Deception for Computer Security Defense

Jim Yuill, Fred Feer, Dorothy Denning, Bowyer Bell
January, 2004

Technical Report

Prepared for the U.S. Department of Defense

Abstract: Deception is an appealing means for computer security defense: hackers rely heavily upon network data, and there is great potential for using network data to deceive them. Further, deception can be used to attack hackers' decision-making processes; thus deception provides an offensive security-measure-- something computer security defenders sorely lack. This paper explains how deception operations can be designed and developed, for computer security defense. Deception processes, principles and techniques are presented. They are based on the underlying nature of deception, and the extensive military deception-literature. An extensive survey was made of computer security deceptions, and they provide examples. A novel intrusion-detection device named *honeyfiles* is presented. Honeyfiles are alarmed bait files that are intended for hackers to access. Honeyfiles were tested by deploying them on a honeynet. The use of honeynets for testing security devices is also addressed. The honeyfiles and the honeynet served to test the paper's deception principles.

Dedication

This report is dedicated to the memory of our fond friend and co-author,
Dr. Bowyer Bell.

Preface to the 2022 distribution: This report was written in 2004, but the authors did not make it publicly available until 2022. Though, copies were on the Internet during those years.

The report focuses on enduring deception principles that are independent of passing technologies. So, we expect most of the present report will still be relevant, now 18 years later.

The lasting nature of deception principles is corroborated by military history:

The basic principles and objectives of reinforcing the desires and perceptions of the deceived will not change, since human nature and the psychological mechanism of human perception are ever the same. In terms of its forms and the means employed, deception will, like war itself, change as new weapons and technologies appear. [Han85]

Acknowledgements: This paper was made possible by funding from The Office of Net Assessment, in The Office of the Secretary of Defense. Our late colleague Dr. Bowyer Bell initiated the project and obtained its funding; we remain thankful. For the authors J. Yuill, D. Denning and F. Feer, their collaboration was brought about by USMC Colonel G.I. Wilson. His support and encouragement are much appreciated.

Table of Contents

1 Introduction

Deception is an appealing means for computer security defense, as it pits the defender's strengths against the hacker's weaknesses. Hackers rely heavily, if not exclusively, on a single source of information—network data. The data is easily manipulated, and the hacker is highly vulnerable to deception. The defender has physical control of the network, and he knows the network well.[1] When a hacker first arrives at the network, he must learn about the network by investigating it. The hacker's investigation, combined with his initial naiveté, create an unavoidable and predictable conduit for deception. The defender can use this conduit to supply the hacker with falsehood and thereby attack his decision-making process.

This paper explains how deception can be used to advantage in computer security defense. It describes the process followed in deception operations, and it describes principles and techniques for developing and conducting deception operations. The intended audience is computer security professionals, though experienced computer users will be able to understand most of the paper.

Deception is an integral part of human nature and experience. There are legitimate, even necessary, reasons for deceiving others, as in sports. Deception is a frequent theme of history, literature and drama, and consumers constantly exercise counter-deception. However, few people use deception in the calculated manner needed for computer security. As military deception reveals, effectively deceiving an adversary is a job skill, though fortunately, one that can be learned.

The principles of military deception are very applicable to computer security deception. The principles are well documented in the military deception-literature, and they are based on millennia of experience and thought. This paper adapts the principles of military deception to computer security deception. The paper also draws upon the intelligence deception-literature. (The primary literature sources are summarized in the section *Sources* on page 80.) In addition, one of this paper's authors has extensive experience in both military and intelligence deception.

The background research for this paper includes an extensive survey of the current uses of deception for computer security. The survey provides many of paper's principles and examples. However, as possible, the paper focuses on deception principles that are of enduring use, and independent of current technologies. For instance, honeypots are the most widely used deception today. The paper uses honeypots to illustrate principles, but they are not its focus.

To test the deception process and principles, we developed an intrusion-detection system that uses deception. To test the intrusion-detection system, we deployed it on a deceptive network, and observed how hackers interact with it. These two case studies are presented in the context of the paper's deception processes and principles.

