Cyber Security Deception

Mohammed H. Almeshekah and Eugene H. Spafford

Abstract Our physical and digital worlds are converging at a rapid pace, putting a lot of our valuable information in digital formats. Currently, most computer systems' predictable responses provide attackers with valuable information on how to infiltrate them. In this chapter, we discuss how the use of deception can play a prominent role in enhancing the security of current computer systems. We show how deceptive techniques have been used in many successful computer breaches. Phishing, social engineering, and drive-by-downloads are some prime examples. We discuss why deception has only been used haphazardly in computer security. Additionally, we discuss some of the unique advantages deception-based security mechanisms bring to computer security. Finally, we present a framework where deception can be planned and integrated into computer defenses.

1 Introduction

Most data is digitized and stored in organizations' servers, making them a valuable target. Advanced persistent threats (APT), corporate espionage, and other forms of attacks are continuously increasing. Companies reported 142 million unsuccessful attacks in the first half of 2013, as reported by Fortinet [1]. In addition, a recent Verizon Data Breach Investigation Report (DBIR) points out that currently deployed protection mechanisms are not adequate to address current threats [1]. The report states that 66 % of the breaches took months or years to discover, rising from 56 % in 2012. Furthermore, 84 % of these attacks only took hours or less to infiltrate computer systems [1]. Moreover, the report states that only 5 % of these breaches were detected using traditional intrusion detection systems (IDSs) while 69 % were detected by external parties [1].

These numbers are only discussing attacks that were discovered. Because only 5 % of the attacks are discovered using traditional security tools, it is likely that the

M.H. Almeshekah (✉)
King Saud University, Riyadh, Saudi Arabia
e-mail: meshekah@ksu.edu.sa

E.H. Spafford
Purdue University, West Lafayette, IN, USA
e-mail: spaf@purdue.edu

S. Jajodia et al. (eds.), *Cyber Deception*, DOI 10.1007/978-3-319-32699-3_2

reality is significantly worse as there are unreported and undiscovered attacks. These findings show that the status quo of organizations' security posture is not enough to address current threats.

Within computer systems, software and protocols have been written for decades with an intent of providing useful feedback to every interaction. The original design of these systems is structured to ease the process of error detection and correction by informing the user about the exact reason why an interaction failed. This behavior enhances the efforts of malfeasors by giving them information that helps them to understand why their attack was not successful, refine their attacks and tools, and then re-attack. As a result, these systems are helpful to attackers and guide them throughout their attack. Meanwhile, targeted systems learn nothing about these attempts, other than a panic in the security team. In fact, in many cases multiple attempts that originate from the same entity are not successfully correlated.

Deception-based techniques provide significant advantages over traditional security controls. Currently, most security tools are responsive measures to attackers' probes to previously known vulnerabilities. Whenever an attack surfaces, it is hit hard with all preventative mechanisms at the defender's disposal. Eventually, persistent attackers find a vulnerability that leads to a successful infiltration by evading the way tools detect probes or by finding new unknown vulnerabilities. This security posture is partially driven by the assumption that "hacking-back" is unethical, while there is a difference between the act of "attacking back" and the act of deceiving attackers.

There is a fundamental difference in how deception-based mechanisms work in contrast to traditional security controls. The latter usually focuses on attackers' actions—detecting or preventing them—while the former focuses on attackers' perceptions—manipulating them and therefore inducing adversaries to take actions/inactions in ways that are advantageous to targeted systems; traditional security controls position themselves in response to attackers' actions while deception-based tools are positioned in prospect of such actions.

1.1 Definition

One of the most widely accepted definitions of computer-security deception is the one by Yuill [2]; Computer Deception is "Planned actions taken to mislead attackers and to thereby cause them to take (or not take) specific actions that aid computer-security defenses." We adapt this definition and add "confusion" as one of goals of using deceit (the expression of things that are not true) in computer system protection. Therefore, the definition of defensive computer deception we will use throughout this chapter is

Definition 1. Deception is "Planned actions taken to mislead and/or confuse attackers and to thereby cause them to take (or not take) specific actions that aid computer-security defenses."

2 A Brief History

Throughout history, deception has evolved to find its natural place in our societies and eventually our technical systems. Deception and decoy-based mechanisms have been used in security for more than two decades in mechanisms such as honeypots and honeytokens. An early example of how deception was used to attribute and study attackers can be seen in the work of Cheswick in his well-known paper "An Evening with Berferd" [3]. He discusses how he interacted with an attacker in real time providing him with fabricated responses. Two of the earliest documented uses of deceptive techniques for computer security are in the work of Cliff Stoll in his book "The Cuckoo's Egg" [4] and the work of Spafford in his own lab [5]. The Deception Toolkit (DTK),[1] developed by Fred Cohen in 1997 was one of the first publicly available tools to use deception for the purpose of computer defenses.

In late 1990s, "honeypots"—"a component that provides its value by being attacked by an adversary" i.e. deceiving the attacker to interact with them—have been used in computer security. In 2003, Spitzner published his book on "Honeypots" discussing how they can be used to enhance computer defenses [6]. Following on the idea of honeypots, a proliferation of "honey-*" prefixed tools have been proposed. Additionally, with the release of Tripwire, Kim and Spafford suggested the use of planted files that should not be accessed by normal users, with interesting names and/or locations and serving as bait that will trigger an alarm if they are accessed by intruders [7].

2.1 Honey-Based Tools

2.1.1 Honeypots

Honeypots have been used in multiple security applications such as detecting and stopping spam[2] and analyzing malware [8]. In addition, honeypots have been used to secure databases [9]. They are starting to find their way into mobile environments [10] where some interesting results have been reported [11].

Honeypots in the literature come in two different types: server honeypot and client honeypot. The server honeypot is a computer system that contains no valuable information and is designed to appear vulnerable for the goal of enticing attackers to access them. Client honeypots are more active. These are vulnerable user agents that troll many servers actively trying to get compromised [12]. When such incidents happen, the client honeypots report the servers that are infecting users' clients. Honeypots have been used in computing in four main areas as we discuss in the following paragraphs.

[1] http://www.all.net/dtk/.

[2] http://www.projecthoneypot.org.

Detection

Honeypots provide an additional advantage over traditional detection mechanisms such as *Intrusion Detection Systems (IDS)* and anomaly detection. First, they generate less logging data as they are not intended to be used as part of normal operations and thus any interaction with them is illicit. Second, the rate of false positive is low as no one should interact with them for normal operations. Angnostakis et al. proposed an advanced honeypot-based detection architecture in the use of *shadow honeypots* [13]. In their scheme they position *Anomaly Detection Sensors (ADSs)* in front of the real system where a decision is made as whether to send the request to a *shadow* machine or to the normal machine. The scheme attempts to integrate honeypots with real systems by seamlessly diverting suspicious traffic to the shadow system for further investigation. Finally, honeypots are also helpful in detecting industry-wide attacks and outbreaks, e.g. the case of the Slammer worm as discussed in [14].

Prevention

Honeypots are used in prevention where they assist in slowing down the attackers and/or deterring them. *Sticky honeypots* are one example of machines that utilize unused IP address space and interact with attackers probing the network to slow them down [15]. In addition, Cohen argues that by using his Deception ToolKit (DTK) we can deter attackers confusing them and introducing risk on their side [16]. However, we are not aware of any studies that investigated those claims.

Beyond the notion of enticement and traps used in honeypots, deception has been studied from other perspectives. For example, Rowe et al. present a novel way of using honeypots for deterrence [17]. They protect systems by making them look like a honeypot and therefore deter attackers from accessing them. Their observation stemmed from the developments of anti-honeypots techniques that employ advanced methods to detect if the current system is a honeypot [18].

Response

One of the advantages of using honeypots is that they are totally independent systems that can be disconnected and analyzed after a successful attack on them without hindering the functionality of the production systems. This simplifies the task of forensic analysts as they can preserve the *attacked* state of the system and extensively analyze what went wrong.

Research

Honeypots are heavily used in analyzing and researching new families of malware. The honeynet project[3] is an "international non-profit security research organization, dedicated to investigating the latest attacks and developing open source security tools to improve Internet security." For example, the HoneyComb system uses honeypots to create unique attack signatures [19]. Other more specific tools, such as *dionaea*,[4] are designed to capture a copy of computer malware for further study. Furthermore, honeypots help in inferring and understanding some widespread attacks such as Distributed Denial of Service (DDoS) [20].

