Cybersecurity: Exploring Core Concepts through Six Scenarios

Alan T. Sherman,[1] David DeLatte,[1] Michael Neary,[1]
Linda Oliva,[2] Dhananjay Phatak,[1] Travis Scheponik,[1]
University of Maryland, Baltimore County (UMBC)
Baltimore, Maryland 21250
email: {sherman, dad, mneary1, oliva, phatak, tschep1}@umbc.edu

Geoffrey L. Herman,[3] Julia Thompson,[3]
University of Illinois at Urbana-Champaign
Champaign, Illinois 61820
email: {glherman, jdthomp}@illinois.edu

November 6, 2016

Abstract

We introduce and explain core concepts of cybersecurity through six engaging practical scenarios. Presented as case studies, the scenarios illustrate how experts may reason through security challenges managing trust and information in the adversarial cyber world. The concepts revolve around adversarial thinking, including understanding the adversary; defining security goals; identifying targets, vulnerabilities, threats, and risks; and devising defenses. They also include dealing with confidentiality, integrity, availability (known as the "CIA triad"), authentication, key management, physical security, and social engineering. We hope that these scenarios will inspire students to explore this vital area more deeply.

Our target audience is anyone who is interested in learning about cybersecurity. We assume little to no background in cybersecurity. This paper will also interest those who teach cybersecurity and are seeking examples and structures for explaining its concepts. For students and educators, we include selected misconceptions we observed in student responses to scenarios.

The scenarios comprise responding to an email about lost luggage containing specifications of a new product, delivering packages by drones, explaining a suspicious database input error, designing a corporate network that separates public and private segments, verifying compliance with the Nuclear Test Ban Treaty, and exfiltrating a USB stick from a top-secret government facility.

Index terms— Cryptography, computer security, cryptology, Cybersecurity Assessment Tools (CATS), cybersecurity education, information assurance.

[1]Cyber Defense Lab, Department of Computer Science and Electrical Engineering
[2]Department of Education
[3]Illinois Foundry for Innovation in Engineering Education

1 Introduction

Securing cyberspace is a vital challenge to business activities, our economy, safety of critical infrastructure, government, individual privacy, and our national security. Criminals, business competitors, nation states, terrorists, political activists, and other malicious and non-malicious adversaries threaten to steal money and resources, manipulate election outcomes, disrupt business operations, destroy property and lives, and undermine military effectiveness. To address these threats, Frost and Sullivan [FS13] project a strong and growing need for trained cybersecurity professionals. This paper aims to increase awareness about cybersecurity, motivate individuals to pursue career opportunities in cybersecurity, and provide effective educational materials for cybersecurity.

We introduce and explore core concepts of cybersecurity through six realistic scenarios, accessible to readers with little to no background in cybersecurity. Our primary goal is to create a useful learning resource that will help anyone understand cybersecurity in an effective and engaging way.

We uncover important, cross-cutting concepts through a series of case studies. These scenarios show how cybersecurity professionals identify their adversaries, detect potential vulnerabilities in computer systems, and devise mitigations that can stop adversaries from exploiting those vulnerabilities.

The paper highlights six scenarios, each beginning with a concise prompt. Each scenario motivates a rich discussion of important, difficult, and timeless cybersecurity concepts. The scenarios raise important issues dealing with:

1. determining whether to trust the purported sender of an email, and deciding how to send information securely over the Internet,

2. analyzing the security of package delivery by drones,

3. validating inputs to mitigate the risk of injection attacks,

4. controlling the flow of information across network boundaries, and safely handling potentially dangerous digital objects,

5. designing a system that applies public-key cryptography to provide authentication without secrecy, and

6. devising attacks involving physical security and social engineering.

Designers and defenders of computer systems must protect against both malicious and non-malicious, intentional and unintentional threats. To this end, it is necessary to think adversarially, a mindset we hope to encourage through this paper.

Adversarial thinking involves reasoning about actions and goals in a context in which there might be bad actors attempting to defeat those goals and carry out their own nefarious actions. Such reasoning requires an understanding of the goal requirements, as well as an understanding of who are the bad actors and what are their objectives, resources, access, capabilities, knowledge, motivations, and risk tolerance. It also requires a technical understanding of the computer systems and their potential vulnerabilities. Adversarial thinking, and the associated management of trust and information in computer systems and networks, is the core of cybersecurity [PDH+16].

Systems that are resilient against intentional malicious actors will also be safer against more benign threats and error conditions. Adversarial thinking is an essential skill for everyone involved with information technology.

Section 4.1 provides a short glossary of selected key terms and concepts for readers who feel the need for supplementary explanations. The reader desiring more background in computer science, cryptography, computer security, and cybersecurity may wish to consult Schneier [Sch96], Bishop [Bis03], Kim and Solomon [KS14], and Wikipedia [Wikf]; Section 4.2 suggests additional resources.

1.1 Note to Educators

We developed the scenarios to elucidate the core concepts of cybersecurity that we identified through two Delphi processes that we carried out in fall 2014 [PDH+16]. Identifying core concepts of cybersecurity is an important step in determining what should be taught and in developing effective strategies for teaching and learning cybersecurity. Section 3.1 explains how we generated the scenarios.

Our primary target readers are students in any first course in cybersecurity, regardless of discipline. This paper may also interest instructors and professionals because the scenarios raise imperfectly solved challenges.

This paper is part of a larger project, educational Cybersecurity Assessment Tools (CATS),[4] which is developing two machine-gradable tests. The first assesses how well students in any first course in cybersecurity understand cybersecurity concepts; the second assesses how well a college curriculum prepared graduates entering a career in cybersecurity. These assessment tools will contribute infrastructure for a rigorous evidence-based improvement of cybersecurity education.[5]

In the first year of the project, we conducted two Delphi processes to identify core concepts of cybersecurity. In the second year, we interviewed twenty-six students to understand how students reason about these concepts (for a preliminary report on these interviews, see [SSD+16]). The six scenarios in this paper are drawn from the twelve prompts we developed for these interviews. Section 3.2 highlights some of the misconceptions and problematic reasoning we encountered during these interviews; at the end of each scenario, we also provide examples of a few notable misconceptions.

2 Six Scenarios

The following six scenarios explore and elucidate core concepts of cybersecurity through concrete challenges. We present each scenario with a prompt, brief initial remarks, a detailed response, notable examples of misconceptions we observed, and some notes to the engineering literature. Although each prompt is concise, it invites a rich, broad, and complex discussion, which can reveal a wide variety of levels of understanding of cybersecurity concepts. We encourage the reader to pause and reflect deeply on each prompt before continuing to read our response.

These scenarios involve (1) responding to an email about lost luggage containing specifications of a new product, (2) delivering packages by drones, (3) explaining a suspicious database input error, (4) designing a corporate network that separates public and private segments, (5) verifying compliance with the Nuclear Test Ban Treaty, and (6) exfiltrating a USB stick from a top-secret government facility.

Our responses should be considered exemplary but not definitive. There is no single "right" answer to any of these complex cybersecurity challenges. Consequently, there is value in iterating over the analysis process multiple times.

2.1 Lost Luggage

Bob's manager Alice is traveling abroad to give a sales presentation. Bob receives an email with the following message: "Bob, I just arrived and the airline lost my luggage. Would you please send me the technical specifications for our new product? Thanks, Alice." What should Bob do?

2.1.1 Preliminary Remarks

This prompt involves several practical aspects of cybersecurity, particularly authentication, but also integrity and confidentiality. The deliberately unspecified adversarial model motivates us to explore the relationship between adversary capabilities and security practices. We encourage the reader to pause to imagine a type of adversary and to offer a solution; then continue with the response below.

[4] `http://www.cisa.umbc.edu/cats/index.html`

[5] Schneider [Sch13] articulates the need for more thought in cybersecurity on what should be taught and how to teach it.

2.1.2 Response

Reading the prompt with an adversarial mindset, some questions arise immediately: Is the communication really from Bob's boss Alice? How can we verify the authenticity of the communication and its sender? What steps need to be taken in advance to support authentication? What adversaries might be interested in obtaining the presentation and what does good judgement and common sense suggest about their capabilities? If Alice's identity is verified, what techniques enable a timely resolution of the problem? How crucial is the sales presentation to the success of the company? Is this a routine presentation given many times before, or is it a new cutting edge product that disrupts the business model of the competition?

Concerns, Adversarial Model, Policy. The questions above highlight some of the many concerns that arise when considering the security of communications involving intellectual property. Preparation is critical, and included in that preparation is defining the adversarial model: What will be protected? What are the adversary's motivations and goals? What are the adversary's capabilities? What do we trust?

To begin, we assume that the presentation is very sensitive proprietary information that could cost a significant percentage of the company's profits if it were released to a competitor. Further, we assume that some competitors would be willing to take risks to acquire the information, such as using deception, hacking, or social engineering.

Impersonating Alice by setting up a new email account is straightforward. Discovering that Alice is traveling might be as simple as an overheard conversation or a post in social media. A company policy and training program that discourage sharing information about business travel could improve security, but we will assume that our adversary is able to learn that Alice is traveling and her destination. With an adversarial mindset, we must have some healthy paranoia and assume that the adversary may know details gathered from a variety of sources.

Bob's dilemma begins with tension between the (apparent) requirement to support his boss with the requirement that company information must be protected from the competition.

Policy is also important in shaping Bob's reaction. It can raise Bob's level of security awareness. Is this sort of problem unexpected due to carefully designed plans for handling company assets? Why was the sensitive material placed in a potentially vulnerable location? Clear guidelines (e.g., "the USB stick must be carried on your person") reduce risk, but perhaps Alice encountered unforeseen circumstances such as being required to check a bag at the airline gate. Security measures must be sufficiently robust to adapt to unexpected events.

Useable security must also be a goal. Both Alice and Bob need training in cybersecurity to perform their duties effectively, but it is not reasonable to expect either of them to be a cybersecurity expert. Practical cybersecurity includes the development by experts of solutions that automatically determine the authenticity of a communication and provide appropriate mechanisms for confidentiality and integrity during the exchange of sensitive information.

Setting Up a Foundation for Secure Communications. Bob's dilemma might be solved, or even have been prevented, if the company had established a secure corporate email system using standard tools of cryptography.

Digital signatures enable *authentication* of Alice as sender of the message. *Encryption* protects the *confidentiality* (but not necessarily integrity) of the sales presentation that Bob would send to Alice. *Hash functions* support *integrity* of transmissions, facilitating the detection of any message modification. *Message Authentication Codes (MACs)* provide authentication and integrity. All require advance preparation (establishing keys and policies) and systems that efficiently provide the necessary cryptographic support in a way that is transparent to the users. Although Bob may not completely understand the technical solution enabling security, his training should include a clear idea of the adversarial model so that simply avoiding the security that is in place by using an ad hoc communication channel is discouraged, difficult, or impossible.

If the company used a secure email system, Bob could check if the email came from Alice's company email server and included a valid digital signature. The National Institute of Standards (NIST) provides specifications for a digital signature based on one of several possible cryptographic primitives, for example RSA-PSS, the Digital Signature Algorithm (DSA), or a variant of DSA relying on elliptic curves, ECDSA. In each of these systems, Alice would need to have a pair of keys: one for signing, and one for verifying signatures. NIST also defines a standard for the SHA-3 hash function [Dwo15].[6]

Message *integrity*, ensuring that Alice's message arrived without modification, can be provided as part of the digital signature process, which provides *authentication.* A cryptographic hash function creates a digest (fixed-length tag) that is generated from Alice's message. The cryptographic aspect of "cryptographic hash" connotes that it is not possible to modify the message and produce the same tag, nor to find any two different messages that produce the same tag. Any modification to the message would result in a detectable change to the hash tag.

