CERIAS Tech Report 2014-7

Planning and Integrating Deception into Computer Security Defenses

by Mohammed H. Almeshekah and Eugene H. Spafford
Center for Education and Research
Information Assurance and Security
Purdue University, West Lafayette, IN 47907-2086

Planning and Integrating Deception into Computer Security Defenses*

Mohammed H. Almeshekah
CERIAS
Purdue University
West Lafayette, IN 47907-2086
malmeshe@purdue.edu

Eugene H. Spafford
CERIAS
Purdue University
West Lafayette, IN 47907-2086
spaf@purdue.edu

ABSTRACT

Deceptive techniques played a prominent role in many human conflicts throughout history. Digital conflicts are no different as the use of deception has found its way to computing since at least the 1980s. However, many computer defenses that uses deception were ad-hoc attempts to incorporate deceptive elements in them. In this paper, we present a model that can be used to plan and integrate deception in computer security defenses. We present an overview of why deception fundamentally works and what are the essential principles in using such techniques. We investigate the unique advantages deception-based mechanisms bring to traditional computer security defenses. Furthermore, we show how our model can be used to incorporate deception to many part of computer systems 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. We investigate these biases and discuss how can they be used by presenting a number of examples.

Categories and Subject Descriptors

H.1.2 [**User/Machine Systems**]: Software Psychology; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*Unauthorized access (e.g., hacking, phreaking)*

General Terms

Security, Human Factors

Keywords

Deception, Biases, Computer Security

1. INTRODUCTION

Deception has been in use for many millennia, perhaps for as long as life existed on planet earth. Plants, animals and insects have been using deceptive techniques as a mean for defense and survival. Humans are no exception to the use of deception. Illusionists use it to entertain us, con artists to cheat us, and military strategists to defend us. Digital realms are no different from the "real world" as deception has found its way into computerized systems. 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" [41] and the work of Spafford in his own lab [38]. Later, in the early 2000s, "honeypots" were introduced which employ deception for many applications [39].

In computer defenses, a broad range of deception techniques have been used for more than three decades [1]. Many defensive technologies have incorporated some use of deceptive techniques, such as honeypots. A proliferation of honey-prefixed methods was seen in the first decade of 2000s. We have argued for the case of using deception in computer defenses in [1] showing how such techniques fit within the bigger picture of information security, as depicted in 1.

As human beings, we are not good at detecting deception. In 39 different studies by Vrij, he found that the mean accuracy rate for college students to detect deception was only 57%, which is almost as poor as random choice [46]. This rate is slightly worse with law enforcement officers who scored a mean accuracy rate of 54% [46]. Whaley clearly stated in his seminal book "Stratagem: Deception and Surprise in War", which is the largest open source empirical analysis of the use of deception in conflicts, that "Indeed, this is a general finding of my study – that is, the deceiver is almost always successful regardless of the sophistication of his victim in the same art" [49].

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 [26]. He refers to deception as the "positive counterpart to security" that provides false clues to be fed to the opponents.

Offensively, many current common attacks use deceptive techniques as a cornerstone of their success. For example, phishing attacks always use two-level deceptive techniques.

*This is the submitted version to appear at the New Security Paradigm Workshop (NSPW'14). The final version can be obtained from the workshop webpage at www.nspw.org

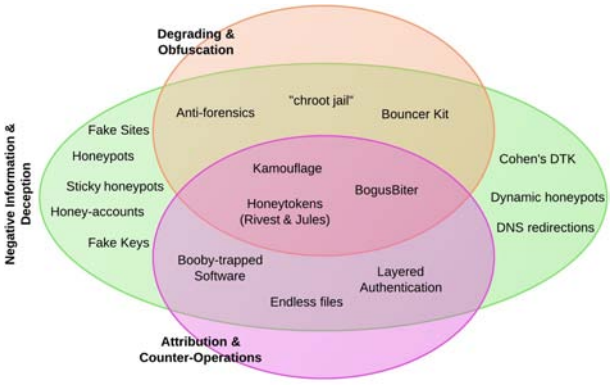

Figure 1: The Relationship between Deception and Other Computer Defense

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.

Despite the use of deception in computing, both offensively and defensively, little formal work has been done in investigating the concept of deception itself: how it works and why it is successful. In addition, to the best of our knowledge, there is no model detailing how deception can be planned and integrated into computer security defenses. This paper is intended to address these questions and discuss related issues that are worthy of future investigation and further research.

2. DECEPTION

A perception [that is not true] that is *intentionally* induced by actions of *other* entities is a deception [48]. One of the most widely accepted definitions of computer-security deception is the one by Yuill [52]:

> 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, as we will discuss later.