2.1.2 Other Honey Prefixed Tools

The prefix "honey-*" has been used to refer to a wide range of techniques that incorporate the act of deceit in them. The basic idea behind the use of the prefix word "honey" in these techniques is that they need to entice attackers to interact with them, i.e. fall for the bait—the "honey." When such an interaction occurs the value of these methods is realized.

The term honeytokens has been proposed by Spitzner [21] to refer to honeypots but at a smaller granularity. Stoll used a number of files with enticing names and distributed them in the targeted computer systems, acting as a beaconing mechanism when they are accessed, to track down Markus Hess [4]. Yuill et al. coined the term *honeyfiles* to refer to these files [22]. HoneyGen was also used to refer to tools that are used to generate honeytokens [23].

Most recently, a scheme named *Honeywords* was proposed by Jules and Rivest to confuse attackers when they crack a stolen hashed password file [24] by hiding the real password among a list of "fake" ones. Their scheme augmenting password databases with an additional $(N - 1)$ fake credentials [24]. If the DB is stolen and cracked, attackers are faced with N different passwords to choose from where only one of them is the correct one. However, if they use any of the fake ones the system triggers an alarm alerting system administrators that the DB has been cracked.

2.2 Limitations of Isolated Use of Deception

Honeypot-based tools are a valuable technique used for the detection, prevention, and response to cyber attacks as we discuss in this chapter. Nevertheless, those techniques suffer from the following major limitations:

[3] www.honeynet.org.

[4] http://dionaea.carnivore.it/.

- As the prefix *honey-** indicates, for such techniques to become useful, the adversary needs to interact with them. Attackers and malware are increasingly becoming sophisticated and their ability to avoid honeypots is increasing [25].
- Assuming we manage to lure the attacker into our honeypot, we need to be able to *continuously* deceive them that they are in the real system. Chen et al. study such a challenge and show that some malware, such as polymorphic malware, not only detects honeypots, but also changes its behavior to deceive the honeypot itself [25]. In this situation, attackers are in a position where they have the ability to conduct counter-deception activities by behaving in a manner that is different than how would they do in a real environment.
- To learn about attackers' objectives and attribute them, we need them to interact with the honeypot systems. However, with a high-interaction honeypot there is a risk that attackers might exploit the honeypot itself and use it as a pivot point to compromise other, more sensitive, parts of the organization's internal systems. Of course, with correct separation and DMZs we can alleviate the damage, but many organizations consider the risk intolerable and simply avoid using such tools.
- As honeypots are totally "fake systems" many tools currently exist to identify whether the current system is a honeypot or not [18, 25]. This fundamental limitation is intrinsic in their design.

3 Deception as a Security Technique

Achieving security cannot be done with single, silver-bullet solutions; instead, good security involves a collection of mechanisms that work together to balance the cost of securing our systems with the possible damage caused by security compromises, and drive the success rate of attackers to the lowest possible level. In Fig. 1, we present a taxonomy of protection mechanisms commonly used in systems. The diagram shows four major categories of protection mechanisms and illustrates how they intersect achieving multiple goals.

The rationale behind having these intersecting categories is that a single layer of security is not adequate to protect organizations so multi-level security controls are needed [26]. In this model, the first goal is to deny unauthorized access and isolate our information systems from untrusted agents. However, if adversaries succeed in penetrating these security controls, we should have degradation and obfuscation mechanisms in place that slow the lateral movement of attackers in penetrating our internal systems. At the same time, this makes the extraction of information from penetrated systems more challenging.

Even if we slow the attackers down and obfuscate our information, advanced adversaries may explore our systems undetected. This motivates the need for a third level of security controls that involves using means of deceit and negative information. These techniques are designed to lead attackers astray and augment our systems with decoys to detect stealthy adversaries. Furthermore, this deceitful information will waste the time of the attackers and/or add risk during their

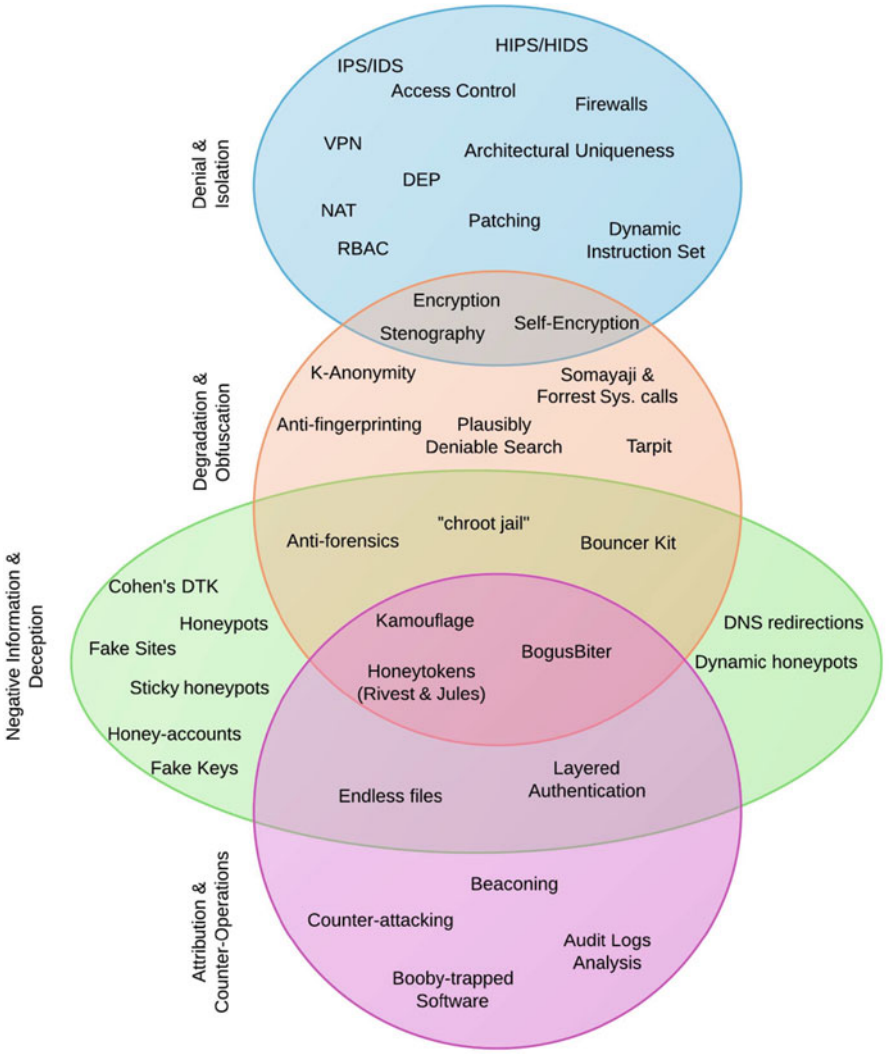

Fig. 1 Taxonomy of information protection mechanisms

infiltration. The final group of mechanisms in our taxonomy is designed to attribute attackers and give us the ability to have counter-operations. Booby-trapped software is one example of counter-operations that can be employed.

Securing a system is an economic activity and organizations have to strike the right balance between cost and benefits. Our taxonomy provides a holistic overview of security controls, with an understanding of the goals of each group and how can they interact with each other. This empowers decision makers on what and which security controls they should deploy.

Despite all the efforts organizations have in place, attackers might infiltrate information systems, and operate without being detected or slowed. In addition, persistent adversaries might infiltrat the system and passively observe for a while to avoid being detected and/or slowed when moving on to their targets. As a result, a deceptive layer of defense is needed to augment our systems with negative and deceiving information to lead attackers astray. We may also significantly enhance organizational intrusion detection capabilities by deploying detection methods using multiple, additional facets.

Deception techniques are an integral part of human nature that is used around us all the time. As an example of a deception widely used in sports: teams attempt to deceive the other team into believing they are following a particular plan so as to influence their course of action. Use of cosmetics may also be viewed as a form of mild deception. We use white lies in conversation to hide mild lapses in etiquette. In cybersecurity, deception and decoy-based mechanisms haven been used in security for more than two decades in technologies such as honeypots and honeytokens.