If Alice additionally wanted *confidentiality*, she would encrypt the message payload. The situation is symmetrical, and Alice should verify that the response to her email came from Bob.

Using a secure corporate email system, Alice and Bob could communicate with mutual authentication, confidentiality, and integrity. That is, Bob has assurance that he is communicating with Alice (and vice-versa); eavesdroppers cannot read the plaintext messages; and Alice and Bob have assurance that the messages have not been modified.[7] With this setup, Bob can send the technical specifications with a high degree of assurance.

This discussion focuses on thwarting imposters and eavesdroppers. Securing the sending and receiving devices is another important consideration, both to protect the unencrypted product specifications and the secret keys needed for security in transit. For example, it is important to guard against possible malware that might compromise these devices.

[6]Secure Hash Algorithm 3 (SHA-3).

[7]Relatedly, the commonly used SSL and TLS protocols [Wikd] establish secure communication sessions with authentication, confidentiality, and integrity.

Some Cryptographic Details.[8] We now briefly summarize some of the mathematical cryptographic details about how to sign, verify, and encrypt messages with the RSA public-key cryptosystem,[9] as shown in Figure 1. For simplicity we omit many details; see [Bar16]. Other encryption strategies, notably using symmetry cryptography (e.g., AES),[10] are also available [NIS01], [DR02].

Each user of RSA is assigned a pair of keys. Each private key is a randomly generated bit string long enough that guessing is infeasible. Each public key is made available to communicants.

For the RSA public-key cryptosystem, NIST recommends that public keys (specifically the integer modulus $n = pq$) be at least 2048 bits long, since the security of RSA depends in part on the adversary's inability to factor the modulus n to find the primes p and q.

Since the sender, Alice, uses her private key to sign, a message with a valid signature implies authenticity of the sender (provided the private key has not been compromised).

Alice's signature σ of a message x could be implemented with RSA as $\sigma = Enc(s_A, h(x))$, where s_A is Alice's secret key, $h(\cdot)$ is a cryptographic hash function, and $Enc(\cdot, \cdot)$ is RSA encryption; specifically, $Enc(k, x)$ denotes RSA encryption of the message x using key k. Here the hash function serves the additional benefit of compressing a long message, making it more efficient to sign while enabling every bit of the message to affect the signature. If the message x is short, h is not required.

Suppose Bob receives $(\hat{\sigma}, \hat{x})$, which might, due to transmission error or deliberate tampering, differ from the signature-message pair (σ, x) sent by Alice. To verify a signature-message pair, Bob computes the verification algorithm $V(\hat{\sigma}, \hat{x})$, which returns true or false. The design of RSA and the key pair (s_A, p_A) make it possible to sign with the secret key and verify with the public key. For RSA, $V(\hat{\sigma}, \hat{x})$ checks if $Enc(p_A, \hat{\sigma}) = h(\hat{x})$, where p_A is Alice's public key.

[8]This optional section may be skipped by the less mathematically-interested reader.

[9]RSA stands for its inventors Rivest-Shamir-Adleman [RSA78].

[10]NIST Advanced Encryption Standard (AES).

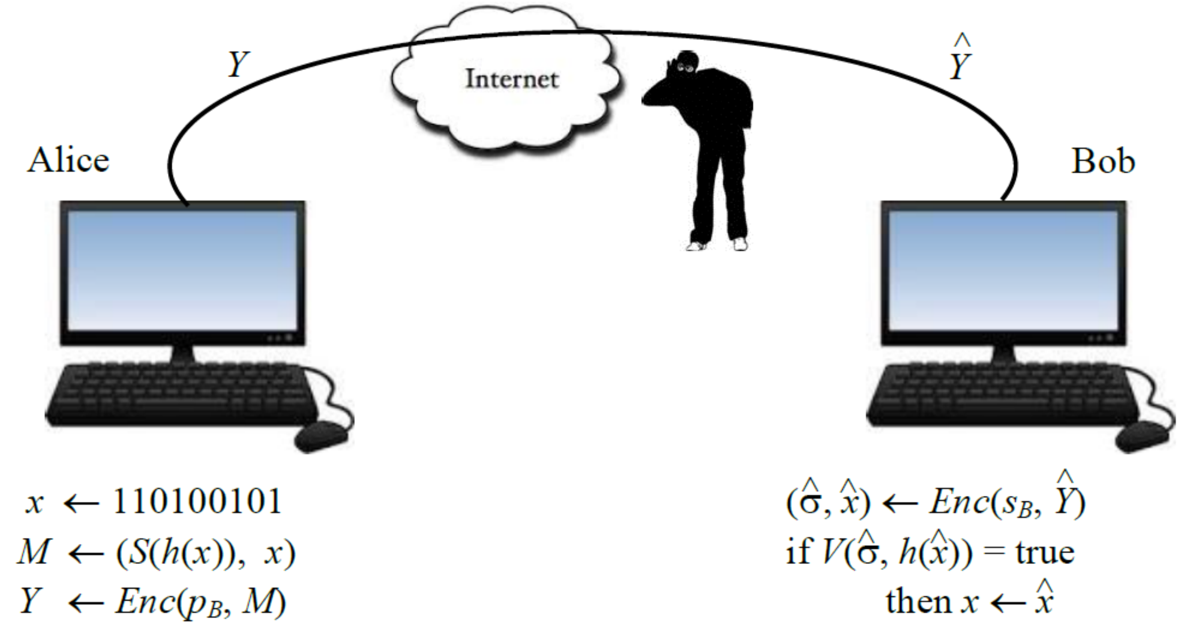
Y
Internet
$\hat{Y}$
Alice
Bob
$x \leftarrow 110100101$
$M \leftarrow (S(h(x)),\ x)$
$Y \leftarrow Enc(p_B, M)$
$(\hat{\sigma}, \hat{x}) \leftarrow Enc(s_B, \hat{Y})$
if $V(\hat{\sigma}, h(\hat{x}))$ = true
then $x \leftarrow \hat{x}$

Figure 1: Alice uses the RSA cryptosystem to send a plaintext message x to Bob with authentication, integrity, and confidentiality. First, using her signature algorithm S, she signs the hash of her message x with her secret key s_A to produce the signature $S(h(x))$. Second, using RSA encryption Enc with Bob's public key p_B, she encrypts the signature-message pair $(S(h(x)), x)$ to produce the ciphertext Y. Upon receipt of ciphertext $\hat{Y}$, Bob first deciphers $\hat{Y}$ with his secret key s_B to produce the signature-message pair $(\hat{\sigma}, \hat{x})$. Second, Bob verifies the signature with the verification algorithm V, which depends on Alice's public key p_A. If the verification succeeds, Bob has assurance that $\hat{x}$ came from Alice and is the unmodified plaintext message x. The adversary cannot read the plaintext message because the adversary does not know Bob's secret key; the adversary cannot forge Alice's signature because the adversary does not know Alice's secret key.

Because the adversary does not know Alice's secret signing key, the adversary cannot forge the signature σ' of any new message x'. Furthermore, any modification to the transmitted message x, say to $\hat{x}$, would result in a signature verification failure because $h(x) \neq h(\hat{x})$ with overwhelming probability.

If Alice additionally wanted confidentiality, she would encrypt the signature-message pair $M = (\sigma, x)$. For example, if Alice and Bob shared a key k for a symmetric cipher such as AES, Alice could encrypt the payload M by computing the ciphertext $\mathrm{AES}_k(M)$. Bob would decipher the ciphertext using k to produce M.

If instead they protected confidentiality with an asymmetric cipher (also called a public-key cryptosystem) such as RSA, Alice would need to know Bob's public key p_B that corresponds to his private key s_B (which only Bob knows). Alice would encrypt the payload with Bob's public key p_B, whereupon Bob would decipher the ciphertext with his private key s_B. Only Bob can decipher the ciphertext because only Bob knows the secret key s_B.

Options Without Secure Corporate Email. Now we consider what Bob should do if the company had not set up a secure email system. The company should have an established policy that guides Bob through this situation, and the company should have educated Alice and Bob about this policy.

If the company had no such policy, then Bob might first try to verify that the email actually came from Alice. One strategy would be to call Alice on her cell phone. Bob could listen and decide if the voice sounded like Alice's. He could also ask questions for which it is likely that only Alice would know the answers (e.g., What did you eat for lunch with me on Tuesday?) So using cellular telephony as a second channel of communication can increase Bob's assurance that he is communicating with Alice.

Bob might also consider the unlikely possibility that Alice may be acting under duress—for example, perhaps a criminal is threatening her with a gun. One technique that can be useful in such situations is a "duress code," a pre-arranged communication through which Alice could signal Bob that Alice is under duress without alerting the coercer. For example, Alice could mention a fictitious co-worker "Jerry," or while entering a PIN permute the last two digits.

Having established that he is indeed communicating with Alice, Bob could then discuss how to send the specifications. The specifications should not simply be sent as plaintext, which would expose them to eavesdroppers. Three possible options might include: (1) Use encrypted email, such as PGP.[11] If Alice and Bob had not already exchanged an encryption key, they could establish one over cellular communications, possibly referring indirectly to several separate pieces of common knowledge (e.g., the color of my office chair). (2) Use a secure cloud-based file-sharing application, such as Dropbox. Both options (1) and (2) require application software. If Alice and Bob do not already have such software, they could possibly download it. (3) Send the specifications by a trusted courier, such as FedEx.

It is essential that Bob recognize the potential vulnerabilities inherent in this scenario. Bob might try first to contact a company official or security officer to ask for guidance. Regardless, he should report the incident.

As the following example illustates, failure to authenticate communications can result in major loss.

Example: Wells Fargo Scam. In 2012, a criminal stole \$2.1 million from a hospital chain's Wells Fargo Bank escrow account by faxing a forged money transfer, inserting a signature of the authorized person copied from the Internet [Zor12]. Failure to authenticate the money transfer properly, including cryptographic binding of the signature to the entire message (e.g., by digitally signing a hash of the message), enabled the crime.

2.1.3 Notable Misconceptions

Some students demonstrated lack of adversarial thinking in suggesting that Bob should simply email the information to Alice. This suggestion reflects lack of awareness of potential threats, such as someone impersonating Alice or eavesdropping on the email. Similarly, others recognized the need to authenticate

[11]Pretty Good Privacy (PGP) [Gar91].

Alice, but still recommended emailing the information without encryption after they authenticated Alice.

2.1.4 Reference Notes

To learn more about cryptography, see Schneier [Sch96] and Stinson [Sti06]. NIST [Bar16] provides guidelines for using public-key cryptosystems. Among such systems, the RSA cryptosystem [RSA78] is widely used, especially for key distribution.

Rescorla [Res01] explains the SSL and TLS protocols, which, among other applications, are widely used by web browsers to provide authentication, confidentiality, and integrity.

For adversarial modeling, see Mateski, et al. [MTV^{+}12] and Bodeau, et al. [BFGG10].

2.2 Delivering Packages by Drones

Consider how a company might deliver packages by drones. As a security engineer for the company, what vulnerabilities, threats, and risks can you identify?