Although appealing, deception is little used for computer security defense. One reason is that few computer security professionals have had an opportunity to develop deception skills. This paper seeks to provide an understanding of deception, as a starting point for using deception and for developing skills. Our hypothesis is that deception can serve a greater, and even indispensable, role in computer security. After years of research and development, computer

[1] Unless stated otherwise, this paper's masculine pronouns refer to both men and women.

security remains an error-prone task and, in some respects, a loosing battle. Deception offers an opportunity to exploit the hacker's weaknesses and to attack his decision-making process. Deception is not useful in all situations. However, even when working with a small security budget, deception can be among the most cost-effective means for securing valuable assets. Conversely, not using deception may be an indication of over estimating the effectiveness of conventional security measures.

A summary of the paper follows:

Chapter 2 describes the deception-operation process. Deception operations vary greatly in the purposes they serve and how they are conducted. However, there is a basic process that is followed in almost all deception operations.

Chapters 3 and 4 explain how to develop a deception operation. Chapter 3 describes deception-operation planning, and it includes setting the deception objective and analyzing the deception target. The *deception objective* is a statement of what action the deception target is to take, and how that action will be exploited, to the deceivers advantage. For the deception planner, the beginning of wisdom is to understand that the goal is not trickery-- it is the deception objective. The target action is much easier to induce if it is something the target already plans to do or wants to do. Deception is aimed at the target's mind and actions, so it is essential to know how the hacker thinks and works. Understanding the target's intelligence systems is especially important, as they can be used to communicate the deception to him.

Chapters 4 describes how to build the deception. The *desired perception* is what the target must believe in order for him to take the intended action. The *deception story* is what the deception operation will portray, to create the desired perception. Principles and techniques are presented for designing and implementing the deception story. For instance, the principle of minimizing falsehood makes the deception story easier to implement, and the deception more difficult to detect. One technique for minimizing falsehood is to weave the deception story into the truth, e.g., the real network. Real systems that are part of the story will not have to be impersonated, and the network itself will help to corroborate the story. *Feedback* about the target is needed in order to know whether the target has received the deception story, interpreted it as intended, and taken the intended action. For computer security deceptions, feedback can be very effective if it is designed into the deception operation, e.g., keystroke loggers.

Chapter 5 shows many uses of deception for computer security. Deception can provide the standard defensive measures of: prevention, detection, response, intelligence, and deterrence. Deception can also counter specific steps in the hacking process, such as scanning and gaining access. Our survey of computer security deceptions provides examples.

One of deception's most promising uses is in attacking the hacker's decision-making process. Another promising use is for internal network security. A long-standing computer security problem is having to use weak internal security because added security impairs productivity. Some of the deception techniques that are shown can significantly increase internal security without adversely affecting normal operations. Chapter 6 briefly addresses the means of deception.

Chapters 7 and 8 present two deceptions that were developed to test the earlier chapters' processes and principles. A novel intrusion-detection device named *honeyfiles* was developed. Honeyfiles are bait files intended for hackers to access. The files reside on a file server, and the

server sends an alarm when a honeyfile is accessed. For example, a honeyfile named *passwords.txt* would be enticing to most hackers. The file server's end users create honeyfiles, and the end users receive the honeyfile's alarms. Honeyfiles can increase a network's internal security without adversely affecting normal operations.

To test the honeyfiles, they were deployed on a fake network called a honeynet. There, hackers' use of honeyfiles was observed. This experiment, itself, represents a type of deception with wide application: using honeynets to test a computer security device. The honeynet's hackers can be very useful for finding the developers' faulty and overlooked assumptions. Usually, it is difficult for developers to see their own bad assumptions.

2 An overview of deception operations

In this chapter, basic deception concepts and terminology are presented, followed by a description of the deception-operation process. The chapter also introduces the process for developing deception operations.

2.1 Basic concepts and terminology

Computer security deception is defined as being those actions taken to deliberately mislead hackers and to thereby cause them to take (or not take) specific actions that aid computer security.[2] Deception aims to mislead the hacker into a predictable course of action or inaction that can be exploited [Dew89]. Tricking the hacker, and making him think a certain way, is important only as a step toward getting him to make the decision that will result in the desired action [JDD96]. Thoughts without action are of little computer security value.