2.1 Deception Components

Bell and Whaley argue that deception always involves two steps; *dissimulation*, hiding the real, and *simulation*, showing the false [2]. They argue that these two concepts are "interdependent" and always work in tandem. Deception must involve these two together, even if only implicitly [2]. The act of hiding and showing can be applied on the (i) nature, (ii) existence and/or (iii) the value of targeted information, as we will discuss later in this paper. The authors also offered a taxonomy of deceptive techniques where they distinguished among three ways of dissimulating — masking, repackaging, and dazzling — and three ways of simulating — mimicking, inventing, and decoying. A brief discussion of each one of those is given below. Later in this paper, we show how each one of these methods can be used in computing.

With ***Masking*** the deceiver focuses on hiding the real by masking it such that it can not be detected. However, fully masking something to make it appear nonexistent can be challenging in many cases; this leads us to the use of repackaging. In ***repackaging*** the goal is to hide something to make it look like something else. If this also turns out to be challenging we can use the techniques of ***dazzling***, which is the weakest form of dissimulation, where we confuse the targeted objects with others making it difficult to distinguish the truth from the deceit.

When simulating, we can ***mimic*** something that is true and present it as the false. As an example, when a mantis insect hides by mimicking a stick appearance, it is also luring prey close enough to be seized. The challenge with mimicking is the need to present something that looks like something else already existing. When this is not possible we move to the second method of simulating; namely ***inventing***. Finally, when we cannot even invent things, we can ***decoy*** the target to attract his attention away from the most valuable parts to lesser important components.

Dunnigan and Nofi propose another taxonomy in [13]. Their taxonomy has the following deception groups: concealment, camouflage, false and planted information, lies, displays, ruses, demonstrations, feints, and insights. We found that each one of these categories either has a direct mapping to one of the Bell and Whaley categories above or is an example of one of these categories.

2.2 Deception Maxims

Bennett and Waltz discussed four deception maxims that are core to any investigation of the user of deception; namely *truth*, *denial*, *deceit* and *misdirection* [3]. In this section we discuss the relationships among these principles adding a fifth one that is equally as important; namely *confusion*.

Truth is the accurate perception of everything around the observed. Deception is an active act directed at manipulating such perception. For deception to succeed there must be an *accurate* perception that we are trying to manipulate [3]. Truth should constitute most of the information that is perceived by the adversary. Mitchell and Thompson articulate this principle stating that *"All deception works within the context of honesty"* [31]. Handel in [18] provide four rules of what truth should be presented to the target:

1. The deceiver should supply the target with correct low-grade information; i.e. "chicken-feed".
2. Correct information that is already known by the opponent should always be presented to target.
3. The deceiver should often pass correct information to the target when he can control its arrival time to be after it is of any use.

4. The deceiver might need to sacrifice some important information such that he can lure the target into believing some deceit that would have not been believed otherwise.

Handel summarizes his discussion with this quote *"The more one has a reputation of honesty – the easier it is to lie convincingly. Even more concisely: honest people/states can deceive the best"* [18].

Deceit. "All deception requires deceit" as said by Bennett and Waltz [3]. In other words, all deception requires the deceiver to intentionally lie about something to the target. Everyone lies in their daily lives. Ford cites some studies showing that 90% of Americans admitted that they lie about their feelings, income, sex, accomplishments, life and age [14]. There is a fundamental difference between simple lies and deception. The former focuses on only one side of the communicated message; namely the liar [12]. The latter adds to that the other side on the message, namely the receiver, and how this lie affects his perception and/or actions [12].

Denial, Misdirection and Confusion. There are three general way to manipulate a target's perception of truth and deceit with respect to deception. We can *deny* the target access to the truth and show him the deceit. When we cannot stop the truth from being observed we can *misdirect* the target's focus to the deceit. When we cannot influence the target's focus, we can *confuse* the target by presenting him with the truth and one or more plausible deceits.

3. DECEPTION-BASED DEFENSES

3.1 Background

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" [8]. He discussed how he interacted with an attacker in real time providing him with fabricated responses. The Deception Toolkit (DTK)[1], develop by Fred Cohen 1997 was one of the first publicly available tools to use deception for the purpose of computing defenses.

In 2003, Spitzner published his book on "honeypots" discussing how they can be used to enhance computer defenses [39]. Following on the idea of honeypots, "a component that provides its value by being attacked by an adversary" i.e. deceiving the attacker to interact with them, a proliferation of "honey-*" prefixed tools have been proposed. *Honeytokens* have been proposed by Spitzner [40] to refer to honeypots but at a smaller granularity. Kim and Spafford suggested the use of planted files, with interesting names, in the early version of Tripwire that should not be accessed by normal users, but will sound an alarm if they are accessed by intruders [28]. Later, Yuill et al coined the term *honeyfiles* for those files [51]. *HoneyGen* was also used to refer to tools that are used to generate honeytokens [4]. Most recently, a scheme named *honeywords* was proposed by Jules and Rivest to confuse attackers when they crack a stolen hashed password file [27].