When attackers infiltrate the system and successfully overcome traditional detection and degradation mechanisms we would like to have the ability to not only obfuscate our data, but also lead the attackers astray by deceiving them and drawing their attention to other data that are false or intestinally misleading. Furthermore, exhausting the attacker and causing frustration is also a successful defensive outcome. This can be achieved by planting fake keys and/or using schemes such as endless files [5]. These files look small on the organization servers but when downloaded to be exfiltrated will exhaust the adversaries' bandwidth and raise some alarms. Moreover, with carefully designed deceiving information we can even cause damage at the adversaries' servers. A traditional, successful, deception technique can be learned from the well-known story of Farewell Dossier during the cold war where the CIA provided modified items to a Soviet spy ring. When the Soviets used these designs thinking they are legitimate, it resulted in a major disaster affecting a trans-Siberian pipeline.

When we inject false information we cause some confusion for the adversaries even if they have already obtained some sensitive information; the injection of negative information can degrade and/or devalue the correct information obtained by adversaries. Heckman and his team, from Lockheed Martin, conducted an experiment between a red and a blue team using some deception techniques, where they found some interesting results [27]. Even after the red team successfully attacked and infiltrate the blue system and obtained sensitive information, the blue team injected some false information in their system that led the red team to devalue the information they had obtained, believing that the new values were correct.

Another relationship can be observed between the last group of protection techniques, namely attribution, and deception techniques. Deception-based mechanisms are an effective way to lure attackers to expose themselves and their objectives when we detect them accessing things and conducting unusual activities. Other tools, such as anomaly-based IDS, have similar goals, but the advantage deception-based tools have is that there is a clear line between normal user activities and abnormal ones. This is because legitimate users are clearly not supposed to access this information.

This difference significantly enhances the effectiveness of deception-based security controls and reduces the number of false-positives, as well as the size of the system's log file.

3.1 Advantages of Using Deception in Computer Defenses

Reginald Jones, the British scientific military intelligence scholar, concisely articulated the relationship between security and deception. He referred to security as a "negative activity, in that you are trying to stop the flow of clues to an opponent" and it needs its other counterpart, namely deception, to have a competitive advantage in a conflict [28]. He refers to deception as the "positive counterpart to security" that provides false clues to be fed to opponents.

By intelligently using deceptive techniques, system defenders can mislead and/or confuse attackers, thus enhancing their defensive capabilities over time. By exploiting attackers' unquestioned *trust* of computer system responses, system defenders can gain an edge and position themselves a step ahead of compromise attempts. In general, deception-based security defenses bring the following unique advantages to computer systems [29]

1. *Increases the entropy of leaked information about targeted systems during compromise attempts.*

   When a computer system is targeted, the focus is usually only on protecting and defending it. With deception, extra defensive measures can be taken by feeding attackers false information that will, in addition to defending the targeted system, cause intruders to make wrong actions/inactions and draw incorrect conclusions. With the increased spread of APT attacks and government/corporate espionage threats such techniques can be effective.

   When we inject false information we cause some confusion for the adversaries even if they have already obtained some sensitive information; the injection of negative information can degrade and devalue the correct information obtained by adversaries. Heckman and her team, developed a tool, referred to as "Blackjack," that dynamically copies an internal state of a production server—after removing sensitive information and injecting deceit—and then directs adversaries to that instance [27]. Even after the red team successfully attacked and infiltrated the blue systems and obtained sensitive information, the blue team injected some false information in their system that led the red team to devalue the information they had obtained, believing that the new values were correct.

2. *Increases the information obtained from compromise attempts.*
   Many security controls are designed to create a boundary around computer systems automatically stopping any illicit access attempts. This is becoming increasingly challenging as such boundaries are increasingly blurring, partly as a result of recent trends such as "consumerization"[5] [30]. Moreover, because of the low cost on the adversaries' side, and the existence of many automated exploitation tools, attackers can continuously probe computer systems until they find a vulnerability to infiltrate undetected. During this process, systems' defenders learn nothing about the intruders' targets. Ironically, this makes the task of defending a computer system harder after every unsuccessful attack. We conjecture that incorporating deception-based techniques can enhance our understanding of compromise attempts using the illicit probing activity as opportunity to enhance our understanding of the threats and, therefore, better protect our systems over time.
3. *Give defenders an edge in the OODA loop.*
   The OODA loop (for Observe, Orient, Decide, and Act) is a cyclic process model, proposed by John Boyd, by which an entity reacts to an event [31]. The victory in any tactical conflict requires executing this loop in a manner that is faster than the opponent. The act of defending a computer system against persistent attacks can be viewed as an OODA loop race between the attacker and the defender. The winner of this conflict is the entity that executes this loop faster. One critical advantage of deception-based defenses is that they give defenders an edge in such a race as they actively feed adversaries deceptive information that affects their OODA loop, more specifically the "observe" and "orient" stages of the loop. Furthermore, slowing the adversary's process gives defenders more time to decide and act. This is especially crucial in the situation of surprise, which is a common theme in digital attacks.
4. *Increases the risk of attacking computer systems from the adversaries' side.*
   Many current security controls focus on preventing the actions associated with illicit attempts to access computer systems. As a result, intruders are using this accurate negative feedback as an indication that their attempts have been detected. Subsequently, they withdraw and use other, more stealthy, methods of infiltration. Incorporating deceit in the design of computer systems introduces a new possibility that adversaries need to account for; namely that they have been detected and currently deceived. This new possibility can deter attackers who are not willing to take the risk of being deceived, and further analyzed. In addition, such technique gives systems' defenders the ability to use intruders' infiltration attempts to their advantage by actively feeding them false information.

[5]This term is widely used to refer to enterprises' employees bringing their own digital devises and using them to access the companies' resources.

3.2 Deception in the Cyber Kill-Chain

The cyber kill-chain introduced by Lockheed Martin researchers advocates an intelligence-driven security model [32]. The main premise behind this model is that for attackers to be successful they need to go through all these steps in the chain in sequence. Breaking the chain at any step will break the attack and the earlier that we break it the better we prevent the attackers from attacking our systems.

The cyber kill-chain model is a good framework to demonstrate the effectiveness of incorporating deception at multiple levels in the chain. With the same underlying principle of the kill-chain—early detection of adversaries—we argue that the earlier we detect adversaries, the better we are at deceiving them and learning more about their methods and techniques. We postulate that full intelligence cannot be gathered without using some means of deception techniques.

Also, the better we know our enemies the better we can defend against them. By using means of deception we can continuously learn about attackers at different levels of the kill-chain and enhance our capabilities of detecting them and reducing their abilities to attack us. This negative correlation is an interesting relationship between our ability to detect attackers and their ability to probe our resources.

There is a consensus that we would like to be at least one step ahead of adversaries when they attack our systems. We argue that by intelligently incorporating deception methods in our security models we can start achieving that. This is because the further we enhance our abilities to detect adversaries the further ahead of them we position ourselves. If we take an example of external network probing, if we simply detect an attack and identify a set of IP address and domain names as "bad," we do not achieve much: these can be easily changed and adversaries will become more careful not to raise an alarm the next time they probe our systems. However, if we go one more step to attribute them by factors that are more difficult to change it can cause greater difficulty for future attacks. For example, if we are able to deceive attackers in manners that allow us to gather more information that allows us to distinguish them based on fixed artifacts (such as distinctive protocol headers, known tools and/or behavior and traits) we have a better position for defense. The attackers will now have a less clear idea of how we are able to detect them, and when they know, it should be more difficult for them to change these attributes.

The deployment of the cyber kill-chain was seen as fruitful for Lockheed when they were able to detect an intruder who successfully logged into their system using the SecurID attack [33]. We adopt this model with slight modification to better reflect our additions.

Many deception techniques, such as honeypots, work in isolation and independently of other parts of current information systems. This design decision has been partly driven by the security risks associated with honeypots. We argue that intelligently augmenting our systems with interacting deception-based techniques can significantly enhance our security and gives us the ability to achieve deception in depth. If we examine Table 1, we can see that we can apply deception at every stage of the cyber kill-chain, allowing us to break the chain and possibly attribute

Table 1 Mapping deception to the kill-chain model

Cyber kill-chain phase	Deception
Reconnaissance	Artificial ports, fake sites
Weaponization and delivery	Create artificial bouncing back, sticky honeypots
Exploitation and installation	Create artificial exploitation response
Command and control (operation)	Honeypot
Lateral movement and persistence	HoneyAccounts, honeyFiles
Staging and exfiltration	Honeytokens, endless files, fake keys

attackers. At the reconnaissance stage we can lure adversaries by creating a site and have honey-activities that mimic a real-world organization. As an example, an organization can subscribe with a number of cloud service providers and have honey activities in place while monitoring any activities that signal external interest. Another example is to address the problem of spear-phishing by creating a number of fake persons and disseminating their information into the Internet while at the same monitoring their contact details to detect any probing activities; some commercial security firms currently do this.