2.2.1 Preliminary Remarks

This timely scenario exposes a rich and wide assortment of cyber-physical issues involving the drone, its controlling infrastructure, and its cargo. These issues include theft, invasion of privacy, control of drone, and the potential use of drones as instruments of crime. Delivery companies and lawmakers need to work out security, privacy, safety, and policy challenges as delivery by drones becomes a reality.

Some may wonder if physical attacks are within the scope of cybersecurity. We take the broad view that any crime involving computers or computer networks is within the domain of cybersecurity; physical security is an important aspect of cybersecurity.

2.2.2 Response

We organize our analysis by considering potential adversaries, vulnerabilities, threats, risks, and mitigations. A *vulnerability* is a weakness that could lead to harm or compromise of a cyber system. A *threat* is a potential action or condition that can cause harm, which might be directed at one or more vulnerabilities. *Risk* refers to the exposure to loss, which involves the probability of certain unfortunate events happening and the consequences if they do happen.

Adversaries. A security engineer must first identify potential adversaries (benign and malicious), their motivations, goals, capabilities, resources, knowledge, access, and risk tolerance. Benign actors include other drones that might cross the flight path. Malicious adversaries might include criminals who wish to steal the drone and/or its cargo, terrorists who wish to use the drones as instruments of crime, thieves who wish to steal information on the drone and/or its associated computer systems, disgruntled employees who wish to hurt the company, business competitors who wish to gain a competitive advantage, or malicious hackers who wish to disrupt delivery operations. We must also ensure that our drones do not drop or misdeliver packages, crash into people, buildings, or objects, and that they do not violate restricted airspace.

Vulnerabilities. Essentially every aspect of the system and its procedures has potential vulnerabilities. These potential vulnerabilities include the drone, its cargo, its onboard computers, the controlling infrastructure, communications, flight path, and all people involved. Some of these vulnerabilities might be exploitable only by a highly capable and motivated adversary; others might be exploitable by less capable adversaries.

Threats. We consider five categories of threats: stealing the drone and/or its cargo, using a drone as an instrument of crime, using a drone to violate someone's privacy, stealing information on the drone and/or its associated computer systems, and attacking the drone communications and infrastructure. Some adversaries may combine threats, such as attacking the drone infrastructure to support other malicious goals. These categories can also overlap and evolve into new threat modalities.

Threat 1: Physical Theft. The most straightforward threat is stealing the cargo and/or the drone. For example, an adversary might try to steal the cargo upon delivery or at the warehouse. As the drone lands, an adversary might physically restrain the drone and take it.

The interactions between the physical and cyber worlds create distinctive challenges in this scenario. Upon capturing a drone, the adversary might attempt to reprogram it and return it to service. Alternatively, after subverting the drone's computer or its controlling infrastructure, the adversary might attempt to cause the drone to misdeliver the cargo to the adversary.

If drones deliver packages from stores to consumers, they will likely also pick up packages from consumers (e.g., returned merchandise). The adversary could try to steal the package at pickup, for example, by subverting the drone or by sending an imposter drone masquerading as the legitimate one.

Conversely, the adversary might intercept the legitimate package (e.g., a cell phone) and substitute an alternative (e.g., the cell phone loaded with malware), either by subverting the drone or by sending an imposter drone.

Threat 2: Using Drone as Instrument of Crime. The potential for an adversary to use a drone as an instrument of crime is particularly troubling. For example, the drone could deliver an explosive, poison, or illegal drugs. Countering malicious drones remains a challenge of significant interest to law enforcement and the military [Rip15, Spa16].

Given that drones are relatively inexpensive and easily available, the adversary could purchase her own drones rather than using ones belonging to the delivery company. Yet, the adversary might find it very appealing to steal or subvert a fleet of drones, in part since delivery drones might not attract the attention of officials.

A particular nasty threat is a "swarm attack," wherein many (perhaps hundreds or thousands) of drones attack a target simultaneously—perhaps a crowded sporting event or other public gathering. Countering a large swarm attack is very difficult.

Criminals may find it very convenient to use drones to deliver contraband (e.g., illegal drugs) to buyers, across boundaries, and into restricted areas.

A hacker might attempt to use drones to broadcast political messages, perhaps by dragging banners.

Threat 3: Using Drone to Violate Privacy. Given that a delivery drone has a special aerial view and permission to enter certain areas for delivery, it is an attractive mechanism through which criminals might take photos and videos, record sounds, and plant spying devices. Intentionally or unintentionally, the drones can also cause air, noise, and visual pollution. The delivery company must ensure that their drones are not modified for this purpose, either by a criminal outsider or by an internal adversary.

Threat 4: Theft of Information. Another threat is theft of information on the drone and on the supporting computer systems. Such information might include customer name, address, item delivered, and billing information, which might be of interest to identity thieves and competing companies.

A related threat to privacy is "traffic analysis," in which the adversary learns meta-information about deliveries without examining the contents of any package. Such information can reveal names and addresses of who is buying from whom, frequency of deliveries, and external package information such as package dimensions, weight, and time of delivery. Traffic analysis can be a powerful tool for criminals and law enforcement.

Threat 5: Attacks on Drone Communications and Infrastructure. Attacking the infrastructure supporting drone delivery is a powerful threat. This infrastructure includes computer systems and databases to manage customers, orders, and deliveries. It also includes computer and radio communication systems to operate, control, and monitor the drones. The computer systems include detailed information about customers and business operations. By subverting the command and/or communication systems, an adversary might be able to gain control of the drones. If radio communications to the drone are not properly protected, an adversary might be able to inject malicious commands to the drone. These systems are a critical target and must be appropriately hardened.

Risks. Without knowing more about the adversary, it is hard to assess the risk. The threats identified above threaten harm to the company (unhappy customers, loss of revenue, damaged reputation, possible legal action against company) and to public safety. All delivery options involve some risk, so one must also balance the relative risks and costs of the options. For example, the risk of loss from a compromised infrastructure is likely similar whether packages are delivered by drones or trucks (especially when self-driving trucks become more common). Nevertheless, the identified risks are real and serious and need to be addressed appropriately.

The most serious risks involving drones may have more to do with their criminal and terrorist use rather than with legitimate companies delivering packages by drones. For example, it seems highly likely that criminals will deliver contraband by drones, and that terrorists will launch violent attacks by drones—as recently happened in northern Iraq [GN16].

Mitigations. Although the prompt did not ask for mitigation strategies, we offer a few suggestions to the challenging engineering task of mitigating the threats identified above.

To safeguard the supporting infrastructure, standard cybersecurity techniques apply, including computer and network security, database security, cryptography, physical security, operations security, policy and people.

Communications between each drone and the supporting infrastructure need to be protected with standard techniques for confidentiality, integrity, availability, and authentication. In particular, messages need to be encrypted, authenticated, and protected for integrity.

When delivering a package, the drone should also leave and send some evidence of its authenticity, for example using a digital signature.

Throughout operations, the base station should maintain communications with the drone and attempt to verify that the drone is operating in a proper state. For example, the base station could send challenges and verify the responses, which can depend in part on cryptographic signatures issued by a trusted piece of hardware on the drone (e.g., one easily available albeit imperfect option is to use a Trusted Platform Module (TPM) [Pea03]). Alternatively or additionally, the response could involve cryptographically signed hashed parameters such as a unique identification number, a modified nonce (use once random number) from the challenge, current time, and the drone's location and/or current camera image.

Reliably verifying control is an extremely difficult, if not impossible task. The drone's computer should be on a tamper-resistant and tamper-responding chip that includes a "failsafe" mode into which the drone can enter if it detects abnormal conditions. This failsafe state might be to land safely and shut down.

The drone should not have any unnecessary information, such as billing information, which could be separately communicated by some trusted channel, such as (encrypted) email or text message. The drone needs to know the delivery address, though it does not necessarily need to know anything else, including the name of the recipient. The association of address with name can be hidden by use of pseudonyms. Information on the drone and supporting infrastructure should be protected with standard cryptographic techniques, though since the information must be used, there will remain the risk of exposure.

Furthermore, as is true for most commercial transactions, companies do not need to, and should not, collect and store the extent of information that they typically collect. For example, a company needs assurance that it will be paid, but it does not typically need to know the name of the buyer. Similarly, there is no need for companies to store traditional credit card numbers. It is safer not to store unneeded sensitive information than to rely on secure technologies and procedures to protect such information (see Chaum [Cha92]).

2.2.3 Notable Misconceptions

Many students revealed misconceptions about how communications might take place between the drone and its command center. For example, one student believed that to carry out attacks, the adversary would have to gain control of the command center. This student failed to recognize other points of potential vulnerability, including the communications and the drone.

Several students misused the words "risk," "threat," and "vulnerability," reflecting a web of confused thinking.

Some students saw encryption as a panacea to many problems, and not just as a tool to protect the confidentiality of data. For example, one student asserted falsely that encryption would prevent signals from being degraded, perhaps confusing encryption and error-correcting codes. Another claimed incorrectly that encryption would prevent message manipulation (encryption provides confidentiality but not necessarily integrity). Several students focused narrowly on only certain aspects, such as encrypting stored data, but failed to recognize the need to protect control signals sent to the drone.

2.2.4 Reference Notes

Melrose [Mel16] and Villasenor [Vil11] warn about dangers posed by drones. Horowitz [Hor16] discusses how to protect drones against cyber attacks. For studies that include analyses of safety, security, and privacy aspects of drones, see Carr [Car13], Ward [War15], and Maddox [MS15]. For more about the development of delivery drones and their economic aspects, see Abrams [Abr15] and Welch [Wel15].

2.3 Database Input Error

When a user Mike O'Brien registered a new account for an online shopping site, he was required to provide his username, address, first and last name. Immediately after Mike submitted his request, you—as the security engineer—receive a database input error message in the logs. What might you infer from this error message?

2.3.1 Preliminary Remarks

This scenario raises the suspicion for one of the most common software vulnerabilities: failure to sanitize user input properly. Malicious users might attempt to exploit this potential vulnerability to launch an injection attack that tricks the database to execute a privileged command, by crafting a clever malicious input. This scenario raises important issues of input validation and the need to protect against potentially dangerous inputs, both at the client and server.

2.3.2 Response

Potentially, the error might have been triggered by any one or more of a large number of possible conditions. Some of these errors might be unnotable from a security perspective, while others might signal a major potential security vulnerability. Regardless, all errors and unusual operating states hold potential for security weaknesses because it is difficult to design, implement, and operate a system that handles all possible such abnormal conditions properly.

After explaining our assumptions, we explore the significance of the apostrophe in the user's name, discuss the potential for an injection-attack vulnerability, recommend mitigations, give a devastating example of an actual SQL injection attack, and summarize recent efforts at Google to reduce the possibility that its software is vulnerable to injection attacks.

Assumptions. We shall assume that the log files record normal operating events and error conditions. It seems likely that the error logged was caused by something that the user entered. Let's assume that the error was not caused by a straight-forward programming error triggered by any user input (such an error would be less likely to cause interesting security issues), nor by the user failing to follow instructions such as entering all required information (in which case the program should respond with helpful feedback to the user). Furthermore, because the error was a "database input error," we may infer that the error was detected by a database program upon attempting to make an input into a database.

We shall also assume, as is common for many web-based shopping sites, that Mike is shopping using a computer (the Client) connected over a network to a shopping site (the Server). This setup is known as a Client-Server model.