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

Beyond the notion of "enticement" and traps used in honeypots, deception has been studied from other perspectives. For example, Cohen argues that using his Deception ToolKit (DTK), someone can deter attackers by confusing them and introducing risk on their side [10]. Rowe et al present a novel way of using honeypots for deterrence [35]. They enhanced the security of critical systems by making them look like a honeypot and therefore detered attackers from accessing them. Their approach stemmed from the development of anti-honeypot techniques that employ advanced methods to detect if the current system is a honeypot [23].

Moreover, a number of interesting uses of deception have been proposed to enhance the overall security of computer systems. Li and Schmitz proposed a framework to address phishing by using deceptive techniques [29]. The main idea in their framework is that when phishing is detected a number of "phoneytokens" will be sent to the phishing site. Banks can monitor these phoneytokens and then follow the money trail when phishers are detected stealing money. *BogusBiter* is a similar scheme proposed by Yue and Wang in [50]. The authors develop a client side add-on to the user's browser that intercepts username/password submissions when users override a phishing warning. Instead of stopping the submission they submit an additional $(N-1)$ username/password pairs based on the user's credentials. The scheme also requires the installation of a server-side component that analyzes username/password submissions and triggers a silent alarm when a "Bogus" credential has been submitted. Ormerod et al proposed a scheme that injects deceptive "fake" information to current botnet zombies for two main goals; dilute the "real" stolen information and trace end-users of a botnet's stolen information when it uses this "fake" information [33].

More recently, Zhao and Mannan use some deceptive techniques to limit the effectiveness of automated online password guessing [53]. They provide "fake" sessions to an adversary who is launching automated attacks while real users will detect the authentication outcome implicitly from the presented user data. In addition, Crane et al discuss the use of "Booby Trapping Software" — an active security defense mechanism for code-reuse attacks where deceptive techniques are used [11].

3.2 Advantages of Incorporating Deception in Computer Defenses

There is a fundamental difference between 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 intruders to take actions/inactions in ways that are advantageous to targeted systems. In other words, traditional security controls position themselves in response to attackers' actions while deception-based tools are positioned in anticipation of such actions.

Deception-based techniques provide significant advantages over traditional security controls. Currently, most defensive measures are playing **"whack-a-mole"** games with attackers. Whenever an attack surfaces, it is hit hard with preventive mechanisms. Eventually, persistent attackers find

a vulnerability that leads to a successful infiltration. This security posture is partially driven by the unquestioned assumption that **"hacking-back"** is unethical, while there is a difference between the act of "attacking back" and the act of deceiving attackers. With such behavior, attackers progressively learn about systems' defensive capabilities, with their continuous probing. Meanwhile, targeted systems learn nothing about these attempts, other than possibly inducing anxiety in the defenders. Many cases of multiple attempts that originate from the same entity are not successfully correlated.

We argue that, by intelligently using deceptive techniques, systems defenders can mislead and/or confuse attackers, thus enhancing defensive capabilities over time. By exploiting attackers' unquestioned trust of computer systems' responses system defenders can gain an edge and position themselves a step ahead of compromise attempts. In the face of increasing incidence of stealthy "advanced persistent threats" (APTs), deceptive techniques can enhance the overall security of computer systems.

Deception brings the following unique advantages to information system defenders:

1. **Increases the entropy of leaked information about targeted systems during compromise attempts.** When computer systems are targeted, the focus is usually only on protecting and defending them. With deception, extra defensive measures can be taken by feeding attackers false information that will, in addition to defending targeted systems, cause intruders to take wrong actions/inactions and/or draw incorrect conclusions. With the increased spread of stealthy attacks and government/corporate espionage threats these techniques can be valuable.

2. **Increases the information obtained from compromise attempts.** Many security controls are intended to create a boundary around computer systems that automatically stop any unauthorized access attempts. This is problematic as such boundaries are increasingly blurring, partly as a result of recent trends such as "consumerization"[2] [19]. 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. During this process, system defenders learn nothing about the intruders' targets. Ironically, this makes the task of defending a computer system potentially more difficult after every unsuccessful attack. We conjecture that incorporating deception-based techniques can enhance our understanding of compromise attempts involving probing activities and therefore better protect our systems over time.

3. **Give defenders an edge in the OODA loop race.** 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[5]. The victory in any tactical conflict requires executing this loop in a manner that is faster than your 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, because of lack of space and travel limitations.

[2]This term is widely used to refer to enterprises' employees bringing their own digital devices and using them to access the companies' resources, e.g., BYOD.

3.3 Kerckhoff's Principle and Deception

Deception always involves two basic steps, hiding the real and showing the false, as we discussed earlier. This, at first glance, contradicts the widely believed mischaracterization 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 principle was original stated with respect to cryptographic algorithms — not systems.

The misinterpretation has led many security practitioners to believe that any "obscurity" is ineffective, which 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 a defensive 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:

- **Truthful.** In such systems, the processes will always respond to any input with full "honesty." That is, the system's responses are always "trusted" and fully represent the internal state of the system. For example, when the user requests a network port, the system responds with either a real port number or denies the request giving the specific reason of such denial.