3.3 Deception and Obscurity

Deception always involves two basic steps, hiding the real and showing the false. This, at first glance, contradicts the widely believed misinterpretation of Kerckhoff's principle; "no security through obscurity." A more correct English translation of Kerckhoff's principle is the one provided by Petitcolas in [34];

> The system must not require secrecy and can be stolen by the enemy without causing trouble.

The misinterpretation leads some security practitioners to believe that any "obscurity" is ineffective, while this is not the case. Hiding a system from an attacker or having a secret password does increase the work factor for the attacker—until the deception is detected and defeated. So long as the security does not materially depend on the obscurity, the addition of misdirection and deceit provides an advantage. It is therefore valuable for a designer to include such mechanisms in a comprehensive defense, with the knowledge that such mechanisms should not be viewed as primary defenses.

In any system design there are three levels of viewing a system's behavior and responses to service requests [29]:

- *Truthful.* In such systems, the processes will always respond to any input with full "honesty." In other words, the system's responses are always "trusted" and accurately represent the internal state of the machine. For example, when the user

asks for a particular network port, a truthful system responds with either a real port number or denies the request giving the specific reason of such denial.

- *Naively Deceptive.* In such systems, the processes attempt to deceive the interacting user by crafting an artificial response. However, if the user knows the deceptive behavior, e.g. by analyzing the previous deceptive response used by the system, the deception act becomes useless and will only alert the user that the system is trying to deceive her. For example, the system can designate a specific port that is used for deceptive purposes. When the attacker asks for a port, without carrying the appropriate permissions, this deceptive port is sent back.
- *Intelligently Deceptive.* In this case, the systems "deceptive behavior" is indistinguishable from the normal behavior even if the user has previously interacted with the system. For example, an intelligently-deceptive system responds to unauthorized port listening requests the same as a normal allowed request. However, extra actions are taken to monitor the port, alert the system administrators, and/or sandbox the listening process to limit the damage if the process downloads malicious content.

3.4 Offensive Deception

Offensively, many current, common attacks use deceptive techniques as a cornerstone of their success. For example, phishing attacks often use two-level deceptive techniques; they deceive users into clicking on links that appear to be coming from legitimate sources, which take them to the second level of deception where they will be presented with legitimate-looking websites luring them to give their credentials. The "Nigerian 419" scams are another example of how users are deceived into providing sensitive information with the hope of receiving a fortune later.

In many of these cases, attackers focus on deceiving users as they are usually the most vulnerable component. Kevin Mitnick showed a number of examples in his book, "The Art of Deception" [35], of how he used social engineering, i.e., deceptive skills to gain access to many computer systems. Trojan horses, which are more than 30 years old, are a prime example of how deception has been used to infiltrate systems.

Phishing, Cross-site Scripting (XSS) [36], and Cross-site Request Forgery (XSRF) [37] are some examples of using deception. Despite more than a decade of research by both the academic and private sectors, these problems are causing more damage every year. XSS and XSRF have remained on the OWASP's top ten list since the first time they were added in 2007 [38]. The effectiveness of offensive deception techniques should motivate security researchers to think of positive applications for deception in security defenses.

4 A Framework to Integrate Deception in Computer Defenses

We presented a framework that can be used to plan and integrate deception in computer security defenses [39]. Many computer defenses that use deception were ad-hoc attempts to incorporate deceptive elements in their design. We show how our framework can be used to incorporate deception in many parts of a computer system and discuss how we can use such techniques effectively. A successful deception should present plausible alternative(s) to the truth and these should be designed to exploit specific adversaries' biases, as we will discuss later.

The framework discussed in this chapter is based on the general deception model discussed by Bell and Whaley in [40]. There are three general phases of any deceptive component; namely planning, implementing and integrating, and finally monitoring and evaluating. In the following sections we discuss each one of those phases in more detail. The framework is depicted in Fig. 3.

4.1 The Role of Biases

In cognitive psychology a bias refers to

> An inclination to judge others or interpret situations based on a personal and oftentimes unreasonable point of view [41]

Biases are a cornerstone component to the success of any deception-based mechanism. The target of the deception needs to be presented with a plausible "deceit" to successfully deceive and/or confuse him. If the target perceives this deceit to be non-plausible she is more inclined to reject it instead of believing it, or at least raise her suspicions about the possibility of currently being deceived. A successful deception should *exploit* a bias in the attackers' perception and provide them with one or more plausible alternative information other than the truth.

Thompson et al. discuss four major groups of biases any analysts need to be aware of: personal biases, cultural biases, organizational biases, and cognitive biases [42]. It can be seen in Fig. 2 that the more specific the bias being exploited in a deceptive security tool is, the less such a tool can be generalized, For example, exploiting a number of personal biases, specific to an attacker, might not be easily generalized to other adversaries who attack your system. However, the more specific the choice of bias enhances the effectiveness of the deceptive component. This is true partly because cognitive biases are well-known and adversaries might intentionally guard themselves with an additional layer of explicit reasoning to minimize their effects in manipulating their perceptions. In the following paragraphs we discuss each one of these classes of biases.

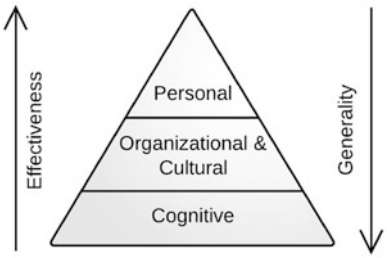

Fig. 2 Deception target biases

4.1.1 Personal Biases

Personal biases are those biases that originate from either first-hand experiences or personal traits, as discussed by Jervis in [43]. These biases can be helpful in designing deceptive components/operation; however, they are (1) harder to obtain as they require specific knowledge of potential attackers and (2) they make deceptive components less applicable to a wider range of attackers while becoming more powerful against specific attackers. Personal biases have been exploited in traditional deception operations in war, such as exploiting the arrogance of Hitler's administration in World War II as part of Operation Fortitude [41].

4.1.2 Cultural Biases

Hofstede refers to cultural biases as the "software of the mind" [44]. They represent the mental and cognitive ways of thinking, perception, and action by humans belonging to these cultures. In a study conducted by Guss and Dorner, they found that cultures influenced the subjects' perception, strategy development and decision choices, even though all those subjects were presented with the same data [45]. Hofstede discusses six main dimensions of cultures and assigns quantitative values to those dimensions for each culture in his website (geerte-hofstede.com). Also, he associates different behavior that correlates with his measurements. Theses dimensions are:

1. **Power Distance Index (PDI)**—PDI is a measure of the expectation and acceptance that "power is distributed unequally." Hofstede found that cultures with high PDI tend to have a sense of loyalty, show of strength, and preference to in-group-person. This feature can be exploited by a deception planner focusing on the attacker's sense of pride to reveal himself, knowing that the attack is originating from a high PDI culture with a show-of-strength property.
2. **Individualism versus Collectivism (IVC)**—A collectivist society values the "betterment of a group" at the expense of the individual. Hofstede found that most cultures are collectivist, i.e. with low IVC index.
3. **Masculine versus Feminine (MVF)**—A masculine culture is a culture where "emotional gender roles are clearly distinct." For example, an attacker coming

from a masculine culture is more likely to discredit information and warnings written by or addressed to a female. In this case, this bias can be exploited to influence attackers' behaviors.

4. **Uncertainty Avoidance Cultures (UAI)**—This measures the cultural response to the unknown or the unexpected. High UAI means that this culture has a fairly structured response to uncertainty making the attackers' anticipation of deception and confusion a much easier task.
5. **Long-Term Orientation Versus Short-Term Orientation (LTO vs. STO)**—STO cultures usually seek immediate gratification. For example, the defender may sacrifice information of lesser importance to deceive an attacker into thinking that such information is of importance, in support of an over-arching goal of protecting the most important information.
6. **Indulgence versus Restraint (IVR)**—This dimension characterizes cultures on their norms of how they choose activities for leisure time and happiness.

Wirtz and Godson summarize the importance of accounting for cultures while designing deception in the following quote; "To be successful the deceiver must recognize the target's perceptual context to know what (false) pictures of the world will appear plausible" [46].