A Suspicious Apostrophe. The most notable aspect of the input data is the apostrophe in Mike's last name. While it is possible that a straight-forward programming mistake simply prevented the system from processing this character, it seems more likely that this character somehow caused the database program to throw an error condition. In some programming languages, the single quote character has a special meaning: it suppresses execution or evaluation of the string that follows.

It is common for web-based applications to feed data received from users into queries of an underlying database. Applications issue database commands to make such queries, which can be used for a variety of purposes, from logging in, to searching a website. Thus, it is possible that the error was caused by the database program interpreting part of the input string immediately following the apostrophe (i.e., "Brien") as a database command. Since "Brien" is not a valid command, the database program would throw an error.

It is true that the single quote character ("'") can be different from the apostrophe character ("’"). We do not know exactly what character Mike typed, nor do we know how the system represented the input characters. Nevertheless, it is plausible that the input reaching the Server was interpreted as a single quote.

A Potential Vulnerability to Injection Attack. It is a cause for significant concern that, somehow, a piece of data from the Client side of a transaction was possibly interpreted by a program on the Server side as a database command. In this scenario, Mike did not intend to cause any harm. What might have happened if a malicious user had instead carefully and devilishly crafted a string following the apostrophe to be a dangerous database command? For example, such a command might modify the contents of the database, output sensitive information stored in the database, or attempt to execute a command in the operating system that controls the database. Such attacks are known as "*injection attacks,*" wherein a user tricks the system into executing a command that the user is not authorized to execute.

Susceptibility to injection attack is one of the most common software vulnerabilities today. A common form of injection attack is "SQL injection," [Wikb] referring to injection attacks involving the Structured Query Language (SQL) programming language commonly used to program relational databases.

We suspect the possibility that a programming error at the Server has created a potential vulnerability for an injection attack, possibly an SQL injection attack.

Mitigations. Several mitigation strategies are possible at the Client and Server sides. We recommend that each mitigation be employed for a defense-in-depth, including sanitizing all inputs at both the Client and Server.

First, no user input should ever be directly forwarded as a parameter for any database command. Instead, the user input should be safely interpreted and converted into "prepared statements," which can be thought of as templates for database commands used to ensure that user input cannot interfere with the enveloping command.

Second, more generally, user inputs should always be carefully validated and sanitized. Failure to validate inputs properly is one of the most common programming errors.[12]

Third, inputs reaching the Server should also be validated and sanitized. It is not sufficient to check only at the Client or only at the Server. Malicious data might originate at the Client or Server, or they might be inserted in the communication between the Client and Server.

Fourth, the database should be configured to reduce the chance of injection attacks succeeding. In particular, in processing data related to user enrollment, the database should limit permissible commands as much as reasonably possible, by disallowing certain commands and by operating at the lowest level of privilege needed.

Vulnerability to injection attack is a serious matter, as the following example demonstrates.

Example: Albert Gonzalez. Circa 2007, Albert Gonzalez and his cronies stole 130 million credit cards using SQL injection attacks against several companies including Heartland Payment Systems [Ver10, Wikb]. In 2009, he was indicted and eventually received a 20-year prison sentence for what was at the time considered to be the biggest case of identity theft in America.

[12] Other common programming errors that can cause security vulnerabilities include integer overflow/underflow and buffer overflow (see Kaza [KTH15].)

Case Study: Google. Extremely concerned about the possibility of injection attacks and related attacks (e.g., cross-site scripting attacks), Google took on the ambitious goal of increasing its assurance that no software written at Google will ever permit any injection attack. Google now insists in the meticulous use of prepared statements to prevent user input from being directly used in database commands.

Furthermore, it enforces this policy through stringent compile-time type-checking, so that each software module can be assured that other separately complied modules also guarantee the use of prepared statements and certain other protections. Provided programmers consistently use proper programming interfaces, the type-checking system can enforce policy across module boundaries, which is extremely useful for large complex programs.

Google's model assumes that its programmers are fallible but not malicious. Programmer education is also part of Google's strategy. In 2015, Kern [Ker15] explained Google's software assurance strategy and reported on its remarkable success at drastically lowering the number of known injection vulnerabilities created by Google software.

2.3.3 Notable Misconceptions

Many students focused narrowly on explanations that dealt with simple programming errors rather than with possible more serious database security issues, such as injection attack. Some students reflected a user-side bias, focusing on the interaction between the user and the client, ignoring activity at the server and database. One student suggested that the defense should be solely at the client side, failing to understand that the server and database also need to be protected, and that the server and/or client might be compromised.

Some students identified potential vulnerabilities such as an imposter registration web page, without explaining how the vulnerabilities might relate to the database input error message. An imposter web page is unlikely to account for this error message.

2.3.4 Reference Notes

Halfond, Viegas, and Orso [HVO06] classify types of SQL injection attacks and discuss methods to detect and mitgate these attacks. Martin, et al. [MBP$^+$11] list and discuss common dangerous software errors; SQL injection tops the list, followed by command injection. The Open Web Application Security Project [OWA16] recommends prudent secure programming practices to mitigate common serious vulnerabilities. Kaza, Taylor, and Hawthorne [KTH15] developed educational modules to help students learn how to program more securely.

2.4 Private Network Design

An enterprise with highly sensitive data needs to be able to retrieve information from the Internet. To support this requirement while protecting its sensitive data, the enterprise partitions its internal computer network into two segments: Public and Private, and isolates Private from the Internet. It must be possible to move data from Public to Private, but no data must ever go from Private to Public. As the security architect, describe a design that meets these requirements.

2.4.1 Preliminary Remarks

This scenario raises difficult issues in controlling the flow of information across segment or network boundaries and the need to handle potentially dangerous files or digital objects with great care. The scenario motivates the use of "one-way data diodes" to restrict the flow of information and "sandboxing" to limit the reach of potentially malicious imported objects. The scenario also exposes limitations of the commonly used mechanisms of firewalls [Wika] and Virtual Private Networks (VPNs) [Wike], and highlights tradeoffs security engineers face balancing security, performance, and ease-of-use.

2.4.2 Response

The security architect must design an enterprise system that prevents sensitive data on the Private segment from being exfiltrated while still enabling the

Public segment to retrieve data from the Internet and forward that data to Private. The task would be much simpler without the requirement for data to flow from Public to Private, when the enterprise could simply strongly isolate Private from all Public and Internet connections. The architect must devise a way to enforce the one-way flow of data from Public to Private.

As is true for all security engineering, the security architect must anchor her system on some foundational trusted elements. One choice is to anchor trust in certain basic physical components, such as a bank vault door and key. As is true for all engineering, she must also consider a variety of tradeoffs in selecting her design. In some cases, these tradeoffs include balancing level of assurance against ease of use. Furthermore, our solution will involve an integration of technologies, policies and procedures, and people.

In the rest of this section, we state our assumptions, identify potential threats, explain three design elements, propose our design, analyze two weak design alternatives, present an example, and discuss our design including its engineering tradeoffs and limitations.

Assumptions. We shall assume that the data the enterprise is trying to protect are highly sensitive. We shall also assume that the enterprise wishes to enforce a strict security policy to limit its risk of exposing these data, yet the enterprise wishes for its employees to remain as productive as possible.

Potential Threats. The security architect must consider a wide range of potential threats including: (1) An adversary exfiltrates sensitive data over a network connection to Private. (2) Malware injected on Private modifies system settings, enabling the exfiltration of sensitive data. (3) A malicious or careless employee exfiltrates sensitive data. (4) An act of nature (e.g., flood) or malicious act causes a critical piece of security infrastructure to fail (e.g., power failure), allowing data exfiltration.

Design Elements. To address these threats, our design will incorporate three important design elements: a "quarantine zone," two one-way data diodes, and sandboxing. We now introduce these elements; the next section provides more details.

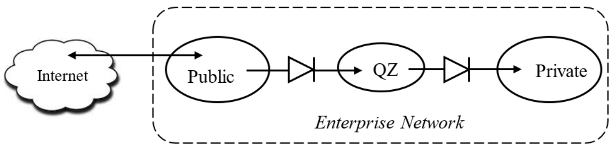

Figure 2: Our design partitions the enterprise network into three parts: Public, Quarantine Zone (QZ), and Private. One-way data diodes prevent data from flowing from Private to QZ, and from QZ to Public. All data from the Internet are considered potentially dangerous; they first pass through QZ where they are safely scanned and inspected using sandboxing before being allowed into Private.

First, it will be helpful to create a third segment of the enterprise network that serves as a *Quarantine Zone (QZ)* between Public and Private, which can be used to inspect any data object that the user is planning to bring from Public into Private. All data objects should be treated with care and suspicion, especially anything originating from the Internet. For example, a document, program, or photograph might contain malware or hidden functionality that could cause harm to the enterprise and its sensitive data. The QZ provides a layer of defense in which untrusted data objects can be safely inspected before bringing them into Private.

A second important element of our design will be a "*one-way data diode*," which is a physical device that permits data to flow in only one direction across a data path [Ste95]. One diode will go from Public to QZ; another will go from QZ to Private. These diodes will prevent data from flowing from Private to QZ, and from QZ to Public.

A third design element will be "*sandboxing*"—a technique for safely inspecting untrusted data objects in the QZ. With sandboxing, one can execute an object in a contained environment in such a way that the object cannot cause any side effects outside of the containment area. This technique is typically carried out using virtualization.

Proposed Design. As sketched in Figure 2, we propose a design that combines the three elements described above: adding the quarantine zone (QZ), enforcing data flows with one-way data diodes, and safely scanning and inspecting all imported data objects in the QZ using sandboxing. The diodes prevent data from flowing from Private to QZ and from QZ to Public. It is also essential to eliminate all other connections into or out of Private.

A worker would use our system as follows. There are three separate workstations disconnected from each other: one to connect to Public, one to connect to QZ, and one to connect to Private. The worker would browse the Internet from the Public workstation. To move a file from Public to Private, the worker would first issue a "push" command from the Public workstation to send the file from Public to QZ. Next, using the QZ workstation, the worker would check the status of the push command and perform all necessary file inspection and virus scanning steps in QZ. If the file is deemed safe, the worker would issue a push command to move the file from QZ to Private. Finally, the worker would use the Private workstation to carry out any desired sensitive tasks on Private.

Virtualization is a convenient technology to support sandboxing because it facilitates containment, enables detailed real-time examination (called "introspection"), and provides a simple way to reset the sandbox after inspection. Thus, instead of running an untrusted program in the sensitive Private segment, in which it might execute system commands and read from and write to important memory locations, the worker first runs the program on an isolated Virtual Machine (VM) in QZ to observe what the program does.

The VM in QZ is set up so that the program running on the VM cannot affect any system outside of the VM; in particular, the program cannot read from or write to any other memory in QZ; the program cannot execute any commands on any operating system outside of the VM; and the program cannot cause any action in Private.

To ensure that no data packets will flow from Private to Public, data paths going from Public to QZ, and from QZ to Private, should deploy the *unidirectional* User Datagram Protocol (UDP) rather than the more convenient *bidirectional* Transmission Control Protocol (TCP) [Wikc].

In addition, the system will employ secure logging using digitally signed write-once media, real-time monitoring, and user education. Among other activities, monitoring will try to detect intrusions and possible network connections. Users will be forbidden from bringing any electronic devices or media (including memory sticks) into the work area, and all computers in the work area will have USB ports and any other removable media ports disabled (e.g., wires cut).