- **Deceptive (naively).** 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 responses used by the system, the deceptive 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 an attacker asks for a port, without carrying the appropriate permissions, this deceptive port is sent back.

- **Deceptive (intelligently).** In this case, the systems' "deceptive behavior" is indistinguishable from

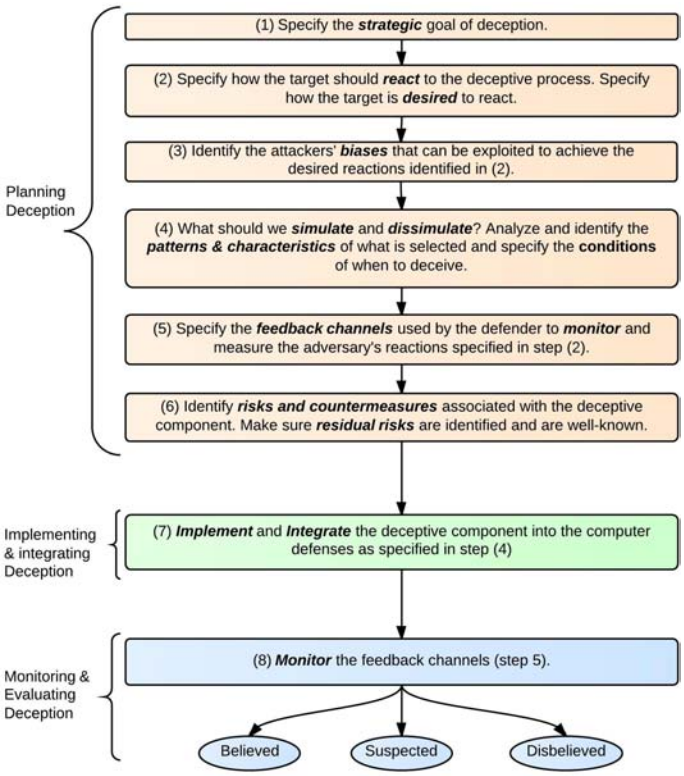

Figure 2: Deception Incorporation Model

the normal behavior even if the user has previously interacted with the system. For example, a system responds to an unauthorized port request identically to 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.

4. A MODEL FOR PLANNING AND INTEGRATING DECEPTION IN COMPUTER SECURITY DEFENSES

As discussed above there have been a number of interesting uses and applications for deception in computer security defenses. However, to the best of our knowledge there has been no discussion on how we can plan and integrate deception in our computer defenses. Moreover, we are not aware of any work that discuss why deception really works and what are biases and weaknesses deception-based mechanisms should exploit to influence the adversaries' perceptions and actions.

In this section we present our model of planning and incorporating deception into computer defenses and security. This model is based on the general deception model discussed by Bell and Whaley in [2]. There are three general phases of any deceptive component; namely planning, implementing and integrating, and finally monitoring and evaluating. In the following section we discuss each one of those phases in more details. The model is depicted in figure 2.

4.1 Planning Deception

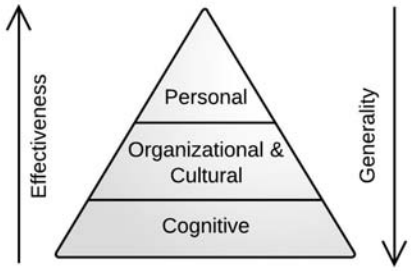

Figure 3: Adversaries' Biases

There are six essential steps to planning a successful deception-based defense component. The first, and often neglected, step is specifying exactly what are the ***strategic goals*** the defender wants to achieve. Simply augmenting computer systems with honey-like components, such as honeypots and honeyfiles, will give 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. To give 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 target of the deception process, we need to specify how the target should react to the deceptive process. This determination is critical to the long-term success of any deceptive process. To give an example, Zhao and Mannan [53] deceive attackers launching online guessing attacks into believing that they have found a correct username and password. The strategic goal of the 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 cease the guessing attack and report to its command and control that a successful username/password pair has been found. In consideration of the second reaction alternative we might need to maintain the username/password pair of the fake account and keep that account information consistent for future targeting.

Part of this second step is to specify how we desire the target to react so 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, we can cause an artificial network disconnection that will force the target to login again.

4.1.1 Adversaries' Biases

A bias refers to

"an inclination to judge others or interpret situa-

tions based on a personal and oftentimes unreasonable point of view" [3]

Biases are a cornerstone component to the success of any deception-based mechanisms. As we discussed above, 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 he is more inclined to reject it instead of believing it, or at least raise his suspicions about the possibility of currently being deceived. A successful deception should ***exploit*** a bias in the attacker's perception and provide him with alternative(s), plausible information other than the truth. When the defender determines the strategic goal of his 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.