The scope of this paper is deception for computer security defense. It focuses on the tactical use of deception for a computer network. Legal issues regarding the use of deception are not addressed, nor is counter deception, i.e., detecting hackers' deceptions. Deception for counter attacks (i.e., hacking hackers) is also out of scope. However, using deception to thwart, or attack, the hacker's decision-making process is one of the paper's primary topics.

Some of the paper's computer and deception terms are defined as:

- **CND**: computer network defense
- **deception planner**, or **planner**: the person who plans, develops and carries out the deception operation
- **deception operation**: the planned development and deployment of a deception-based computer security measure
- **target**: the person the deception operation seeks to deceive
- **intelligence**: information and knowledge obtained through observation, investigation, analysis, or understanding. For example, hackers use port scanning to *collect intelligence*, and the *intelligence sources* are the network's computers.
- **ruse**: a trick designed to deceive
- **channel**: a means of communicating information

Two types of deception are *hiding the real* and *showing the false* [BW82]. Although most deceptions involve both hiding and showing, a deception can often be categorized as hiding or showing, according to the planners primary intent. Further, most hiding involves showing something else instead. Also, showing the false usually involves hiding what is real. Not all hiding is deceptive. We consider hiding to be deceptive if its intention is misrepresentation.[3] Hiding that conceals, but does not misrepresent, is called *denial*. For example, keeping the curtains drawn while out of town is usually not deception, as the curtains simply conceal the home's interior. However, if the house contained something unusual, like a fugitive, then drawing the curtains would be deceptive if the intent is to make the house appear unremarkable to law enforcement personnel, i.e., like it does not contain the fugitive.

There is wide variability among deception operations. Deception operations vary in the

[2] This definition is adapted from the U.S. DoD definition of military deception [JDD96].
[3] Thanks to Larry Colon, at Symantec, for this distinction.

purposes they serve, the networks on which they are used, and the different types of hackers they target. Deception stories are only limited by the planners' imagination, and the stories can be implemented by a wide array of technology. Some deceptions' development and use are simple and predictable. For instance, camouflaged clothing is readily available and its effects are predictable. Similarly, hacker tools for fingerprinting systems (e.g., NMAP) might be easily and predictably deceived. Other deceptions are complex and uncertain. For two years the Allies invested enormous effort in hiding the nature of Operation Overlord, the invasion of France, offering false options, using hundreds of channels for thousands of ruses. Similarly, a honeynet can be large, and there can be many servers with extensive false content.

Although deception operations vary widely, there are processes and principles that are applicable to many, or even all, deception operations. As a theoretician of military deception has observed:

> The basic principles and objectives of reinforcing the desires and perceptions of the deceived will not change, since human nature and the psychological mechanism of human perception are ever the same. In terms of its forms and the means employed, deception will, like war itself, change as new weapons and technologies appear. [Han85]

As mentioned, this paper seeks to focus on enduring deception principles that are independent of passing technologies.

2.2 The deception-operation process

The **deception-operation process** involves complex adversarial relationships and complex engineering systems. This chapter gives an overview of the process, and subsequent chapters describe the process' major components. These descriptions of deception operations are *process models*. The models explain how deception operations work, and they provide a framework for planning and conducting deception operations. Where the context is clear, we'll refer to the deception-operation process as simply *the deception process*, or *the process*.

Although the process of deception operations can be complex, there is a **basic deception-process** that is followed in almost all operations. This basic deception-process is shown in Figure 1, and it is described below. (For abbreviation, the basic deception-process is referred to as *the basic process*.) The basic process consists of the major process-components used in deception operations, and other complicating effects are omitted.

Due to the complexity of deception operations, the basic process is a simplified conceptual model. It is not meant to provide a complete description of all deception operations' elements and interactions. Indeed, many deception operations are not as linear as the flow chart implies. Fortunately, though, many of the complexities that do arise in deception operations can be understood in terms of the basic deception-process, as will be described.