4.1.3 Organizational Biases

Organizational biases are of importance when designing deception for an target within a heavily structured environment [41]. In such organizations there are many keepers who have the job of analyzing information and deciding what is to be passed to higher levels of analysts. This is one example of how organizational biases can be used. These biases can be exploited causing important information to be marked as less important while causing deceit to be passed to higher levels. One example of organizational biases is uneven distribution of information led to uneven perception and failure to anticipate the Pearl Harbor attack in 1941 by the United States [41].

4.1.4 Cognitive Biases

Cognitive biases are common among all humans across all cultures, personalities, and organizations. They represent the "innate ways human beings perceive, recall, and process information" [41]. These biases have long been studied by many researchers around the world in many disciplines (particularly in cognitive psychology); they are of importance to deception design as well as computing.

Tversky and Kahneman proposed three general heuristics our minds seem to use to reduce a complex task to a simpler judgment decision—especially under conditions of uncertainty—thus leading to some predictable biases [47]. These are: representativeness, availability, and anchoring and adjustment. They defined the representativeness heuristic as a "heuristic to evaluate the probability of an

event by the degree to which it is (i) similar in essential properties to its parent population; and (ii) reflects the salient features of the process by which it is generated" [47]. The availability heuristic is another bias that assess the likelihood of an uncertain event by the *ease* with which someone can bring it to mind. Finally, the anchoring/adjustment heuristic is a bias that causes us to make estimations closer to the initial values we have been provided with than is otherwise warranted.

Solman presented a discussion of two reasoning systems postulated to be common in humans: associative (system 1) and rule-based (system 2) [48]. System 1 is usually automatic and heuristic-based, and is usually governed by habits. System 2 is usually more logical with rules and principles. Both systems are theorized to work simultaneously in the human brain; deception targets System 1 to achieve more desirable reactions.

In 1994, Tversky and Koehler argued that people do not subjectively attach probability judgments to events; instead they attach probabilities to the description of these events [49]. That is, two different descriptions of the same event often lead people to assign different probabilities to their likelihood. Moreover, the authors postulate that the more *explicit* and detailed the description of the event is, the higher the probability people assign to it. In addition, they found that unpacking the description of the event into several disjoint components increases the probability people attach to it. Their work provides an explanation for the errors often found in probability assessments associated with the "conjunction fallacy" [50]. Tversky and Kahneman found that people usually would give a higher probability to the conjunction of two events, e.g. P(X and Y), than a single event, e.g. P(X) or P(Y). They showed that humans are usually more inclined to *believe* a detailed story with explicit details over a short compact one.

4.2 Planning Deception

There are six essential steps to planning a successful deception-based defensive component. The first, and often neglected, step is specifying exactly the *strategic goals* the defender wants to achieve. Simply augmenting a computer system with honey-like components, such as honeypots and honeyfiles, gives us a false sense that we are using deception to lie to adversaries. It is essential to detail exactly what are the goals of using any deception-based mechanisms. As an example, it is significantly different to set up a honeypot for the purpose of simply capturing malware than having a honeypot to closely monitor APT-like attacks.

After specifying the strategic goals of the deception process, we need to specify—in the second step of the framework—how the target (attacker) should react to the deception. This determination is critical to the long-term success of any deceptive process. For example the work of Zhao and Mannan in [51] deceive attackers launching online guessing attacks into believing that they have found a correct username and password. The strategic goal of this deception process is to direct an attacker to a "fake" account thus wasting their resources and monitoring

their activities to learn about their objectives. It is crucial to analyze how the target should react after the successful "fake" login. The obvious reaction is that the attacker would continue to laterally move in the target system, attempting further compromise. However, an alternative response is that the attacker ceases the guessing attack and reports to its command and control that a successful username/password pair has been found. In consideration of the second alternative we might need to maintain the username/password pair of the fake account and keep that account information consistent for future targeting.

Moreover, part of this second step is to specify how we desire an attacker to react such that we may try to influence his perception and thus lead him to the desired reaction. Continuing with the example in the previous paragraph, if we want the attacker to login again so we have more time to monitor and setup a fake account, we might cause an artificial network disconnection that will cause the target to login again.

4.2.1 Adversaries' Biases

Deception-based defenses are useful tools that have been shown to be effective in many human conflicts. Their effectiveness relies on the fact that they are designed to exploit specific biases in how people think, making them appear to be plausible but false alternatives to the hidden truth, as discussed above. These mechanisms give defenders the ability to learn more about their attackers, reduce indirect information leakages in their systems, and provide an advantage with regard to their defenses.

Step 3 of planning deception is to understand the attackers' biases. As discussed earlier, biases are a cornerstone component to the success of any deception-based mechanisms. The deceiver needs to present a plausible deceit to successfully deceive and/or confuse an adversary. If attackers decide that such information is not plausible they are more inclined to reject it, or at least raise their suspicions about the possibility of currently being deceived. When the defender determines the strategic goal of the deception and the desired reactions by the target, he needs to investigate the attacker's biases to decide how best to influence the attacker's perception to achieve the desired reactions.

One example of using biases in developing some deceptive computer defenses is using the "confirmation bias" to lead adversaries astray and waste their time and resources. Confirmation bias is defined as "the seeking or interpreting of evidence in ways that are partial to existing beliefs, expectations, or a hypothesis in hand" [52]. A computer defender can use this bias in responding to a known adversarial probing of the system's perimeter. Traditional security defenses are intended to detect and prevent such activity, by simply dropping such requests or actively responding with an explicit denial. Taking this a step further by exploiting some pre-existing expectation, i.e. the confirmation bias, we might provide a response that the system is being taken down for some regular maintenance or as a result of some unexpected failure. With such a response, the defender manages to prevent illicit activity, provide a pause to consider next steps for the defender, and perhaps waste the adversary's time as they wait or investigate other alternatives to continue their attacks.

Cultural biases play an important role in designing deceptive responses, as discussed in Sect. 4.1.2. For example, some studies found relationships between the type of computer attacks and the culture/country from which the attack originated [53].

In computing, the conjunction fallacy bias, discussed in Sect. 4.1.4, can be exploited by presenting the deception story as a conjunction of multiple detailed components. For example, if deceivers want to misinform an attacker probing their system by creating an artificial network failure, instead of simply blocking these attempts, it is better to give a longer story. A message that says "Sorry the network is down for some scheduled network maintenance. Please come back in three hours" is more plausible than simply saying "The network is down" and thus more likely to be believed.

4.2.2 Creating the Deception Story

After analyzing attackers' biases the deceiver needs to decide exactly what components to simulate/dissimulate; namely step 4 of the framework in Fig. 3.

In Fig. 4 we provide an overview of the different system components where deception can be applied, exploiting the attacker's biases to achieve the desired reaction. Overall, deceit can be injected into the functionality and/or state of our systems. We give a discussion of each one of these categories below and present some examples.

System's Decisions

We can apply deception to the different decisions any computer system makes. As an example, Zhao and Mannan work in [51] apply deception at the system's authentication decision where they deceive adversaries by giving them access to "fake" accounts in the cases of online guessing attacks. Another system's decision we can use concerns firewalls. Traditionally, we add firewall rules that prevent specific IP addresses from interacting with our systems after detecting that they are sources of some attacks. We consider this another form of data leakage in accordance with the discussion of Zhao and Mannan in [51]. Therefore, we can augment firewalls by applying deception to their decisions by presenting adversaries with plausible responses other than simply denying access.