Weak Alternative Designs: Firewall and VPN. Some people might consider basing their designs on a firewall [Wika] or VPN [Wike]. We now discuss these two design alternatives and explain why a firewall works poorly for our purpose and why a VPN fails to solve the problem.

Positioning a *firewall* between Public and Private is an intuitive choice because its purpose is to monitor and regulate the flow of data across a network boundary, as guided by a set of rules. If firewalls worked perfectly, this design might be adequate. Unfortunately, firewalls have significant limitations: they are often misconfigured in practice, and there is a potential vulnerability that an adversary might be able to modify their settings or behavior. For example, documents leaked by Edward Snowden revealed that, unbeknownst to consumers, The National Security Agency (NSA) had infiltrated the source code of Juniper Network's Netscreen firewalls, enabling it to read traffic encrypted on a VPN [Goo16].

A *VPN* is simply the wrong tool for this application. A VPN encrypts traffic to prevent an eavesdropper from reading the traffic; it does not stop the flow of traffic. Moreover, by encrypting the traffic, a VPN makes it more difficult for the enterprise to monitor what data are flowing into and out of its networks. A VPN supports two-way communications.[13] A VPN would not prevent an authorized user to establish a connection between Public and Private and then use that connection to exfiltrate sensitive data from Private to Public. Hence, using a VPN does not meet our assumed design requirement that a strict

[13]It would be an interesting capability useful for this design challenge if it were possible to configure a VPN for one-way only communications.

data-flow security policy must be enforced on all employees.

Example: BlackEnergy Malware. In 2014, Kaspersky Labs [Kas16] identified a piece of malware that infiltrated sensitive networks and attacked Ukranian critical infrastructure, explaining, "The BlackEnergy malware performs DDoS style attacks,[14] cyber espionage and information destruction attacks." Stronger network design and defensive measures could have prevented the spread of this malware.

The U.S. Department of Homeland Security (DHS) [Ley16] recommends using one-way data diodes to prevent this malware from executing on vulnerable networks. By implementing a one-way data diode and requiring vendors and employees to use the same connection paths, the remote exploitation of BlackEnergy can be reduced. DHS elaborates, "During the cyber-attacks, malicious remote operation of the breakers was conducted by multiple external humans using either existing remote administration tools at the operating system level or remote industrial control system (ICS) client software via virtual private network (VPN) connections."

By sandboxing the malware, as it moves from the Internet into a quarantine zone, the code exploits could have been detected and the code would never have made it onto the sensitive network [CC15].

Engineering Tradeoffs. Our design embodies a number of tradeoffs among security, usability, and performance. For example, moving files from Public to Private through QZ, and inspecting them in QZ, causes some delays. Also, requiring the worker to use three separate workstations adds some complexity to the worker's tasks. For situations where it is extremely important not for any sensitive data to move from Private to Public, the enterprise may deem these delays and inconveniences worthwhile tradeoffs. It may also calculate that the costs of our design, including the one-way data diodes and the additional required workstations, may be much less than the costs of the alternative of not attempting to thwart the flow of information from Private to Public.[15]

[14] Distributed Denial of Service (DDOS).

[15] An unsolved challenge of security engineering is the difficulty of estimating costs for actions and inactions.

Our recommended policies may also have some negative impact on worker productivity and morale. Employees working in Private will have to carry out their tasks without connecting to the Internet from their Private workstations. They may find it inconvenient not to be able to use removable media. They may dislike not being allowed to bring a smartphone to work or not being allowed to work remotely from home.

As noted in the design section, it is prudent for data paths from Public to QZ, and from QZ to Private, to use the unidirectional UDP protocol rather than the bidirectional TCP protocol. A consequence of this decision, however, is that UDP is less robust: it cannot handle lost packets nor packets delivered out of order. Additional delays might happen from the resulting need to retransmit files.

Discussion. We conclude by discussing the reasons for our design and pointing out some of its limitations.

We chose our design because one-way data diodes provide a higher level of assurance than would adapting a more complex and less reliable technology, such as firewalls. We prefer a design in which it is physically impossible for data to travel in unauthorized directions, rather than one that depends on workers to follow certain policies and procedures correctly. Although physical devices can sometimes be corrupted, we take some comfort in rooting our trust in part in physical one-way data diodes rather than on the correct operation and configuration of firewalls with complex software.

Nevertheless, our design has some limitations. For example, no inspection can detect all malware (formally, the problem is undecidable). If sophisticated malware could distinguish sandboxing in QZ from execution in Private, then it could behave properly during the sandboxing inspection. It is virtually impossible to stop determined malicious insiders from exfiltrating sensitive data. Careful background checks, periodic security checks, and employee vigilance are tools for mitigating the risk of insider attacks and detecting losses. Unless workstations are physically protected from the employees, there is a risk that a corrupt worker might remove the hard drive. There is no technical barrier to prevent a malicious worker from exfiltrating sensitive data by typing on the Public workstation.

Our design meaningfully raises assurance that malicious outsiders and careless insiders will not move sensitive data from Private to Public.

2.4.3 Notable Misconceptions

All of the students presented with this prompt suggested using firewalls or VPNs; none seemed aware of less-known one-way diode technologies. As we explain above, firewalls are imperfect and easily misconfigured, and VPNs do not block the flow of information. Some students suggested reactionary measures, such as sounding an alarm if an inappropriate memory stick were inserted into a computer. But malware on the memory stick might already become installed by the time the alarm sounded or anyone responded to it.

2.4.4 Reference Notes

For a detailed description of one solution to this design problem, see Moore [Moo00] and Kang and Moskowitz [KM93, KM96], who describe a network security device called the network pump. For more information about data diodes, see Stevens [Ste95] and Ginter [Gin10].

2.5 Nuclear Test Ban Treaty

To comply with the terms of the Nuclear Test Ban Treaty, Country A would like to implant a seismic sensor under Country B's soil to monitor underground weapons testing. Country A fears that B will try to falsify the signals of the sensor, and Country B fears that A will try to exfiltrate spy information embedded in the seismic data. Neither party trusts the other. Requirements of the system include each of the following:

1. *Country A wants assurance that the seismic data it receives came from its sensor and were not modified.*
2. *Country B wants to be able to monitor the signals transmitted from the sensor in real time. It too wants assurance that the signals were not modified.*
3. *The design should be fair to both parties.*

How would you design a system that complies with these requirements? Draw a sketch to illustrate your design.

2.5.1 Preliminary Remarks

Designing such a system is challenging, since encrypting the sensor's output with a single-key cryptosystem does not work. To decipher the encrypted signal, both countries need to know the key, but anyone who knows the key can forge data.

This scenario raises important issues in trust, key management, and authentication without secrecy. The problem highlights a beautiful application of public-key cryptography. It also exposes the importance of physical security, replay attacks, trusted hardware, the challenge of preventing clandestine channels, and the difficulty of dealing with disclosed keys. By the early 1980's at Sandia Labs, Simmons [Sim92] solved this real problem.

2.5.2 Response

The core of this challenge is to provide *authentication without secrecy.* There are several additional subtleties, including the need to protect the sensor physically, the need to prevent replay attacks (where signals are recorded and retransmitted), the desire to prevent hidden (e.g., steganographic) channels (e.g., where Country A tries to hide spy information in other legitimate data or communications), and the consequences if one party maliciously discloses a secret authentication key (thereby casting doubt on the legitimacy of all transmitted data).

Initial Observations. Both countries may be motivated to falsify the seismic signals. Country B may wish to hide unauthorized nuclear tests by fabricating seismic data, and Country A might want to forge incriminating signals.

Also note that, if Country B did not wish to monitor the transmissions, then the problem could be easily solved using standard authentication techniques—for example, using a keyed message authentication code (or even possibly a suitable encryption function), with the secret authentication key known only to the device and Country A. A major difficulty of this problem stems from the requirements that each country must be able to authenticate the signals, yet neither country should be able to forge signals without detection.

Basic Design. As Figure 3 shows, public-key cryptography (e.g., RSA) provides an elegant solution. A (secret) authentication/signing key s_D can be used to encrypt the sensor's signals, which can then be read by anyone who knows the corresponding (public) verification key p_D. In particular, the verification key can be given to both countries and optionally also to certain third parties.

As is standard for signing long messages, the signature is applied not directly to the long message but to a short hash value (called a "tag") of the message computed by a cryptographic hash function, such as SHA-3.

Physical Security. A package comprising a sensor and the cryptographic hardware used to process its signals is inserted into a borehole. The output messages are transmitted for satellite reception. It is important that this package be physically protected. If Country B can tamper with the package, then it might be able to extract the authentication key or modify the functionality of the sensor.

It is possible to protect the package with tamper-responding technology, which technology will erase all sensitive cryptographic variables (including the authentication key) if it detects any physical tampering. In particular, the seismic sensor can be used to detect tampering. The context facilitates this strategy because the package will be underground, physically isolated, and difficult to access in a borehole.

Replay Attacks. To protect against replay attacks (e.g., where Country B records and retransmits previous innocuous signals), each message includes information such as location, date, time, and message number, in an agreed-upon format.

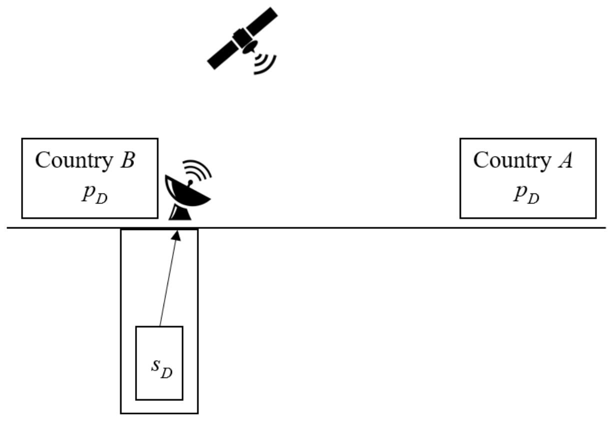

Figure 3: Using public-key cryptography, the underground device encrypts seismic signals with a secret signing key s_D that is generated on the device and never leaves the device. Countries A and B read and verify the authenticity of the signals using the corresponding public verification key p_D output by the device.

Clandestine Channels. It is virtually impossible to provide very strong assurance to B that the sensor is not exfiltrating any unauthorized data via some clandestine (hidden) channel. B can plant its own sensor nearby and compare its output with the transmitted data. All data could be transmitted strictly according to an agreed-upon format. But some spy data might still be hidden, for example, as slight variations in timings of transmitted data or as low-order bits of seismic data.

One possible strategy for trying to eliminate some clandestine channels is sequential "reprocessing" of the data stream within the package in the borehole. Hardware supplied by A formats the seismic data, computes an authentication tag (based on the content payload), and forwards it to hardware supplied by B. Then, B's hardware reformats the message, reclocks the message, and transmits it (including the authentication tag computed by A). Because the bits of the content payload have not changed, the authentication tag computed by A remains valid. While not entirely foolproof, this strategy eliminates many possible clandestine channels, including ones based on timing and message formatting. It increases the difficulty of exfiltrating large amounts of spy data without detection.

Inspection of the hardware and software is another means of providing assurance to B that the sensor is not exfiltrating spy data (see trusting hardware).

Third Parties. An important distinction in the requirements is if County A simply wants to convince itself whether B is cheating, or if a neutral third-party arbiter (e.g., United Nations) is to be convinced. If the goal is to convince a third party, then it is essential that neither country know the authentication key. In particular, since anyone who knows the authentication key can forge signals, if either country knows the key, then the third party could not be certain whether the signals were valid or fabricated by one of the countries.