Thompson et al discussed four major group of biases intelligence analysts need to be aware of; personal biases, cultural biases, organizational biases and cognitive biases [42]. It can be seen in figure 3 that the more specific biases that are exploited the less general are the deceptive components, e.g. personal biases that can be exploited for a specific target might not apply to other adversaries who attack a system. Alternatively, more specific choices of biases supports more accurate deceptive components. This is 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 will discuss each one of these classes of biases and give a number of examples of how can they be exploited to enhance computer security using deception.

Personal Biases. Personal biases are those biases that originated from either first-hand experiences and/or personal traits, as discussed by Jervis in [25]. These biases can be helpful in designing deceptive component/operation, however, they are (i) harder to obtain and know as they require specific knowledge of potential adversaries and (ii) they make deceptive components less applicable to a wider range of adversaries while becoming more powerful against specific attackers. Personal biases have been used and exploited in traditional deceptions in war, such as exploiting the arrogance of Hitler's administration in World War II as part of Operation Fortitude [3].

Cultural Biases. Hofstede refers to cultural biases as the *"software of the mind"* [22]. 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 the subjects were presented with the same information [17]. Some studies found relationships between the type of computer attacks and the culture/country from which the attack originated [36]. Hofstede discuss six main dimensions of cultures and provided some quantitative values of these in (geerte-hofstede.com) where he associated different behavior that correlates with his measurements. 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"* [16].

Organizational Biases. Organizational biases are of importance when designing deception for an attacker within a bureaucratic environment [3]. 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. These biases can be valuable in understanding targets for deception, exploiting their weaknesses to make important information marked as less important while any deceit is passed to higher levels. The uneven distribution of information led to uneven perception and failure to anticipate Pearl Harbor by the United States [3].

Cognitive Biases. Cognitive biases are common among all humans across all different cultural, personal and organizational differences. They represent the "innate ways that human beings perceive, recall, and process information [3]. 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 more generally.

Tversky and Kahneman proposed three general heuristics our minds use to reduce a complex task to a simpler judgment decision, especially under condition of uncertainty, thus leading to some predictable biases [43]. These are; representativesness, 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"* [43]. 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 heuristic is a bias that cause us to make estimations closer to the initial values we have been provided with than is otherwise warranted.

Solman presented a discussion on two reasoning systems postulated to be common in humans: associative (system 1) and rule-based (system 2) [37]. 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 [45]. 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 to the errors often found in probability assessments associated with the *"conjunction fallacy"* [44]. 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.

In computing, the bias in the conjunction fallacy can be exploited by deceivers by presenting the deception story as a conjunction of multiple detailed components. For example, if a deceiver wants to misinform an attacker probing her 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 due to some scheduled maintenance. Please visit us back in three hours" is more plausible than simply saying "The network is down".

4.1.2 Creating the Deception Story

After analyzing the target's biases that can be exploited to deceive them, the deceiver needs to decide exactly what components to simulate/dissimulate; namely step 4 of the model in figure 2.

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

System's Decisions. We can apply deception to the different decisions any computer system makes. As an example, Zhao and Mannan in [53] apply deception at the system's authentication decision where they discuss that negative authentication decisions leak some data and can be harmful. They discuss a system where adversaries can be deceived by giving them access to "fake" accounts in case of online guessing attacks. Another system's decision we can use concerns firewalls. Traditionally, we add firewall rules that prevent specific IP address from interacting with our systems after detecting that they are sources of some attacks against us. We consider this another form of data leakage in accordance with the discussion of Zhao and Mannan in [53]. We can apply deception to such decisions by presenting adversaries with other plausible responses 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 [24]. Providing fake systems and services has been the main focus of honeypot-based mechanisms. Honeypots are intended to provide attackers with a number of fake systems running fake services. Moreover, we can use deception to mask the identities our current existing software/services. Murphy et al. studied the efficacy of using operating system obfuscation and recommended the use of these tools for the Air Force computer defenses [32].

System's Internal and Public Data. A honeyfile [51] 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. This can also be applied to administrative data that are used to make decisions and/or monitor the system's activities. An example of that can be seen in the *honeyword* proposal [27]. 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 [47]. 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 the system are considered as one source of information leakage. For example, traffic flow analysis has long been studied as a mean for attackers to deduce information [15]. Additionally, a system's activity has been used as a means of distinguishing between a "fake" and a real system [7]. We can intelligently inject some data about activities into our system to influence the attacker's perception and, therefore, his 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 is leaking information that these vulnerabilities have been 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 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 he causes 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. 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 [30] that are intended to slow the adversary's probing activity. Also, tarpits have been