2.2.1 The basic deception-process

The basic deception-process is shown in Figure 1, and it is described here. References to items in the figure are in **bold text**. The deception operation begins with **Deception-Operation Development** (top-most box in figure). The deception operation's **plan**, **deception,** and means for **engaging the target** are developed, roughly in that order. **Planning** is an iterative process that is conducted throughout the deception operation. Its first step is recognition of the need or **opportunity** to deceive a target. What must be done in deception planning, and often is not, is to

determine the result desired from the deception. Mere acceptance of the deception may not be advantageous, and it may in fact prove costly. For example, a clever honeypot could attract an unwanted horde of script kiddies, and hiding a host's log files may make the hacker uncertain of the evidence he's left, prompting him to erase the entire file system, just to be safe (e.g., "rm –rf /"). Thus deception is a means, not an end. The **objective** of a deception operation is: **1)** to induce the target to take some specific action-- perhaps to do nothing, and **2)** to exploit that action, or otherwise use it to advantage.

Deception operations are ultimately against individual hackers, so planning includes **identification** of the deception **targets**, and **analysis** of their vulnerabilities to deception. Planning also involves **risk analysis** and **operations security**, to ensure the deception is not revealed to the target.

To induce the target to take the intended action, a **deception story** is designed, and it is implemented using various ruses. The deception story is presented to the target in his observation arenas. Typically, the most effective observation arenas are the target's intelligence sources. One of the primary ruses used in computer security are honeypots. They are computer systems that are designed to be probed, attacked or compromised by hackers. A honeypot contains servers and content that are attractive to hackers. Honeypots are placed on a network, where hackers are likely to encounter it. Honeypots are most useful for detecting attacks and for collecting intelligence about hackers. A honeynet is a network of honeypots.

The deception operation is **Deployed** (second box in figure) by presenting the deception story to the target, in his **observation arenas**. This is a key transition in the deception process, as the deception operation is now out the planner's control until the return of feedback that suggests an appropriate response. The deception story is maintained until it is received by the target. This can occur almost immediately, as with honeypots on a network's DMZ. Alternatively, the deception story might be maintained for months or years before being received, as might occur with an intranet honeypot used for detecting insider hacking.

The **Target is Engaged** (third box in figure) once he receives the deception story. The target is successfully **deceived** when he **receives** the deception story, **accepts** it, and, as a consequence, **takes the intended action**.

Feedback channels provide information about the target's reception of the deception story, and his response to it. The ultimate goal of deception operations is **exploiting the target's response**.[4] This occurs after the **feedback is collected and analyzed**, and it is known that the target has taken the intended action. For honeypots, feedback channels are an essential feature. For example, Symantec's ManTrap honeypot can record much of hacker's activity, including network traffic, process activity, and keystrokes. ManTrap can also detect hacker activity and send alerts.

[4] Thanks to Fred Feer for showing us how the exploit is the deception operation's ultimate goal. In the military deception literature that we used, the exploit's central role is under emphasized.