To prevent either country from learning the authentication key, it can be generated at random on the device and never leave the device. Only the corresponding verification key, and the authenticated data stream, leave the device. Still, there is a risk that the hardware that processes the key might maliciously leak the key.

Trusting Hardware and Software. Agreeing on what hardware and software to use and who should manufacture it is a thorny issue. Malicious hardware or software might include hidden logic that leaked sensitive information including the authentication key. One "cut-and-choose" solution might work as follows: Several copies of the hardware can be made. The one to use could be chosen at random, with the others to be examined by the two countries. Cryptographic checksums of the software can help detect modifications of software, but they do not verify that the software works correctly.

Unilateral Actions. In the real problem solved by Simmons [Sim92], the countries further demanded that unilateral action by any one of the countries (including intentional key disclosure) should not undermine the confidence of the other country (or that of a third party) in the authenticity of the messages. For example, after innocuous seismic data are sent, Country A might try to undermine confidence by disclosing the authentication key and by then claiming that Country B could have forged the data with the compromised key. Conversely, after incriminating seismic data are sent, Country B might disclose the authentication key and then claim that Country A could have forged the data. Even if the hardware generates the keys, the countries feared that possibly malicious hardware might leak the key.

To address these concerns, Simmons recommended using a "concatenated" (not sequential) authentication system design, where the signature is a list of two or three separately computed authentication tags. Each of the countries (and the third party if present) would supply its own authentication hardware under the control of its own separate authentication key, with all parties knowing the corresponding verification keys. The authentication algorithms do not have to be the same. The authentication tag of the concatenated system would be an ordered pair (or triple) of the two (or three) tags computed by each of the parties. Unilateral key disclosure by any one country would not undermine the confidence of the other parties. Collusion by A with B would not be in either country's best interest.

2.5.3 Notable Misconceptions

Several students suggested incorrectly that the device should encrypt its data using a symmetric cipher. However, each party would need to know the key to decrypt the data, and anyone who knows the key could modify or forge the data. Another student suggested storing and transmitting seismic data in three separate channels, each encrypted with a separate symmetric key known by the device and one country. Again, this solution does not prevent the key holder from modifying the seismic data.

2.5.4 Reference Notes

Simmons [Sim83, Sim92] describes his solution based on public-key cryptography.

FIPS140 [FIP01] specifies standards for cryptographic modules, including their physical security. Weingart [Wei00] surveys attacks and defenses for physical security. Michaud [MS11] discusses attacking tamper-proof seals.

For an introduction to steganography, see Cole [Col03].

2.6 USB Stick under Floor Tile

Alice works in a top-secret government facility where she has hidden a USB memory stick, with critical information, under a floor tile in her workspace. Starting from outside the fence of the building, how would you, as a penetration tester, retrieve the USB stick?

2.6.1 Preliminary Remarks

This prompt motivates discussion of a wide range of security issues, from physical security to personnel security and social engineering. This prompt illustrates why security engineers must consider a wide range of potential attacks and countermeasures. It also illustrates how people (including insiders) are often the most vulnerable links in any security system. The prompt underscores the strong need for sound training, policies (including what to do in unusual situations), and technologies to achieve security goals.

2.6.2 Response

We shall consider the following classes of attacks: direct attack, high-technology, and social engineering [MW02]. Since people are often the weakest links in any security system, the most attractive attack will likely be, or at least involve to some degree, social engineering. Because there are many aspects to this challenge that are underspecified, we shall begin with some assumptions. We conclude with some proactive recommendations.

Assumptions. The open-ended nature of the prompt raises many questions. What type of security protects the facility? We shall assume that the facility has formidable security with guards, 24/7 surveillance, fences, locked doors, sensors, alarms, windows that do not open (or no windows), security badges, and all employees have undergone security training and hold top-secret security clearances.

The problem does not specify whether we must physically retrieve the USB stick, or if it would be sufficient to exfiltrate the information on the stick. For some attacks, it might be easier to transfer the data from the stick onto some other medium and exfiltrate the data without removing the physical stick.

The problem does not state if we know anything about the layout of the building, the location of the workspace or floor tile in question. We shall assume we know the exact or approximate location of the target floor tile within the workspace. We shall assume that we are not given any other information about the facility, but our solution will begin by learning as much as possible about the facility through reconnaissance.

The problem does not describe what resources we are permitted to use, how much money we are permitted to spend, by when we must retrieve the stick, what might be the penalty for being caught, how much risk we may assume in carrying out the attack, or the nature and value of the information on the stick. We shall assume that we have considerable time and financial resources to carry out the attack, but that we will aim to avoid detection, minimize risk, and not spend an excessive amount of money.

Direct Attacks. Crudely trying to break in by cutting through the fence and entering through a window or door, or by dropping onto to the roof from a helicopter, is highly likely to be detected.

Overrunning the perimeter and penetrating the building with guns and explosives would run contrary to the goal of avoiding detection, and such an attempt would eventually be met with overwhelming counterforce.

One might try to masquerade as an authorized employee—perhaps a janitor who cleans near the workplace—and enter through the main employee gate. This attack is highly risky and must overcome checks of badges, physical authentication tokens, and possible passcodes and biometric identification. Also, the imposter must not raise the suspicion of coworkers or superiors.

One might find an accomplice who looks like the janitor. By breaking into the janitor's home or car, one might be able to copy the credentials and obtain fingerprints (retinal scans would be more difficult to fake). Obtaining the required passcodes is problematic by direct attack; maybe it is possible to have someone observe the entered codes. The accomplice could try to arrive before the legitimate employee at the time the employee typically arrives. Another accomplice might delay the employee, for example by causing a traffic jam. After obtaining the USB stick, the attacker could hide it in his or her clothing, shoes, or in a body cavity.

Tunneling under the fence and up into the building might have a better chance of avoiding detection, but unless the workspace is in a basement, there would remain the difficulty of how to proceed from the tunnel exit to the workspace. The entrance to the tunnel would have to be far away, and tunneling would be difficult, expensive, and require considerable skill (but the attack is plausible). The tunnel would have to evade possible ground sensors and it would likely have to breach a concrete slab. Eventually, the tunnel would likely be detected.

Of these direct attacks, tunneling is relatively most attractive, but each of these attacks has a low chance of success and a high risk of detection.

High-Technology Attacks. One could try to enter the facility with a sophisticated intelligent autonomous miniature robot, for example disguised as a fly, ant, or cockroach [Szo15]. Autonomous control would alleviate the need for one-way or two-way communications, which would be highly problematic and fairly easily detected. The robotic device could move about by walking (simplest), or by a combination of flying, walking, and possibly swimming. Robotic competitions held by DARPA [DAR] provide snapshots of some of the current capabilities of autonomous robots.

It is likely that such a device could enter the facility without detection, for example through some crack. Any homeowner knows that it is essentially impossible to exclude all insects from a structure. Once in the facility, the device would navigate to the workspace, retrieve the information, and then exit the facility. Navigating within the facility ought to be relatively simple—for example, crawling along pipes through walls and vertical shafts. Detailed floorplans of the building would be useful but not essential. A risk is detection by sensors that scan for power sources.

Once the USB stick is located, there remains the challenge of extracting the information from the stick. The robotic insect might insert electric probes into the USB stick and copy out the stored data. Depending on the characteristics of the USB stick, accessing the USB stick's connections might be easy, or it might require removing a plastic cap or drilling into the USB stick. The insect would then exit the facility with the information.

The strategy of using a robotic insect might be combined with the tunneling strategy: the insect might enter a sewer pipe from some distant access point (simplest) or via a tunnel, and then navigate through the sewer pipe to a toilet or sink drain within the facility. Rats have entered houses through this technique.

This high-tech strategy, properly executed, has a high chance of success, but it would require a very sophisticated autonomous robotic insect.

Social Engineering Attacks. A variety of social-engineering attacks are possible, exploiting a multitude of human weaknesses. In such attacks, one could attempt to bribe, entice, coerce, or trick legitimate employees into carrying out certain actions and/or releasing certain information. One difficulty of such attacks is that the target employee might, in part due their security training, resist and report such attempts. Eventually, employees must undergo polygraph reviews, and for most people it is very difficult to fool such reviews. For these reasons, trickery (without the target realizing what has happened) is more attractive than bribery or extortion.

A simple social engineering attack is to offer a potentially vulnerable employee with access to the workspace a large amount of money to retrieve the USB stick. Similarly, one could threaten to reveal damaging information about the employee or threaten to harm the employee or a loved one unless he or she complies. Sophisticated, skillful prostitutes have tricked many people.

During the initial surveillance phase, one could assemble many separate small bits of information about the facility and its employees to gain an understanding of the workplace and its workers. This surveillance might include observations, conversations with employees, examination of trash, scrutiny of social media, and cyber attacks including of personal electronic devices of employees.

"Piggy-backing" is a crude attempt at entering through the main gate: try to slip in immediately behind someone else. This technique might work better if an accomplice simultaneously created a distraction, such as a medical emergency or a fight. A variation is pretending to be a delivery person, when someone might even open a door for you. Proper procedures and training of guards and employees should stop these crude attempts.

One might try to become employed at the facility. This strategy requires passing a thorough background investigation and polygraph. Also, it may be difficult to become assigned to the area near the workplace.

One might attempt to be invited into the facility as a visitor, for example, to give a guest lecture on a topic of interest to the people in the workspace. While visiting, one might create a distraction, such as a feigned medical emergency, which might include lying on the floor near the target floor tile. One might also infiltrate an ambulance crew to try to bring additional accomplices into the area.

This scenario also illustrates why many government facilities disallow removable media (including USB sticks) into work areas: they might facilitate the exfiltration of sensitive data. Some organizations support this policy with education and physical modification of machines (e.g., cutting wires to external ports). Removable media, despite their convenience, also present a risk for infecting machines with malware.

Proactive Measures. This scenario also illustrates the value of protecting data at rest. If the data on the USB stick were encrypted, then the data would be protected even if an adversary obtained the stick.

Furthermore, if there were no removable media in the workspace, then an adversary would be unable to remove such media. The organization could forbid all removable media in the workspace and modify all computers so that they are incapable of accepting, reading from, or writing to removable media. A cost of such policies is that they tend to interfere with work efficiency. Also, it is essentially impossible to stop a trusted insider from secretly bringing in a small memory device.

2.6.3 Notable Misconceptions

Student responses lacked breadth and useful details of potential solutions. For example, some students focused solely on social engineering attacks or on physical attacks (e.g., tunneling).

Some students proposed policies that created negative consequences. For example, one student suggested that employee credentials be left at the office. This suggestion, however, would create an attractive target for theft of credentials, simplfy the task of an attacker who has entered the office, prevent the employee from being able to authenticate herself while outside the office, and increase the risk of insider attacks by people with access to the office. Others stated that passwords should be complex and changed frequently, but such policies can reduce security by encouraging users to engage in risky adaptive behavior such as writing down passwords.

2.6.4 Reference Notes

For an introduction to social engineering, see Mitnick and Wozniak [MW02].

Among the secrets leaked by CIA mole Robert Hanssen is the existence of a tunnel the United States built under the Soviet embassy in Washington, DC [DCt01]. In 1955-1956, the United States had operated a tunnel crossing from West Berlin into East Berlin, to monitor signals [Ber88].

NIST special publication 800-53 [NIS13] specifies security and privacy controls for federal information systems and organizations.