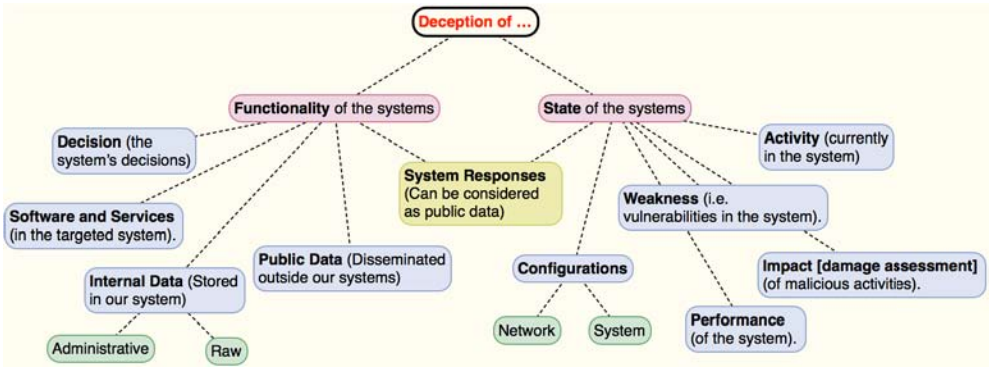

Figure 4: Computer Systems Components that can be Used to Integrate Deception Into

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 [21].

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, the attacker needs to know how and where to move to act on the targets. In a red-teaming experiment, Cohen and Koike showed how they could induce adversaries to attack the targeted system in a particular sequence from a networking perspective using deception [9].

After deciding which component to simulate/dissimulate, we can apply one of Bell and Whaley's techniques discussed earlier. We give an example of how each one of these techniques can be used in the following paragraphs. We should point out that, in any deception the act of simulation and dissimulation must be combined as we discussed earlier.

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. Hiding is only considered deception if it is an act of deceit, otherwise it is considered denial. When we apply hiding to software and services, we can hide the fact that we are running some specific services when we have detected 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 that the service is not working or is encountering an error.

Using Repackaging – In several cases it might be easier to "repackage" data as something else. We can apply repacking to a system's internal raw data as in the use of HoneyFiles [51]. These files are repackaged as normal files while internally they act 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. This can be used for a system's internal administrative data as in the recent *"honeywords"* proposal by Rivest and Jules [27]. Their scheme proposes confusing a users's hashed password with another (N 1) hashes of other, similar, passwords where they are dazzling the attacker who obtains the credentials database.

Using Mimicking – When we hide the real we necessarily show the false, even if only implicitly by showing "nothing." The first method of simulation techniques is to show the false while mimicking something true to gain an advantage. In computing, phishing attacks are a traditional example of evil deceiving act by mimicking a real website. The attackers can take advantage of users by deceiving them into giving up their credentials by appearing as a 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 a version of Windows XP while we are running Windows 7. This will waste the attackers' resource in trying to exploit our Windows 7 machine thinking it is Windows XP, as well as heighten the opportunity for discovery.

Using Inventing – Mimicking requires the results look like something else; when this is not easy to achieve invention can be used instead. This deception technique has seen the most research in the application of deception to computer security defenses. Honeypots are a prominent example of inventing a number of nodes in an organizations with the goal of deceiving the attacker that they are the real systems.

Using Decoying – This technique is used to attract adversaries' attention away from the most valuable parts of the 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 in [6].

After deciding which deceptive technique to use we need to analyze the characteristics and patterns the attacker perceives and then apply one or more of the techniques above to achieve the desired reactions. Deceit is an active manipulation of reality. We argue that reality can be manipulated in

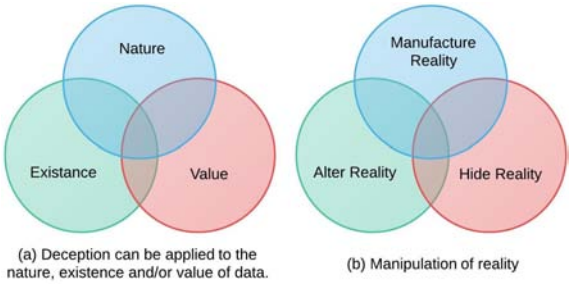

Figure 5: Creating Deceit

one of three general ways, as depicted in figure 5-b. We can *manufacture* reality, *alter* reality and/or *hide* reality. This can be applied to any one of the components we discussed in the previous section.

In addition, reality manipulation is not to only be applied on the existence of the data in our systems; in fact is can be applied to two other features of the data. As represented in figure 5-a, 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 just 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 in [38], when and why the data has been created and so on. The value of the data can also be manipulated. For example, log files are usually considered important data that usually adversaries try to delete to cover the 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 should the deception process be activated. It is vital that normal legitimate user's 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 actives might be blur. With that being said, we argue that there are many defensive measure that can apply some deceptive techniques in in place of the traditional denial-based defenses.

4.1.3 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 any attackers' perceptions and actions. We recognize that if an attacker detects that he is being deceived, he can use this to his advantage to launch a counter-deception operation. 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.

Finally, 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.

4.2 Implementing and Integrating Deception

Many deception-based mechanisms are implemented as a separate disjoint component from the 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 [7], and then change behavior accordingly. A successful deception operations needs to be integrated with the real operation. The honeywords proposal is one of the examples of this tight integration as there is no obvious way to distinguish between a real and a "fake" password [27].