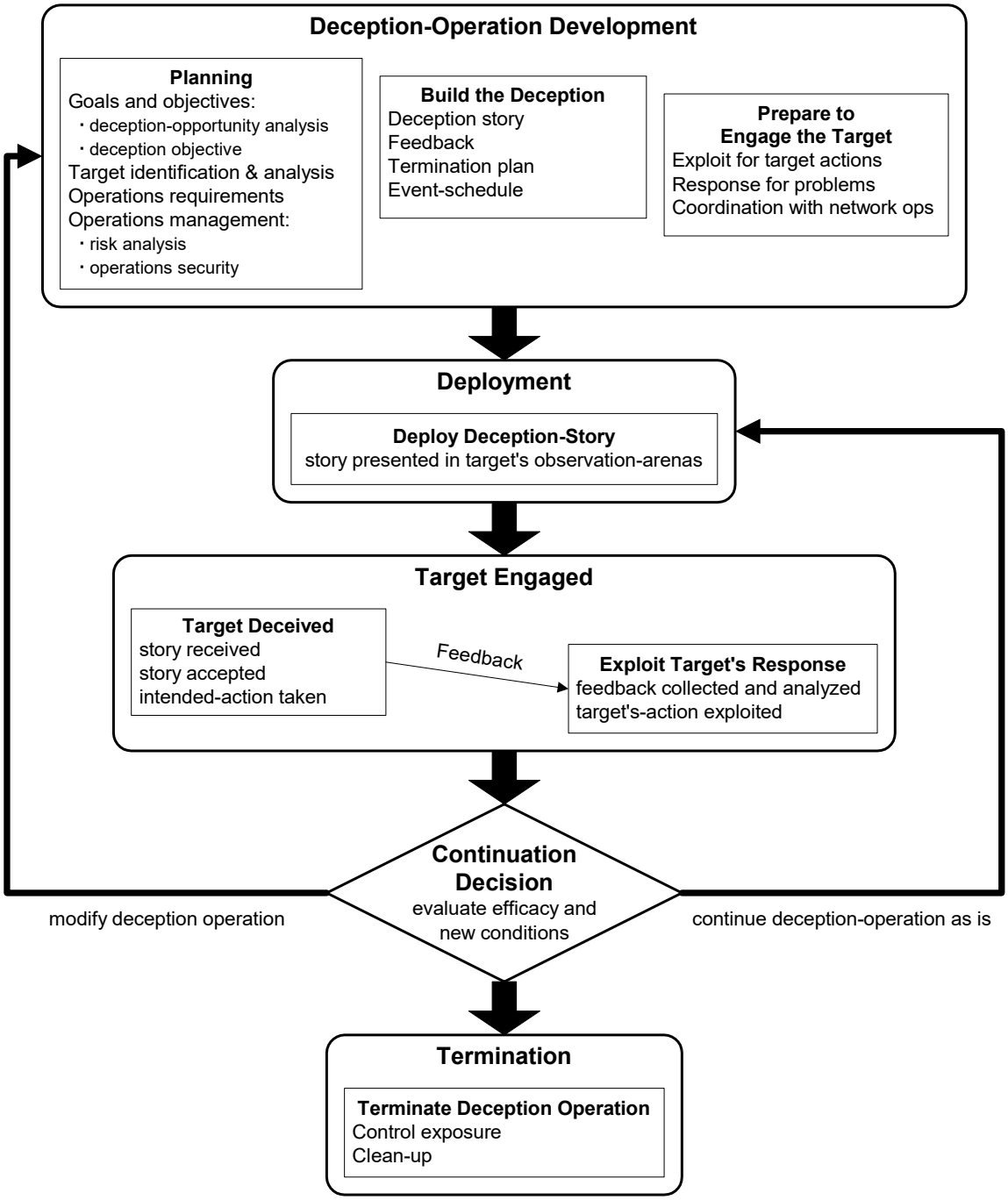


Figure 1 : The basic deception process

The deception story exerts control on the target, manipulating him at a distance. Such manipulation may be intended to have a very short existence. For instance, BackOfficer Friendly (BOF) is a honeypot that can impersonate unauthorized remote-access servers, like BackOrifice. Such servers are installed by hackers via Trojan horses. BOF's impersonation is superficial and its ruse can quickly be discovered by the hacker, but not before he is detected. Other deceptions

may be intended to last indefinitely. For example, a fake VPN interface can be used to draw attention away from a network's real VPN interface. The deception is intended to last indefinitely.

A **continuation decision** (diamond in figure) is made for the deception operation, based on its efficacy and the current situation. The process can be **terminated**, **continued as-is**, or **modified**, in which case the process returns to deception-operation development. **Termination** (last box in figure) occurs when the deception story has achieved its purpose and is no longer needed, or when the target discovers the ruse. The target often discovers the ruse when his response to it is exploited. For example, hardware keystroke-loggers are dongles that attach to the keyboard cable. Their effectiveness depends on stealth: they are located behind the computer, appear to be a normal part of the cable, and few people know about them. When a hacker is confronted with evidence from a keystroke logger, the ruse will probably become apparent. Thus, it can no longer be used against him or his accomplices. Terminating the deception involves **controlling exposure** of the ruse, so it might be used again, as well as **cleaning-up** its affects upon computer systems and personnel.

2.2.2 Complexities in the deception process

Real-world deception operations tend to be more complex than the basic deception-process shown in Figure 1. Fortunately, these complexities can be understood in terms of the basic process, as it is the core of the deception operation. Two of the major sources of complexity are described here: 1) multiple deception stories, and 2) operational failures.