Distributing flash drives containing malware is a well-known trick to infect computers [Doc12]. The U.S. Department of Defense admitted to being compromised by such an attack [Kno10].

3 Note to Educators

In this section, we explain how we generated the prompts, and we describe some of the ways we observed students misunderstand cybersecurity issues.

3.1 How We Generated the Prompts

To generate prompts, our main starting point was a list of cybersecurity concepts produced from our two Delphi processes [PDH^{+}16], ranked by importance. During these Delphi processes, we asked thirty-six experts to identify cybersecurity concepts that are important, difficult, and timeless.

Our goals in producing prompts included covering a variety of concepts and contexts with varying degrees of difficulty. We generated most of the prompts in brain-storming sessions while seated around a conference table. We sought prompts that would stimulate students to talk about solving concrete cybersecurity problems, thereby revealing their understandings, misconceptions, and problematic reasonings. We tried to produce concise, engaging prompts that exposed important, challenging, practical issues that can be discussed deeply.

Because cybersecurity is about securing computers and computer networks, we set the prompts in cyber contexts (as opposed to non-cyber security contexts, such as protecting physical mail). Still, we aimed for our prompts to be understandable by students in any first course in cybersecurity, building on common life experiences. Team members drew upon their experiences teaching cybersecurity and working in the field.

While our scenarios cover a wide spectrum of important concepts, we do not claim that our coverage is complete.[16] Our scenarios cover many of the top-rated concepts identified in our Delphi processes, and these processes were not intended to produce a complete list of cybersecurity concepts. We invite the reader to construct additional prompts that elicit exploration of important, timeless concepts not explored by our prompts, and we would be happy to hear from anyone who does so.

[16]For example, our scenarios do not explore cryptographic commitment, secret sharing, principle of least privilege, formal methods, code obfuscation, multi-party computations, private information retrieval, zero-knowledge proofs, nor homomorphic encryption.

3.2 Misconceptions and Problematic Reasoning

To illustrate some of the many and varied misconceptions we observed in student responses, at the end of each scenario, we summarize a few notable examples. We plan to explore these misconceptions and problematic reasonings in future work (for some preliminary analysis, see [SSD^{+}16]). Some of the ways we observed students misunderstand cybersecurity concepts include conflating concepts (e.g., encryption vs. hashing, and authentication vs. authorization), biased reasoning, unsound logic, and factual errors. Furthermore, in comparison with the responses we give in Section 2, student responses tended to reflect incomplete and narrowly-focused observations, and they seemed to lack an explicit and sound framework (such as one centered on adversarial thinking) around which to organize their thoughts.

4 Supplemental Explanations and Resources

Cybersecurity is an interdisciplinary field that concerns the management of information and trust in an adversarial cyber world. It integrates people, policies and procedures, and technology. Contexts of interest include any situation that involves computers or information in electronic form, including computer systems, computer networks, databases, and applications.

In this section we briefly explain four essential cybersecurity concepts, including the so-called CIA Triad (confidentiality, integrity, and availability) and authentication.[17] We also point out several introductory textbooks on cybersecurity. We hope this section will be helpful to readers who seek additional explanations of terms and concepts encountered in the case studies.

[17]Parts of Section 4.1 are drawn from our companion paper [SSD^{+}16].

4.1 Glossary of Selected Terms

Four essential concepts include confidentiality, integrity, availability, and authentication. See Section 1 for an explanation of adversarial thinking.

Confidentiality refers to keeping information secret from unauthorized entities. Encryption is a tool for keeping information confidential. An encryption function mixes a plaintext with a secret key in a complicated way to produce ciphertext, with the intention that an eavesdropper seeing only ciphertext cannot decrypt the ciphertext to produce the plaintext without knowledge of the secret key.

Integrity refers to the problem of detecting whether data (either at rest or in transit) have been modified. Cryptographic hash functions are useful tools for achieving integrity. A hash function takes an arbitrarily long input and produces a short fingerprint (also called a tag) such that, if any change is made to the input (even just one bit), then with overwhelming probability the tag will change. Message authentication codes also protect integrity.

Availability refers to systems, services, and networks being up and running.

Authentication refers to the task of, say, Alice convincing Bob that a message purporting to have originated from Alice did indeed come from Alice. Digital signatures and message authentication codes are tools for achieving authentication. For example, Alice can sign a message using her private signature key. Using Alice's public verification key, Bob can verify Alice's signature. A related concept is *non-repudiation*, which refers to the inability of a party to deny having signed a document. By contrast, *authorization* refers to whether an entity is allowed to perform some action, for example, reading some data or gaining access to some computer system.

4.2 Introductory Sources on Cybersecurity

We identify a few resources for learning more about cybersecurity.

Introductory textbooks on cybersecurity include Kim and Solomon [KS14], Smith [Smi16], Shoemaker and Conklin [SC12], and Singer and Friedman [SF14].

Textbooks on computer security include Bishop [Bis03], Pfleeger [PPM15], and Stallings [SB14].

For more about cryptography, see Schneier [Sch96] and Stinson [Sti06]. Anderson [And08] and Furgeson, et al. [FSK10] explain engineering aspects cryptography and security.

Katz, et al. [KL15] offer an accessible introduction to the modern theory of provable security, and Shoup [Sho09] explains number theory underlying many modern cryptographic systems. Bernstein, et al. [BBD09] discuss approaches to cryptography that aim to resist attack by quantum computers.

The NICE Framework [NIC] establishes a common lexicon to define the activities of cybersecurity professionals.

5 Conclusion

We have explored fundamental concepts of cybersecurity through describing and discussing six scenarios. We present cybersecurity concepts through scenarios in part because of our strong belief in the power of learning through case studies. We hope that students find these scenarios helpful and engaging, and that educators can incorporate them into a variety of learning activities.

Abstracting from our responses to the scenarios, a useful structure emerges for reasoning about cybersecurity tasks: define requirements; adopt an adversarial model and state trust assumptions; identify potential vulnerabilities, threats, and risks; devise defenses; evaluate the defenses; and prepare response and recovery plans in case of failures.

These scenarios highlight the importance of adversarial thinking, which composes the essential core of cybersecurity and which connects and transcends all of the many diverse disciplines therein. Effective cybersecurity, however, needs more than abstract adversarial thinking: adversarial thinking must also be integrated with deep expertise on a wide variety of relevant technical subjects, including, for example, computer networks, operating systems, databases, software engineering, hardware, forensics, and behavioral psychology. The world would be a more secure place if everyone—including computer scientists, engineers, policy makers, students and educators—integrated adversarial thinking into their everyday work and thereby meaningfully improved their policies, practices, goods and services.

Acknowledgments

We thank Richard Forno, Enis Golaszewski, and Konstantinos Patsourakos for helpful comments. Thanks to Andy White for suggesting a scenario similar to our USB stick prompt.

This work was supported in part by the U.S. Department of Defense under CAE-R grants H98230-15-1-0294 and H98230-15-1-0273, and by the National Science Foundation under SFS grant 1241576.

References

[Abr15] M. Abrams. Developing Delivery Drones. `https://www.asme.org/engineering-topics/articles/robotics/developing-delivery-drones`, August 2015. [Online; accessed 9-30-16].

[And08] Ross Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems, Wiley.* Wiley, Hoboken, NJ, second edition, 2008.

[Bar16] Elaine Barker. NIST Special Publication 800-175B Guideline for Using Cryptographic Standards in the Federal Government: Cryptographic Mechanisms. `http://dx.doi.org/10.6028/NIST.SP.800-175B`, August 2016. [Online; accessed 8-October-2016].

[BBD09] Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen. *Post-Quantum Cryptography.* Springer, 2009.

[Ber88] Operation REGAL: The Berlin Tunnel. `http://boingboing.net/2012/07/10/dropped-infected-usb-in-the-co.html`, 1988. [Online; accessed 8-October-2016].

[BFGG10] Deb Bodeau, Jenn Fabius-Greene, and Rich Graubart. How Do You Assess Your Organization's Cyber Threat Level? The MITRE Corporation, `https://www.mitre.org/sites/default/files/pdf/10_2914.pdf`, August 2010. [Online; accessed 8-October-2016].

[Bis03] Matt Bishop. *Computer Security: Art and Science.* Addison-Wesley, Boston, MA, 2003.

[Car13] E. B. Carr. Unmanned Aerial Vehicles: Examining the Safety, Security, Privacy and Regulatory Issues of Integration into U.S. Airspace. `http://www.ncpa.org/pdfs/sp-Drones-long-paper.pdf`, 2013. [Online; accessed 9-30-16].

[CC15] National Cybersecurity and Communications Integration Center. Seven Steps to Effectively Defend Industrial Control Systems, 2015. [Online; accessed 18-Oct-2016].

[Cha92] David Chaum. Achieving Electgronic Privacy. *Scientific American*, pages 96–101, August 1992.

[Col03] Eric Cole. *Hiding in Plain Sight: Steganography and the Art of Covert Communication.* John Wiley & Sons, Inc., New York, NY, USA, 1 edition, 2003.

[DAR] DARPA Robotics Challenge. `https://en.wikipedia.org/wiki/DARPA_Robotics_Challenge`. [Online; accessed 11-June-2016].

[DCt01] A Not-So-Secret Tunnel. CBSnews.com, `http://www.cbsnews.com/news/a-not-so-secret-tunnel/`, March 4 2001. [Online; accessed 8-October-2016].

[Doc12] Cory Doctorow. Dropped Infected USB in the Company Parking Lot as a Way of Getting Malware onto the Company Network. `http://boingboing.net/2012/07/10/dropped-infected-usb-in-the-co.html`, July 10 2012. [Online; accessed 8-October-2016].

[DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES—The Advanced Encryption Standard*. Springer, 2002.

[Dwo15] Morris J. Dworkin. NIST Federal Information Processing Standards 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. `https://www.nist.gov/node/555116?pub_id=919061`, August 2015. [Online; accessed 30-October-2016].

[FIP01] Federal Information Processing Standards (FIPS) Publications: FIPS 140–2, Security Requirements for Cryptographic Modules. `http://csrc.nist.gov/publications/PubsFIPS.html#140-2`, May 2001. [Online; accessed 9-October-2016].

[FS13] Frost and Sullivan. The 2013 (ISC)^2 Global Information Security Workforce Study. `https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&cad=rja&uact=8&ved=0CDgQFjAB&url=https%3A%2F%2Fwww.isc2.org%2FGISWSRSA2013%2F&ei=-HgUVMCVC8-GyATo3YCoDg&usg=AFQjCNEF5GJscvZ11lqHcRzTdZb5_gNhsQ&bvm=bv.75097201,d.aWw`, 2013. [Online; accessed 21-June-2016].

[FSK10] Niels Ferguson, Bruce Schneier, and Tadayoshi Kohno. *Cryptography Engineering: Design Principles and Practical Applications*. Wiley, 2010.

[Gar91] Simson Garfinkel. *PGP: Pretty Good Privacy*. O'Reilly & Associates, 1991.

[Gin10] Andrew Ginter. ICS Cyber Convergence, Security Basics: One-way Diodes. `http://id.lockheedmartin.com/blog/security-basics-one-way-diodes`, November 1 2010. [Online; accessed 8-October-2016].