System's Software and Services

Reconnaissance is the first stage of any attack on any computing system, as identified in the kill-chain model [32]. Providing fake systems and services has been the main focus of honeypot-based mechanisms. Honeypots discussed earlier in this chapter are intended to provide attackers with a number of fake systems running

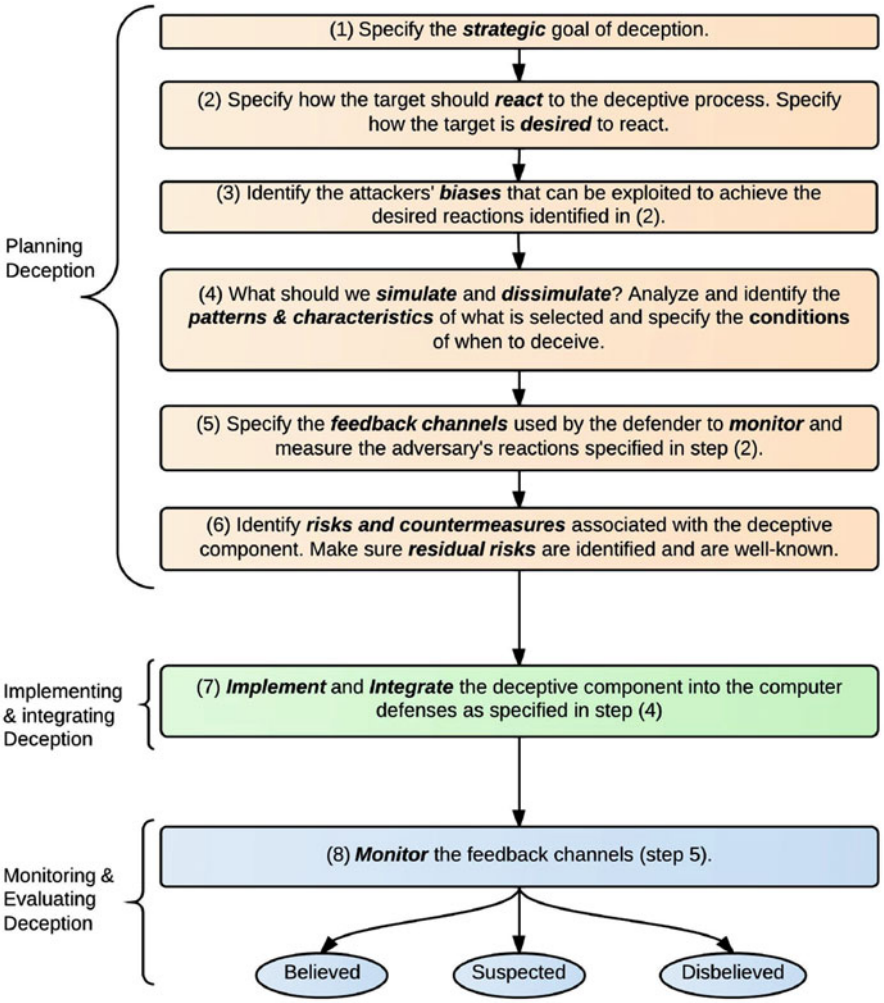

Fig. 3 Framework to incorporate deception in computer security defenses

fake services. Moreover, we can use deception to mask the identities of our current existing software/services. The work of Murphy et al. in [54] recommended the use of operating system obfuscation tools for Air Force computer defenses.

System's Internal and Public Data

A honeyfile, discussed above, is an example of injecting deceit into the system's internal data. It can be applied to the raw data in computer systems, e.g., files and directories, or to the administrative data that are used to make decisions and/or

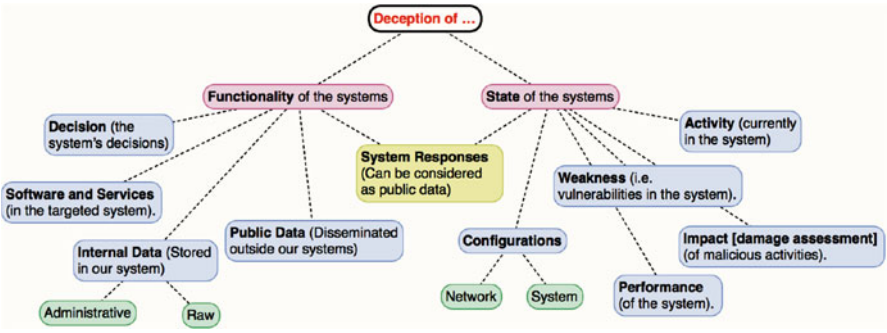

Fig. 4 Computer systems components where deception can be integrated with

monitor the system's activities. An example applying deception to the administrative data can be seen in the *honeywords* proposal [24]. Deceit can also be injected into the public data about our systems. Wang et al. made the case of disseminating public data about some "fake" personnel for the purpose of catching attacks such as spear phishing [55]. Cliff Stoll did this during the story of his book [4]. In addition, we note that this category also includes offline stored data such as back-ups that can be used as a focus of deception.

System's Activity

Different activities within a system are considered as one source of information leakage. For example, traffic flow analysis has long been studied as a means for attackers to deduce information [56]. Additionally, a system's activity has been used as a means of distinguishing between a "fake" and a real system [25]. We can intelligently inject some data about activities into our system to influence attackers' perception and, therefore, their reactions.

System's Weaknesses

Adversaries probe computer systems trying to discover and then exploit any weakness (vulnerability). Often, these adversaries come prepared with a list of possible vulnerabilities and then try to use them until they discover something that works. Traditional security mechanisms aid adversaries by quickly and promptly responding back to any attempt to exploit fixed, i.e. patched, vulnerabilities with a denial response. This response leaks information that these vulnerabilities are known and fixed. When we inject deceit into this aspect of our systems we can misinform adversaries by confusing them—by not giving them a definitive answer whether the exploit has succeeded—or by deceiving them by making it appear as if the vulnerability has been exploited.

System's Damage Assessment

This relates to the previous component; however, the focus here is to make the attacker perceive that the damage caused is more or less than the real damage. We may want the adversary to believe that he has caused more damage than what has happened so as to either stop the attack or cause the attacker to become less aggressive. This is especially important in the context of the OODA loop discussed earlier in this chapter. We might want the adversary to believe that he has caused less damage if we want to learn more about the attacker by prompting a more aggressive attack.

System's Performance

Influencing the attacker's perception of system's performance may put the deceiver at an advantageous position. This has been seen in the use of *sticky honeypots* and tarpits discussed at the beginning of this chapter that are intended to slow the adversary's probing activity. Also, tarpits have been used to throttle the spread of network malware. In a related fashion, Somayaji et al. proposed a method to deal with intrusions by slowing the operating system response to a series of anomalous system calls [57].

System's Configurations

Knowledge of the configuration of the defender's systems and networks is often of great importance to the success of the adversary's attack. In the lateral movement phase of the kill-chain adversarial model, attackers need to know how and where to move to act on their targets. In the red-teaming experiment by Cohen and Koike they deceived adversaries to attack the targeted system in a particular sequence from a networking perspective [58].

After deciding which components to simulate/dissimulate, we can apply one of Bell and Whaley's techniques discussed in [29]. We give an example of how each one of these techniques can be used in the following paragraphs.

- *Using Masking*—This has been used offensively where attackers hide potentially damaging scripts in the background of the page by matching the text color with the background color. When we apply hiding to software and services, we can hide the fact that we are running some specific services when we detect a probing activity. For example, when we receive an SSH connection request from a known bad IP address we can mask our SSHd demon and respond as if the service is not working or as if it is encountering an error.
- *Using Repackaging*—In several cases it might be easier to "repackage" data as something else. In computing, repackaging has long been used to attack computer users. The infamous cross-site scripting (XSS) attack uses this technique where

an attacker masks a dangerous post as harmless to steal the user's cookies when they view such post. Another example can be seen in the cross-site request forgery (XSRF) attacks where an adversary deceives a user into visiting some innocuous looking web pages that silently instruct the user's browser to engage in some unwanted activities. In addition, repackaging techniques are used by botnet Trojans that repackage themselves as anti-virus software to deceive users into installing them so an attacker can take control of their machines. From the defensive standpoint, a repackaging act can be seen in HoneyFiles, discussed above, that repackage themselves as normal files while acting internally as silent alarms to system administrators when accessed.

- *Using Dazzling*—This is considered to be the weakest form of dissimulation, where we confuse the targeted objects with others. An example of using dazzling can be seen in the "honeywords" proposal [24]. The scheme confuses each user's hashed password with an extra $(N - 1)$ hashes of other, similar, passwords dazzling an attacker who obtains the credentials database.
- *Using Mimicking*—In computing, phishing attacks are a traditional example of an unwanted deceiving login page mimicking a real website login. An attacker takes advantage of users by deceiving them into giving up their credentials by appearing as the real site. From a defensive perspective, we can apply mimicking to software and services by making our system mimic the responses of a different system, e.g., respond as if we are running a version of Windows XP while we are running Windows 7. This will waste attackers' resources in trying to exploit our Windows 7 machine thinking it is Windows XP, as well as increase the opportunity for discovery. This is seen in the work of Murphy et al. in operating system obfuscation [54].
- *Using Inventing*—Mimicking requires the results to look like something else; when this is not easy to achieve invention can be used instead. This technique has seen the most research in the application of deception to computer security defenses. Honeypots are one prominent example of inventing a number of nodes in an organizations with the goal of deceiving an attacker that they are real systems.
- *Using Decoying*—This technique is used to attract adversaries' attention away from the most valuable parts of a computer system. Honeypots are used, in some cases, to deceive attackers by showing that these systems are more vulnerable than other parts of the organization and therefore capture attackers' attention. This can be seen in the work of Carroll and Grosu [59].