Deception operations can use multiple deception stories, and there can be multiple actions that the target is intended to take. The stories and actions may be inter-related, requiring them to be conducted in parallel or serially. Further, there can be multiple targets. For such deception operations, the basic process is used, but its components occur more than once: multiple deception stories are developed; there are multiple deployments; and there are multiple target-engagements. The order of the process components is determined by their inter-relationships.

In deception operations, there are a plethora of potential problems that lead to operational failure. For example, the deception will fail if: the deception story isn't received; the target discovers the ruse; the story is not interpreted as intended; or the intended action isn't taken. For each potential problem, there are many possible reactions, and this makes problem-handling a complex process. However, the process can be understood by viewing the problems, and reactions to them, as exceptions to the basic deception-process.[5] For example, in Figure 1, if the target is engaged, but does not take the intended action, an exception occurs and the basic process is no longer followed. Instead, a new exception process is followed. Possible exception processes include: 1) terminating the basic process, 2) starting over at the beginning of the basic process, and 3) addressing the problem, then resuming the basic process. In the last example, the problem could be addressed by implementing and deploying an additional ruse.

[5] This is a common technique for understanding complex processes: an error-free process is identified, and error conditions are understood as exceptions to it. For example, in software engineering, software is initially designed to process valid data, and invalid data are processed as exceptions [Jac75]. In the study of social relationships, Tolstoy uses a similar technique to describe family-life, "all happy families resemble one another; each unhappy family is unhappy in its own way" [Tol75].

2.3 Developing the deception operation

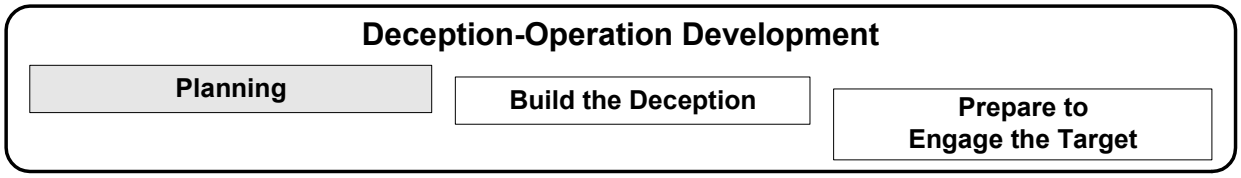
This paper focuses on deception-operation planning and on building the deception. In the basic deception process shown in Figure 1 (page 7), they are represented by the first two boxes in deception-operation development.

Developing a deception operation is similar to developing weapon systems, in that both engineering and tactical skills are needed. As with weapon systems, the deception operation is developed and deployed in a changing, uncertain and adversarial environment. As with software development, deception-operation development is an iterative process. It requires continual reexamination of the target, objectives, stories, and means, throughout the life of the deception operation [JDD96]. Deception planners must be prepared to respond to the dynamics of the network, including changes in its threats and security needs.

Deception-operation development can start in a top-down or bottom-up manner. In the top-down approach, the deception planner first identifies an action he'd like the target to take. He then develops a deception story and ruses to induce that action. In the bottom-up approach, the planner starts with a ruse, or with a vulnerability to deception in the target. He then finds a way to use the deception to advantage. However, whether the planner starts top-down or bottom-up, the operation is composed by working both top-down and bottom-up. For example, the deception objective must be designed with ruses in mind as they have a significant affect on the operation's effectiveness and costs.

In all cases the deception operation is created by the planner using skill, analysis, guile, cunning, ingenuity, and practice-- and at times ignorance and desperation. In addition, military deception sources stress the need for creativity in deception planning [JDD96, FN95].

3 Deception planning



A prince or general can best demonstrate his genius by managing a campaign exactly to suit his objectives and his resources, doing neither too much nor too little. Carl von Clausewitz

Deception-operation planning provides direction for the operation by developing its goals, objectives and requirements.[6] In conjunction, the targets are analyzed to learn their vulnerabilities to deception. Planning also includes operations management activities that are conducted throughout the deception operation. Deception-operation planning is described in the following sections:

3.1 Deception opportunity analysis

Development of the deception operation often starts with **deception opportunity analysis**. It identifies ways deception can be used to support CND. For the deception operation to be effective, it should be **fully integrated** with the overall CND effort. Conversely, deception should not be implemented as an isolated stand-alone security measure. The deception operation should be an appropriate security-measure, given its costs and benefits and the security objectives it serves. Further, deception should be used only when it is the most effective measure, among the alternatives. The deception operation must be compatible with, and coordinated with, the network's security and production operations. Deception is **not an end** in itself, and it should not be used simply because there are clever ways to trick hackers.