4.3 Monitoring and Evaluating the Use of Deception

As we discussed earlier, monitoring feedback-channels is a vital step 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 [20]:

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 model. We note that there are usually three general outputs from the use of any deceptive components. The adversary might (i) *believe* it, where the defender usually sees one of the three signs of a successful deception above, (ii) *suspect* it or (iii) *disbelieve* it. When an attacker suspects a deceptive component we should make the decision whether to increase the level of deception or withdraw the deceptive component to avoid exposure. Often deception can be enhanced by presenting more, and probably, true information that makes the deception story more plausible. This can be used as a feedback loop in the model. This observation should be analyzed by the defender to review his analysis of the attacker's biases, step (3), and the methodology used to create the deceit in step (4). Furthermore, the deceiver might have 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 active monitoring and a detailed plan of actions. This should be part of step (6) of planning in our model where risks are assessed. In addition, during discussions with many security practitioners they pointed out 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 the attackers' perception to the deceptive component. In addition, this behavior can be

used as one of the biases to exploit by other deceptive mechanisms that may focus on deceiving the attacker about the *system's damage assessment*, as discussed in section 4.1.2 above.

5. CONCLUSIONS

Deception-based defenses are powerful 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 false alternative to the hidden truth. These mechanisms give defenders the ability to learn more about the attackers, reduce indirect information leakages in their systems and give them an advantageous step with regard to their defenses.

In this paper, we presented a novel model to plan and integrate deception for defender to use as part of the their computer defenses. Also, we discussed how they can monitor and evaluate the success of such mechanisms. In addition, we explain how defenders integrate deception in their computer security defenses and how they can create plausible alternatives to reality, thus misinforming the attackers and wasting their resources.

6. ACKNOWLEDGMENTS

The authors would like to extends their thanks to Mikhail Atallah, Joshua Corman, and George Kurtz for their time and the valuable discussion and insightful ideas they provided.

7. REFERENCES

[1] M. H. Almeshekah and E. H. Spafford. The Case of Using Negative (Deceiving) Information in Data Protection. Academic Conferences and Publishing International, 2014.

[2] J. B. Bell and B. Whaley. *Cheating and deception*. Transaction Publishers New Brunswick, 1991.

[3] M. Bennett and E. Waltz. *Counterdeception Principles and Applications for National Security*. Artech House, 2007.

[4] M. Bercovitch, M. Renford, L. Hasson, A. Shabtai, L. Rokach, and Y. Elovici. HoneyGen: An automated honeytokens generator. In *Intelligence and Security Informatics (ISI), 2011 IEEE International Conference on*, pages 131–136. IEEE, 2011.

[5] J. Boyd. The Essence of Winning and Losing, 1995.

[6] T. E. Carroll and D. Grosu. A game theoretic investigation of deception in network security. *Security and Communication Networks*, 4(10):1162–1172, 2011.

[7] 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 *Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on*, pages 177–186. IEEE, 2008.

[8] B. Cheswick. An Evening with Berferd in which a cracker is Lured, Endured, and Studied. In *Proc. Winter USENIX Conference, San Francisco*, 1992.

[9] F. Cohen and D. Koike. Misleading attackers with deception. In *information assurance workshop, 2004. Proceedings from the fifth annual IEEE SMC*, pages 30–37. IEEE, 2004.

[10] F. Cohen and Others. The deception toolkit. *Risks Digest*, 19, 1998.

[11] S. Crane, P. Larsen, S. Brunthaler, and M. Franz. Booby trapping software. *Proceedings of the 2013 workshop on New security paradigms workshop - NSPW '13*, pages 95–106, 2013.

[12] D. Daniel and K. Herbig. Propositions on military deception. *The Journal of Strategic Studies*, 1(5):155–177, 1982.

[13] J. F. Dunnigan and A. A. Nofi. *Victory and Deceit: Deception and Trickery at War*. Writers Club Press, 2001.

[14] C. Ford. *Lies! Lies!! Lies!!!: The Psychology of Deceit*. American Psychiatric Publishing, Inc., 1st edition, 1999.

[15] X. Fu. *On Traffic Analysis Attacks and Countermeasures*. Phd thesis, Texas A & M University, 2005.

[16] R. Godson and J. Wirtz. *Strategic Denial and Deception*. Transaction Publishers, 2002.

[17] D. Gus and D. Dorner. Cultural Difference in Dynamic Decision-Making Strategies in a Non-lines, Time-delayed Task. *Cognitive Systems Research*, 12(3-4):365–376, 2011.

[18] M. Handel. *War, Strategy and Intelligence*. Routledge, London, UK, 1989.

[19] M. Harkins. A New Security Architecture to Improve Business Agility. In *Managing Risk and Information Security*, pages 87–102. Springer, 2013.

[20] R. Hesketh. *Fortitude: The D-Day Deception Campaign*. Overlook Hardcover, Woodstock, NY, 2000.