After deciding which deceptive technique to use we need to analyze the patterns attackers perceive and then apply one or more of those techniques to achieve the desired reactions.

Deceit is an active manipulation of reality. We argue that reality can be manipulated in one of three general ways, as depicted in Fig. 5a. We can *manufacture* reality, *alter* reality, and/or *hide* reality. This can be applied to any one of the components we discussed above.

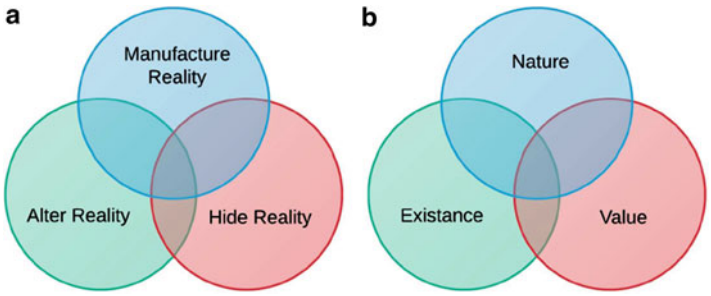

Fig. 5 Creating deceit. (**a**) Manipulation of reality. (**b**) Deception can be applied to the nature, existence and/or value of data

In addition, reality manipulation is not only to be applied to the existence of the data in our systems—it can be applied to two other features of the data. As represented in Fig. 5b, we can manipulate the reality with respect to the *existence* of data, *nature* of the data, and/or *value* of the data. The existence of the data can be manipulated not only for the present but also when the data has been created. This can be achieved for example with the manipulation of time stamps. With regard to the nature of the data, we can manipulate the size of the data, such as in the example of endless files, when and why the data has been created. The value of the data can also be manipulated. For example, log files are usually considered important data that adversaries try to delete to cover their tracks. Making a file appear as a log file will increase its value from the adversary's perspective.

At this step, it is crucial to specify exactly when the deception process should be activated. It is usually important that legitimate users' activity should not be hindered by the deceptive components. Optimally, the deception should only be activated in the case of malicious interactions. However, we recognize that this may not always be possible as the lines between legitimate and malicious activities might be blurry. We argue that there are many defensive measures that can apply some deceptive techniques in place of the traditional denial-based defenses that can make these tradeoffs.

4.2.3 Feedback Channels and Risks

Deception-based defenses are not a single one-time defensive measure, as is the case with many advanced computer defenses. It is essential to monitor these defenses, and more importantly measure the impact they have on attackers' perceptions and actions. This is step 5 in the deception framework. We recognize that if an attacker detects that he is being deceived, he can use this to his advantage to make a counter-deception reaction. To successfully monitor such activities we need to clearly identity the deception channels that can and should be used to monitor and measure any adversary's perceptions and actions.

In the sixth and final step before implementation and integration, we need to consider that deception may introduce some new risks for which organizations need to account. For example, the fact that adversaries can launch a counter-deception operation is a new risk that needs to be analyzed. In addition, an analysis needs to done on the effects of deception on normal users' activities. The defender needs to accurately identify potential risks associated with the use of such deceptive components and ensure that residual risks are accepted and well identified.

4.3 Implementing and Integrating Deception

Many deception-based mechanisms are implemented as a separate disjoint component from real production systems, as in the honeypot example. With the advancement of many detection techniques used by adversaries and malware, attackers can detect whether they are in real system or a "fake" system [25], and then change behavior accordingly, as we discussed earlier in this chapter. A successful deception operation needs to be integrated with the real operation. The honeywords proposal [24] is an example of this tight integration as there is no obvious way to distinguish between a real and a "fake" password.

4.4 Monitoring and Evaluating the Use of Deception

Identifying and monitoring the feedback channels is critical to the success of any deception operation/component. Hesketh discussed three general categories of signals that can be used to know whether a deception was successful or not [60]:

1. The target acts in the wrong time and/or place.
2. The target acts in a way that is wasteful of his resources.
3. The target delays acting or stop acting at all.

Defenders need to monitor all the feedback channels identified in step 5 of the framework. We note that there are usually three general outputs from the use of any deceptive components. The adversary might (1) *believe* it, where the defender usually sees one of the three signs of a successful deception highlighted above, (2) *suspect* it or (3) *disbelieve* it. When an attacker suspects that a deceptive component is being used, we should make the decision whether to increase the level of deception or stop the deceptive component to avoid exposure. Often deception can be enhanced by presenting more (and perhaps, true) information that makes the deception story more plausible. This can be included as a feedback loop in the framework. This observation should be analyzed by the defender to review his analysis of the attacker's biases, (i.e., step 3), and the methodology used to create the deceit (i.e., step 4). Furthermore, the deceiver might employ multiple levels of deception based on the interaction with the attacker during the attack.

When an attacker disbelieves the presented deceit we need to have an active monitoring and a detailed plan of action. This should be part the sixth step of planning in our framework where risks are assessed. In addition, during our discussions with security practitioners many have indicated that some attackers often act aggressively when they realize that they have been deceived. This can be one of the signals that is used during the monitoring stage to measure attackers' reaction of the deceptive component. In addition, this behavior can be used as one of the biases to be exploited by other deceptive mechanisms that may focus on deceiving the attacker about the *system's damage assessment*.

Acknowledgements The material in the chapter is derived from [29]. Portions of this work were supported by National Science Foundation Grant EAGER-1548114, by Northrop Grumman Corporation (NGCRC), and by sponsors of the Center for Education and Research in Information Assurance and Security (CERIAS).

References

1. Verizon, "Threats on the Horizon – The Rise of the Advanced Persistent Threat." http://www.verizonenterprise.com/DBIR/.
2. J. J. Yuill, *Defensive Computer-Security Deception Operations: Processes, Principles and Techniques*. PhD Dissertation, North Carolina State University, 2006.
3. B. Cheswick, "An Evening with Berferd in Which a Cracker is Lured, Endured, and Studied," in *Proceedings of Winter USENIX Conference*, (San Francisco), 1992.
4. C. P. Stoll, *The Cuckoo's Egg: Tracing a Spy Through the Maze of Computer Espionage*. Doubleday, 1989.
5. E. H. Spafford, "More than Passive Defense." http://goo.gl/5lwZup, 2011.
6. L. Spitzner, *Honeypots: Tracking Hackers*. Addison-Wesley Reading, 2003.
7. G. H. Kim and E. H. Spafford, "Experiences with Tripwire: Using Integrity Checkers for Intrusion Detection," tech. rep., Department of Computer, Purdue University, West Lafayette, IN, 1994.
8. D. Dagon, X. Qin, G. Gu, W. Lee, J. Grizzard, J. Levine, and H. Owen, "Honeystat: Local Worm Detection Using Honeypots," in *Recent Advances in Intrusion Detection*, pp. 39–58, Springer, 2004.
9. C. Fiedler, "Secure Your Database by Building HoneyPot Architecture Using a SQL Database Firewall." http://goo.gl/yr55Cp.
10. C. Mulliner, S. Liebergeld, and M. Lange, "Poster: Honeydroid-Creating a Smartphone Honeypot," in *IEEE Symposium on Security and Privacy*, 2011.
11. M. Wählisch, A. Vorbach, C. Keil, J. Schönfelder, T. C. Schmidt, and J. H. Schiller, "Design, Implementation, and Operation of a Mobile Honeypot," tech. rep., Cornell University Library, 2013.
12. C. Seifert, I. Welch, and P. Komisarczuk, "Honeyc: The Low Interaction Client Honeypot," *Proceedings of the 2007 NZCSRCS*, 2007.
13. K. G. Anagnostakis, S. Sidiroglou, P. Akritidis, K. Xinidis, E. Markatos, and A. D. Keromytis, "Detecting Targeted Attacks Using Shadow Honeypots," in *Proceedings of the 14th USENIX Security Symposium*, 2005.
14. D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, "Inside the Slammer Worm," *IEEE Security & Privacy*, vol. 1, no. 4, pp. 33–39, 2003.
15. T. Liston, "LaBrea: "Sticky" Honeypot and IDS." http://labrea.sourceforge.net/labrea-info.html, 2009.