Deception is not appropriate in all situations. Usually, deception should not be used until basic computer security measures have been properly implemented, such as having a documented security policy, enforcing strong passwords and installing security patches. Unfortunately, for many organizations, implementing the basic measures is difficult, and there are not adequate resources for it, much less for deception. A U.S. Marine Corps (USMC) deception manual makes a similar observation, "we're too busy trying to do things the way they're supposed to be done to try to deceive anybody" [USM89]. The use of deception depends upon having the ability and resources to carry-out operations beyond basic computer-security measures.

[6] This planning process is adapted from the U.S. Joint Forces' deception process [JDD96].

To identify deception opportunities, planners need an understanding of the uses and means of deception. To this end, the uses of deception for computer security are described at length in chapter 5, and the means are addressed in chapter 6.

For each use of deception that is identified, its **benefits** and **limitations** should be carefully analyzed. Of particular interest is the deception operation's tactical effectiveness (e.g., how well does it detect attacks), and its contribution to the overall CND effort (e.g., does it help solve CND's most pressing problems). A deception operation's benefits and limitations will be analyzed throughout its development and deployment.

3.2 The deception objective

...it became a creed [among deception planners] to ask a General, "What do you want the enemy to do," and never, "What do you want him to think?" Dudley Clark, WWII deception planner[7]

The **deception objective** is the desired result of the deception operation; it consists of: **1)** the intended *target action*, and **2)** the *deception exploit.*[8] The **target-action** is a statement of what the hacker is to do (or not do) at some time and location. It is always stated in terms of specific actions, such as, "cause the targets' attacks against our server to be performed, instead, against the honeypot server". A statement such as "have the hacker think that the honeypot server is the real server" is not a target-action, rather, it is a desired perception (described in section 4.1, page 17). Having the hacker think a certain way is important only as a step toward getting him to make the decision that will result in the intended action. Thoughts without action are of little security value.

The **deception exploit** is a statement of how the target-action will benefit CND, e.g., through attack detection, prevention, or response. The deception exploit may include actions to be taken against the target, following the target-action. For instance, the prior example's deception exploit would be, "for successful attacks against the honeypot, the honeypot will record the attack and send an alert." Some deception exploits do not require taking action against the target, e.g., when using a ruse to confound operating system (OS) fingerprinting, the deception exploit is thwarting attacks that depend on accurate OS fingerprinting.

The deception operation's ultimate goal is successful completion of the deception exploit. The deception-story and ruses are just means for inducing the target-action. After the story is deployed, feedback is analyzed to determine when the target-action is taken. The deception exploit can go into effect after the action is taken.

3.3 Target identification and analysis

It was so important to the deception work to be able to put oneself completely in the mind of the enemy, to think as they would think on their information, and decide what they would do. WWII deception planner[9]

Deception attacks the target's perception and his thinking process, so effective deception requires intelligence on who the target is, how he works, and how he thinks.

---

[7] [Mur80]

[8] The *deception objective* is adapted from the U.S. Joint Forces deception manual [JDD96]. However, its *deception objective* only consists of the *target action*. We include the *deception exploit* with the deception-objective, as it is the deception-operation's ultimate objective.

[9] [Mon78]

3.3.1 Intelligence concepts and definitions

The U.S. military defines *intelligence* as "information and knowledge obtained through observation, investigation, analysis, or understanding" [JDD01]. The obtaining of intelligence information is called *intelligence collection*. For hackers, any information they obtain for hacking will be considered intelligence collection. Observation includes surveillance and reconnaissance. *Surveillance* is systematic observation used to collect whatever data is available, while *reconnaissance* is a mission performed to obtain specific data. *Intelligence analysis* is the process of deriving knowledge and understanding from intelligence information. The *intelligence process* uses the *intelligence activities* of collection and analysis.

