design are obvious with clear configuration files and step-by-step instruction on how the team built the infrastructure. An independent party would clearly be able to follow these instructions to quickly replicate the experimental setup. There is clear evidence that the team learned how to use the provided infrastructure and gave serious consideration into the best configuration that would enable a clear demonstration of the attack and its defense.
- 4 points: The infrastructure setup is generally good, but some details may be missing or some steps in the instructions may have minor errors. With some problem-solving efforts and additional time, an independent party would likely be able to replicate a close approximation of the experimental setup. The team used the provided infrastructure and designed a configuration that adequately enables a demonstration of an attack and defense.

- 3 points: The infrastructure setup has flaws or significant omissions in its instructions or design documentation. An independent party would have trouble recreating the experimental setup. The team used the provided infrastructure, but the configuration limits the effectiveness of an attack or defense demonstration.

- 2 points: The infrastructure setup is flawed and/or is inadequately described to allow replication. The team deviated from the provided infrastructure or the application of the infrastructure has the potential to undermine the validity of the attack or defense experiments.

- 0 points or 1 point: The infrastructure setup has severe flaws that greatly undermine its utility.

Attack: 5 points

Rubric:

- 5 points: The team identifies a realistic, high-impact attack vector. The attack is implemented flawlessly and the description of the attack and its supporting documentation allows an independent party to replicate the attack. The documentation provides evidence that conclusively shows that the team mounted the attack and that the attack was successful. To earn this score, the attack and/or the implementation of the attack must demonstrate significant independent learning or creative thinking by the team. The attack must be sufficiently complex that the students needed to invent or configure existing tools to effectively launch it.

- 4 points: The team identifies a realistic attack vector. The attack is implemented well and the supporting documentation is solid, but may be missing minor details that are needed to replicate the attack. The supporting documentation generally shows the attack was implemented and was likely successful. The attack showed that the students learned about a new way to launch an attack, but the team may have used an obvious or straightforward methodology to launch the attack.

- 3 points: The team identifies an attack vector, but it may be low impact or may be less realistic. The attack implementation or its supporting documentation may have small flaws. The supporting evidence suggests the attack was implemented and may have been effective, but the support is not conclusive. The attack itself may not have been sophisticated or provided significant learning opportunities for students.

- 2 points: The students identified an attack vector, but it may be low impact or unrealistic. The supporting documentation or evidence of the attack is lacking. The attack itself is trivial and there is little evidence the students learned much from the exercise.

- 0 points or 1 point: The attack has severe flaws that greatly undermine its utility.

Defense: 5 points

Rubric:

- 5 points: The team identifies a realistic, high-impact defense. The defense is implemented flawlessly and the description of the defense and its supporting documentation allows an independent party to replicate the defense. The documentation provides evidence that conclusively shows that the team mounted an effective attack and that the defense prevented the attack from being successful. To earn this score, the defensive approach and/or the implementation of the defense must demonstrate significant independent learning or creative thinking by the team. The defense must be sufficiently complex that the students needed to invent or configure existing tools to effectively deploy it.

- 4 points: The team identifies a realistic defensive strategy. The defense is implemented well and the supporting documentation is solid, but may be missing minor details that are needed to replicate the defense. The supporting documentation generally shows the defense was implemented and was likely successful. The defense showed that the students learned about a new way to protect a system, but the team may have used an obvious or straightforward methodology to deploy the defense.

- 3 points: The team identifies a defensive strategy, but it may be low impact or may be less realistic. The defense implementation or its supporting documentation may have small flaws. The supporting evidence suggests the defense was implemented and may have been effective, but the support is not conclusive. The defense itself may not have been sophisticated or provided significant learning opportunities for students.
- 2 points: The students identified a defensive strategy, but it may be low impact or unrealistic. The supporting documentation or evidence of the defense is lacking. The defense itself is trivial and there is little evidence the students learned much from the exercise.
- 0 points or 1 point: The defense has severe flaws that greatly undermine its utility.

Mission Debriefing Document: 6 points

The mission debriefing document includes the write-up associated with the reconnaissance, attack, and defense. The primary grading of those sections are described above. The mission debriefing document includes overall communication strategy, organization of the information, and interpretation of the data. It also includes supporting appendices and related files.

Rubric:

- 6 points: The report has an introduction and a conclusion section that clearly explain the scenario being evaluated, its importance, the key components that require evaluation, and the results of the specific attack and defense the team evaluated. The report makes use of figures or tables where appropriate to succinctly inform the reader of results. Key supporting information is placed in an easy to read appendix to the report. Additional files are clearly labeled and easy to understand. A README file is provided describing each of the supporting files. All tools are described and the mechanism to obtain the tools (e.g., URLs, relevant repository names) are provided. Where necessary, appropriate academic or technical sources are referenced and cited. The document is compelling, accurate, and leaves little doubt about the attack, defense, and recommendations.
- 5 points: The report has an adequate introduction and conclusion that describe the scenario, its importance, key components, and results. The report is understandable and makes the appropriate points, but the presentation may be suboptimal. The appendix contains supporting information but may have only basic explanation or description of the information. All the supporting files and evidence are included, but it may take extra time to locate. The tools are described, but information on how to obtain them may be missing in some cases. Some academic and technical sources are cited, but there may be instances in which additional citations would be useful. The document is accurate and descriptive, but the document may not be as convincing due to writing limitations or insufficient evidence.
- 4 point: The report has some flaws or omits some of the important points described above. The report may be unclear or unconvincing in places and supporting evidence may be missing, hard to find, or inconclusive. The report is generally accurate, but with apparent flaws or omissions.
- 0-3 points: The report has deficiencies that hinder understanding, omits important data, or provides inadequate support. One of the report sections may be missing or incomplete.