16. F. Cohen, "The Deception Toolkit." http://www.all.net/dtk/, 1998.
17. N. Rowe, E. J. Custy, and B. T. Duong, "Defending Cyberspace with Fake Honeypots," *Journal of Computers*, vol. 2, no. 2, pp. 25–36, 2007.
18. T. Holz and F. Raynal, "Detecting Honeypots and Other Suspicious Environments," in *Information Assurance Workshop*, pp. 29–36, IEEE, 2005.
19. C. Kreibich and J. Crowcroft, "Honeycomb: Creating Intrusion Detection Signatures Using Honeypots," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 1, pp. 51–56, 2004.
20. D. Moore, C. Shannon, D. J. Brown, G. M. Voelker, and S. Savage, "Inferring Internet Denial-of-Service Activity," *ACM Transactions on Computer Systems (TOCS)*, vol. 24, no. 2, pp.115–139, 2006.
21. L. Spitzner, "Honeytokens: The Other Honeypot." http://www.symantec.com/connect/articles/honeytokens-other-honeypot, 2003.
22. J. J. Yuill, M. Zappe, D. Denning, and F. Feer, "Honeyfiles: Deceptive Files for Intrusion Detection," in *Information Assurance Workshop*, pp. 116–122, IEEE, 2004.
23. M. Bercovitch, M. Renford, L. Hasson, A. Shabtai, L. Rokach, and Y. Elovici, "HoneyGen: An Automated Honeytokens Generator," in *IEEE International Conference on Intelligence and Security Informatics (ISI'11)*, pp. 131–136, IEEE, 2011.
24. A. Juels and R. L. Rivest, "Honeywords: Making Password-Cracking Detectable," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, pp. 145–160, ACM, 2013.
25. X. Chen, J. Andersen, Z. M. Mao, M. Bailey, and J. Nazario, "Towards an Understanding of Anti-Virtualization and Anti-Debugging Behavior in Modern Malware," in *IEEE International Conference on Dependable Systems and Networks*, pp. 177–186, IEEE, 2008.
26. M. Sourour, B. Adel, and A. Tarek, "Ensuring Security-In-Depth Based on Heterogeneous Network Security Technologies," *International Journal of Information Security*, vol. 8, no. 4, pp. 233–246, 2009.
27. K. Heckman, "Active Cyber Network Defense with Denial and Deception." http://goo.gl/Typwi4, Mar. 2013.
28. R. V. Jones, *Reflections on Intelligence*. London: William Heinemann Ltd, 1989.
29. M. H. Almeshekah, *Using Deception to Enhance Security: A Taxonomy, Model and Novel Uses*. PhD thesis, Purdue University, 2015.
30. M. Harkins, "A New Security Architecture to Improve Business Agility," in *Managing Risk and Information Security*, pp. 87–102, Springer, 2013.
31. J. Boyd, "The Essence of Winning and Losing." http://www.danford.net/boyd/essence.htm, 1995.
32. E. M. Hutchins, M. J. Cloppert, and R. M. Amin, "Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains," *Leading Issues in Information Warfare & Security Research*, vol. 1, p. 80, 2011.
33. K. J. Higgins, "How Lockheed Martin's 'Kill Chain' Stopped SecurID Attack." http://goo.gl/r9ctmG, 2013.
34. F. Petitcolas, "La Cryptographie Militaire." http://goo.gl/e5IOj1.
35. K. D. Mitnick and W. L. Simon, *The Art of Deception: Controlling the Human Element of Security*. Wiley, 2003.
36. P. Vogt, F. Nentwich, N. Jovanovic, E. Kirda, C. Kruegel, and G. Vigna, "Cross-Site Scripting Prevention with Dynamic Data Tainting and Static Analysis," in *The 2007 Network and Distributed System Security Symposium (NDSS'07)*, 2007.
37. A. Barth, C. Jackson, and J. C. Mitchell, "Robust Defenses for Cross-Site Request Forgery," *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS'08)*, 2008.
38. O. W. A. S. P. (OWASP), "OWASP Top 10." http://owasptop10.googlecode.com/files/OWASPTop10-2013.pdf, 2013.

39. M. H. Almeshekah and E. H. Spafford, "Planning and Integrating Deception into Computer Security Defenses," in *New Security Paradigms Workshop (NSPW'14)*, (Victoria, BC, Canada), 2014.
40. J. B. Bell and B. Whaley, *Cheating and Deception*. Transaction Publishers New Brunswick, 1991.
41. M. Bennett and E. Waltz, *Counterdeception Principles and Applications for National Security*. Artech House, 2007.
42. J. R. Thompson, R. Hopf-Wichel, and R. E. Geiselman, "The Cognitive Bases of Intelligence Analysis," tech. rep., US Army Research Institute for the Behavioral and Social Sciences, 1984.
43. R. Jervis, *Deception and Misperception in International Politics*. Princeton University Press, 1976.
44. G. Hofstede, G. Hofstede, and M. Minkov, *Cultures and Organizations*. McGraw-Hill, 3rd ed., 2010.
45. D. Gus and D. Dorner, "Cultural Difference in Dynamic Decision-Making Strategies in a Non-lines, Time-delayed Task," *Cognitive Systems Research*, vol. 12, no. 3–4, pp. 365–376, 2011.
46. R. Godson and J. Wirtz, *Strategic Denial and Deception*. Transaction Publishers, 2002.
47. A. Tversky and D. Kahneman, "Judgment under Uncertainty: Heuristics and Biases.," *Science*, vol. 185, pp. 1124–31, Sept. 1974.
48. S. A. Sloman, "The Empirical Case for Two Systems of Reasoning," *Psychological Bulletin*, vol. 119, no. 1, pp. 3–22, 1996.
49. A. Tversky and D. Koehler, "Support Theory: A Nonextensional Representation of Subjective Probability.," *Psychological Review*, vol. 101, no. 4, p. 547, 1994.
50. A. Tversky and D. Kahneman, "Extensional Versus Intuitive Reasoning: The Conjunction Fallacy in Probability Judgment," *Psychological review*, vol. 90, no. 4, pp. 293–315, 1983.
51. L. Zhao and M. Mannan, "Explicit Authentication Response Considered Harmful," in *New Security Paradigms Workshop (NSPW '13)*, (New York, New York, USA), pp. 77–86, ACM Press, 2013.
52. R. S. Nickerson, "Confirmation Bias: A Ubiquitous Phenomenon in Many Guises," *Review of General Psychology*, vol. 2, pp. 175–220, June 1998.
53. C. Sample, "Applicability of Cultural Markers in Computer Network Attacks," in *12th European Conference on Information Warfare and Security*, (University of Jyvaskyla, Finland), pp. 361–369, 2013.
54. S. B. Murphy, J. T. McDonald, and R. F. Mills, "An Application of Deception in Cyberspace: Operating System Obfuscation," in *Proceedings of the 5th International Conference on Information Warfare and Security (ICIW 2010)*, pp. 241–249, 2010.
55. W. Wang, J. Bickford, I. Murynets, R. Subbaraman, A. G. Forte, and G. Singaraju, "Detecting Targeted Attacks by Multilayer Deception," *Journal of Cyber Security and Mobility*, vol. 2, no. 2, pp. 175–199, 2013.
56. X. Fu, *On Traffic Analysis Attacks and Countermeasures*. PhD Dissertation, Texas A & M University, 2005.
57. S. A. Hofmeyr, S. Forrest, and A. Somayaji, "Intrusion Detection Using Sequences of System Calls," *Journal of Computer Security*, vol. 6, no. 3, pp. 151–180, 1998.
58. F. Cohen and D. Koike, "Misleading Attackers with Deception," in *Proceedings from the 5th annual IEEE SMC Information Assurance Workshop*, pp. 30–37, IEEE, 2004.
59. T. E. Carroll and D. Grosu, "A Game Theoretic Investigation of Deception in Network Security," *Security and Communication Networks*, vol. 4, no. 10, pp. 1162–1172, 2011.
60. R. Hesketh, *Fortitude: The D-Day Deception Campaign*. Woodstock, NY: Overlook Hardcover, 2000.

http://www.springer.com/978-3-319-32697-9

Cyber Deception
Building the Scientific Foundation
Jajodia, S.; Subrahmanian, V.S.; Swarup, V.; Wang, C. (Eds.)
2016, X, 312 p. 111 illus., Hardcover
ISBN: 978-3-319-32697-9