[21] S. A. Hofmeyr, S. Forrest, and A. Somayaji. Intrusion detection using sequences of system calls. *Journal of computer security*, 6(3):151–180, 1998.

[22] G. Hofstede, G. Hofstede, and M. Minkov. *Cultures and Organizations*. McGraw-Hill, 3rd editio edition, 2010.

[23] T. Holz and F. Raynal. Detecting honeypots and other suspicious environments. In *Information Assurance Workshop, 2005. IAW'05. Proceedings from the Sixth Annual IEEE SMC*, pages 29–36. IEEE, 2005.

[24] 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*, 1:80, 2011.

[25] R. Jervis. *Deception and Misperception in International Politics*. Princeton University Press, 1976.

[26] R. V. Jones. *Reflections on intelligence*. William Heinemann Ltd, London, 1989.

[27] A. Juels and R. L. Rivest. Honeywords: making password-cracking detectable. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 145–160. ACM, 2013.

[28] G. Kim and E. Spafford. Experiences with tripwire: Using integrity checkers for intrusion detection. Technical report, Purdue University, West Lafayette, IN, 1994.

[29] S. Li and R. Schmitz. *A novel anti-phishing framework based on honeypots*. IEEE, 2009.

[30] T. Liston. LaBrea:"Sticky" Honeypot and IDS, 2009.

[31] R. Mitchell and N. Thompson. *Deception: Perspectives on Human and Nonhuman Deceit*. State University of New York Press, 1985.

[32] 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)*, pages 241–249, 2010.

[33] T. Ormerod, L. Wang, M. Debbabi, A. Youssef, H. Binsalleeh, A. Boukhtouta, and P. Sinha. Defaming botnet toolkits: A bottom-up approach to mitigating the threat. In *Emerging Security Information Systems and Technologies (SECURWARE), 2010 Fourth International Conference on*, pages 195–200. IEEE, 2010.

[34] F. Petitcolas. la cryptographie militaire.

[35] N. C. Rowe, E. J. Custy, and B. T. Duong. Defending cyberspace with fake honeypots. *Journal of Computers*, 2(2):25–36, 2007.

[36] C. Sample. Applicability of Cultural Markers in Computer Network Attacks. In *12th European Conference on Information Warfare and Security*, pages 361–369, University of Jyvaskyla, Finland, 2013.

[37] S. Sloman. The empirical case for two systems of reasoning. *Psychological bulletin*, 119(1):3–22, 1996.

[38] E. Spafford. More than Passive Defense.

[39] L. Spitzner. *Honeypots: tracking hackers*, volume 1. Addison-Wesley Reading, 2003.

[40] L. Spitzner. Honeytokens: The other honeypot, 2003.

[41] C. P. Stoll. The Cuckoo's Egg: Tracing a Spy Through the Maze of Computer Espionage, 1989.

[42] J. Thompson, R. Hopf-Wichel, and R. Geiselman. The Cognitive Bases of Intelligence Analysis. *U. S. ARMY RESEARCH INSTITUTE FOR THE BEHAVIORAL AND SOCIAL SCIENCES*, 2:2–9, 1984.

[43] A. Tversky and D. Kahneman. Judgment under Uncertainty: Heuristics and Biases. *Science (New York, N.Y.)*, 185(4157):1124–31, Sept. 1974.

[44] A. Tversky and D. Kahneman. Extensional versus Intuitive Reasoning: The Conjunction Fallacy in Probability Judgment. *Psychological review*, 90(4):293–315, 1983.

[45] A. Tversky and D. Koehler. Support theory: a nonextensional representation of subjective probability. *Psychological Review*, 101(4):547, 1994.

[46] A. Vrij. *Detecting lies and deceit: pitfalls and opportunities. Wiley Series in the Psychology of Crime, Policing and Law*. 2008.

[47] 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*, 2:175–199.

[48] B. Whaley. Toward a general theory of deception. *The Journal of Strategic Studies*, 5(1):178–192, 1982.

[49] B. Whaley. *Stratagem: Deception and Surprise in War*, volume 11. Artech House Information Warfare Library, 2007.

[50] C. Yue and H. Wang. BogusBiter: A transparent protection against phishing attacks. *ACM Transactions on Internet Technology (TOIT)*, 10(2):6, 2010.

[51] J. Yuill, M. Zappe, D. Denning, and F. Feer. Honeyfiles: deceptive files for intrusion detection. In *Information Assurance Workshop, 2004. Proceedings from the Fifth Annual IEEE SMC*, pages 116–122. IEEE, 2004.

[52] J. J. Yuill. *Defensive Computer-Security Deception Operations: Processes, Principles and Techniques*. ProQuest, 2006.

[53] L. Zhao and M. Mannan. Explicit authentication response considered harmful. *Proceedings of the 2013 workshop on New security paradigms workshop - NSPW '13*, pages 77–86, 2013.