[GN16] Thomas Gibbson-Neff. ISIS Used an Armed Drone to Kill Two Kurdish Fighters and Wound French Troops, Report Says. Washington Post, `https://www.washingtonpost.com/news/checkpoint/wp/2016/10/11/isis-used-an-armed-drone-to-kill-two-kurdish-fighters-and-wound-french-troops-report-says/`, October 11 2016. [Online; accessed 11-October-2016].

[Goo16] Dan Goodin. RISK ASSESSMENT / SECURITY & HACKTIVISM Juniper Drops NSA-Developed Code following New Backdoor Revelations, Jan 2016. [Online; accessed 18-Oct-2016].

[Hor16] Barry M. Horowitz. Cybersecurity for Unmanned Aerial Vehicle Missions. `http://www.afcea.org/content/?q=Article-cybersecurity-unmanned-aerial-vehicle-missions`, April 1 2016. [Online; accessed 9-October-2016].

[HVO06] William G Halfond, Jeremy Viegas, and Alessandro Orso. A Classification of SQL-Injection Attacks and Countermeasures. In *Proceedings of the IEEE International Symposium on Secure Software Engineering*, volume 1, pages 13–15, 2006. [Online; accessed 23-July-2016].

[Kas16] Kaspersky. BlackEnergy APT Attacks in Ukraine. `http://www.kaspersky.com/internet-security-center/threats/blackenergy`, 2016. [Online; accessed 22-July-2016].

[Ker15] Christoph Kern. Preventing Software Bugs through Security Design. Invited talk, 24th USENIX Security Symposium, `https://www.usenix.org/conference/`

usenixsecurity15/symposium-program/presentation/kern, August 2015. [Online; accessed 11-October-2016].

[KL15] Johnathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography.* CRC Press, second edition edition, 2015.

[KM93] Myong H. Kang and Ira S. Moskowitz. A Pump for Rapid, Reliable, Secure Communication. In *CCS '93 Proceedings of the 1st ACM conference on Computer and communications security*, pages 119–129. , 1993.

[KM96] Myong H. Kang and Ira S. Moskowitz. A Pump for Rapid, Reliable, Secure Communication. http://www.dtic.mil/dtic/tr/fulltext/u2/a465065.pdf, 5 1996. [Online; accessed 24-July-2016].

[Kno10] Brian Knowlton. Military Computer Attack Confirmed. *New York Times*, http://www.nytimes.com/2010/08/26/technology/26cyber.html, August 25 2010. [Online; accessed 8-October-2016].

[KS14] David Kim and Michael G. Solomon. *Fundamentals of Information Systems Security.* Jone & Bartlett Learning, Burlinton, MA, second edition, 2014.

[KTH15] Siddarth Kaza, Blair Taylor, and Elizabeth K. Hawthorne. Introducing Secure Coding in CS0, CS1, and CS2: Conference Workshop. *Journal of Computing Sciences in Colleges*, 3:11–12, June 2015.

[Ley16] John Leyden. BlackEnergy Malware Activity Spiked in Runup to Ukraine Power Grid Takedown: But its Role in the Attack Remains Unclear. http://www.theregister.co.uk/2016/03/04/ukraine_blackenergy_confirmation/, March 4 2016. [Online; accessed 22-July-2016].

[MBP$^+$11] Bob Martin, Mason Brown, Alan Paller, Dennis Kirby, and Steve Christey. 2011 CWE/SANS Top 25 Most Dangerous Software Errors. http://cwe.mitre.org/top25/archive/2011/2011_cwe_sans_top25.pdf, 2011. [Online; accessed 23-July-2016].

[Mel16] Jeff Melrose. Drone Attacks on Industrial Wireless A New Front in Cyber Security. https://www.blackhat.com/docs/us-16/materials/us-16-Melrose-Drone-Attacks-On-Industrial-Wireless-A-New-Front-In-Cyber-Security.pdf, 2016. [Online; accessed 7-August-2016].

[Moo00] Andrew P. Moore. Naval Research Laboratory NRL/MR/5540–00-8459, Network Pump (NP) Security Target. handle.dtic.mil/100.2/ADA380262, May 29 2000. [Online; accessed 9-October-2016].

[MS11] Eric Michaud and Jamie Schwettmann. Drone Attacks on Industrial Wireless A New Front in Cyber Security. https://www.scribd.com/document/47334072/How-to-Steal-a-Nuclear-Warhead-Without-Voiding-Your-XBox-Warranty-paper, 2011. [Online; accessed 7-August-2016].

[MS15] S. Maddox and D. Stuckenberg. Drones in the U.S. National Airspace System: A Safety and Security Assessment. *National Security Journal, Harvard Law School*, Feb. 2015. [Online; accessed 9-30-16].

[MTV$^+$12] Mark Mateski, Cassandra M. Trevino, Cynthia K. Veitch, John Michalski, J. Mark Harris, Scott Maruoka, and Jason Frye. Cyber Threat Metrics. Sandia National Laboratories, http://nsarchive.gwu.edu/NSAEBB/NSAEBB424/docs/Cyber-065.pdf, 2012. [Online; accessed 8-October-2016].

[MW02] Kevin Mitnick and Steve Wozniak. *The Art of Deception.* John Wiley & Sons, Indianapolis, Indiana, 2002.

[NIC] NICE Framework. http://csrc.nist.gov/nice/framework/. [Online; accessed 8-October-2016].

[NIS01] Announcing the Advanced Encryption Standard. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf, November 2001.

[NIS13] NIST Special Publication 800-53, Revision 4, Security and Privacy Controls for Federal Information Systems and Organizations. http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf, April 2013. [Online; accessed 9-October-2016].

[OWA16] OWASP Open Web Application Security Project. OWASP Top 10 Proactive Controls 2016: 10 Critical Security Areas that Web Developers Must Be Aware of. https://www.owasp.org/images/5/57/OWASP_Proactive_Controls_2.pdf, 2016. [Online; accessed 23-July-2016].

[PDH$^+$16] Geet Parekh, David DeLatte, Geoffrey L. Herman, Linda Oliva, Dhananjay Phatak, Travis Scheponik, and Alan T. Sherman. Identifying Core Concepts of Cybersecurity: Results of Two Delphi Processes. *IEEE Transactions on Education*, May 2016. submitted.

[Pea03] S. Pearson. *Trusted Computing Platforms: TCPA Technology in Context*. Prentice-Hall, Upper Saddle River, NJ, 2003.

[PPM15] Charles P. Pfleeger, Shari L. Pfleeger, and Jonathan Margulies. *Security in Computing*. Prentice Hall, Westford, MA, fifth edition, 2015.

[Res01] Eric Rescorla. *SSL and TLS: Designing and Building Secure Systems*. Addison-Wesley, 2001.

[Rip15] Amanda Ripley. Playing Defense against the Drones: We've Managed to Create Armies of Flying Robots: Can We Control Them? *The Atlantic*, November 2015. [Online; accessed 13-July-2016].

[RSA78] Ronald Rivest, Adi Shamir, and Leonard Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[SB14] William Stallings and Lawrie Brown. *Computer Security: Principles and Practice*. Pearson, Boston, MA, third edition, 2014.

[SC12] Dan Shoemaker and William Arthur Conklin. *Cybersecurity: The Essential Body of Knowledge*. Course Technology, Boston, MA, 2012.

[Sch96] Bruce Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. Wiley, New York, NY, second edition, 1996.

[Sch13] Fred B. Schneider. Cybersecurity Education in Universities. *IEEE Security and Privacy*, July/August 2013. [Online; accessed 8-October-2016].

[SF14] P.W. Singer and Allan Friedman. *Cybersecurity and Cyberwar: What Everyone Needs to Know*. Oxford University Press, 2014.

[Sho09] Victor Shoup. *A Computational Introduction to Number Theory and Algebra*. Cambridge University Press, second edition edition, 2009.

[Sim83] Gustavus J. Simmons. Verification of Treaty Compliance – Revisited. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 61–66. IEEE Conference Publications, 1983. [Online; accessed 23-July-2016].

[Sim92] Gustavus J. Simmons. *Contemporary Cryptology: The Science of Information Integrity*, chapter How to Insure That Data

Acquired to Verify Treaty Compliance Are Trustworthy, pages 615–630. Wiley-IEEE Press, Piscataway, NJ, 1992.

[Smi16] Richard E. Smith. *Elementary Information Security.* Jone & Bartlett Learning, Burlinton, MA, second edition, 2016.

[Spa16] Steve Spaleta. Eagles Trained to Intercept Drones. *Livescience*, February 3 2016. [Online; accessed 13-July-2016].

[SSD$^+$16] Travis Scheponik, Alan T. Sherman, David DeLatte, Dhananjay Phatak, Linda Oliva, Julia Thompson, and Geoffrey L. Herman. How Students Reason About Cybersecurity Concepts. In *Proceedings of the Frontiers in Education Conference*, October 2016.

[Ste95] Malcom W. Stevens. An Implementation of an Optical Data Diode. `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.83.8650&rep=rep1&type=pdf`, 1995. [Online; accessed 17-July-2016].

[Sti06] Douglas R. Stinson. *Cryptography: Theory and Practice.* Chapman & Hall/CRC, Boca Raton, FL, third edition, 2006.

[Szo15] David Szondy. Insect-Sized RoboBee Robot Can Now Fly and Swim. `http://www.gizmag.com/robobee-robot-fly-swim/39993/`, October 2015. [Online; accessed 30-May-2016].

[Ver10] James Verini. The Great Cyberheist. *The New York Times Magazine*, November 10 2010. `http://www.nytimes.com/2010/11/14/magazine/14Hacker-t.html`.

[Vil11] John Villasenor. The Drone Treat to National Security. *Scientific American*, November 11 2011. `https://www.scientificamerican.com/article/the-drone-threat-to-national-security/`.

[War15] B. Ward. Commercial Drones in the U.S.: Privacy, Ethics, Economics and Journalism. `http://journalistsresource.org/studies/economics/business/commercial-drones-united-states-privacy-ethics-economics`, 2015. [Online; accessed 9-30-16].

[Wei00] Steve H. Weingart. Physical Security Devices for Computer Subsystems: A Survey of Attacks and Defenses. In C.K. Ko and C. Paar, editors, *CHES 2000*, volume LNCS 1965, pages 302–317. Springer-Verlag, 2000.

[Wel15] A. Welch. Cost Benefit Analysis of Amazon Prime Air, 2015. honors thesis, University of Tennessee at Chattanooga.

[Wika] Firewall. `https://en.wikipedia.org/wiki/Firewall_(computing)`. [Online; accessed 15-July-2016].

[Wikb] SQL Injection. `https://en.wikipedia.org/wiki/SQL_injection#Examples`. [Online; accessed 14-July-2016].

[Wikc] Transmission Control Protocol. `https://en.wikipedia.org/wiki/Transmission_Control_Protocol`. [Online; accessed 22-July-2016].

[Wikd] Transport Layer Security. `https://en.wikipedia.org/wiki/Transport_Layer_Security`. [Online; accessed 14-July-2016].

[Wike] Virtual Private Network. `https://en.wikipedia.org/wiki/Virtual_private_network`. [Online; accessed 15-July-2016].

[Wikf] Wikipedia. `www.wikipedia.org`. [Online; accessed 14-July-2016].

[Zor12] Zeljka Zorz. $2.1 Million Stolen with Clever Social Engineering. `https://www.helpnetsecurity.com/2012/03/01/21-million-stolen-with-clever-`

`social-engineering/`, March 1 2012.
[Online; accessed 23-July-2016].

Submitted to *Cryptologia*.