The definition of counterintelligence varies among the U.S. armed services [DD00]. For this paper, *counterintelligence* is defined as "intelligence on the hacker's own intelligence processes and activities." Within computer security, we categorize counterintelligence as a subset of *intelligence*.

There are two types of intelligence: basic and current. *Basic intelligence* describes long-term conditions and *current intelligence* describes the present situation. Current intelligence for computer-security incident response is described by Yuill, et al. [YWS00]. From a tactical perspective, intelligence seeks to understand the *area of operations*, as well as the enemy's *capabilities*, *intentions*, and *courses of action*. From an intelligence perspective, the hacker includes systems and tools that act on his behalf, e.g., a worm or port-scanner, respectively. Similarly, the defender includes systems and tools that act on his behalf, e.g., a firewall or vulnerability-scanner, respectively. Such systems and tools have a course-of-action, and it is understood by knowing *how the system or tool works*. There are three types of courses-of-action: possible, most likely and most dangerous.

3.3.2 Identifying targets

Computer-security systems face a wide variety of threats. Howard classifies hackers primarily by their intentions: professional criminals, corporate raiders, hackers, vandals, terrorists, and spies [How98]. Hackers also vary widely in their capabilities and physical locations. It is possible to create deceptions that are effective against a wide variety of hackers. For example, Cohen, et al., have conducted experiments in which a particular deception worked against both undergraduate neophyte-hackers and seasoned penetration-testers [CMS01].[10] However, for deception operations, if a specific **type of target** can be identified, then its unique vulnerabilities to deception can be exploited. For example, script-kiddies' have a youthful naiveté that is vulnerable to deception.

Two specific types of hackers are exceptionally vulnerable to deception: insiders and the hackers investigated during incident response.[11] **Insiders** are hackers that belong to the organization that owns the network, e.g., an employee. Insiders have several unique vulnerabilities to deception: **1)** deception planners work for the same organization as the insiders, and this provides planners with much understanding about how insiders might work and think, **2)** physical surveillance can be used for deception feedback, and **3)** all of the

[10] The experiments tested the efficacy of a device that apparently creates many imposter computers on a network. However, the paper does not appear to describe the device in detail.

[11] These two types of hackers are not mutually exclusive. Incident response investigates hackers, including insiders.

organization's communication channels can be used to communicate the deception, e.g., internal memos. The hackers investigated during incident response also have unique vulnerabilities to deception. These hackers can be insiders or outsiders. During **incident response**, deception operations are against a specific person or group, and not just a potential type of hacker. During the incident, the investigator learns about the hacker himself, including his vulnerabilities to deception.

Many hackers work in small groups. For deception, it is helpful to know a group's decision-making process and its key decision-makers. For example, there are many reliable reports of state-sponsored hacking, and some of these hacking groups may be large [Fre00, Lic02, Sto89]. Some state-sponsored hackers are intelligence-collections agents. Their personal decision-making will be limited mostly to hacking tactics, as the object and conduct of their hacking is determined by their intelligence customer. Such hierarchical decision-making can also occur with hackers-for-hire and for terrorists. In general, when **deceiving groups**, the deception target is the decision-maker with the authority to make the decision that will achieve the deception objective [JDD96]. The target can be the intelligence collector, analyst, or customer. However, this paper will assume the deception target is the hacker himself, in order to focus on the central deception processes and to avoid distracting complications.

3.3.3 How hackers work

An understanding of how hackers work reveals their vulnerabilities to deception and how those vulnerabilities can be exploited. Fortunately, much is understood about how hackers work, as the complexity of hacking compels hackers to use publicly available tools and information. There are many books on hacking techniques [MP01, MSK99], and the Honeynet Project has reported the findings from their extensive surveillance of hackers [Hon01]. An up-coming chapter (5, page 41) presents two models of the hacking process, and it shows how these models reveal opportunities for using deception.

It is especially important to understand how the **hacker intelligence-process** works, as it is both a means and an end for deception operations. For instance, the hacker's intelligence collection is a means for communicating the deception story, and one use (end) of deception is thwarting the hacker's intelligence process. For the hacker's intelligence **collection**, it is helpful to know: **1)** the information that is potentially available to him, **2)** his tools and capabilities for information gathering, **3)** the sources he uses, **4)** the reliability he assigns to information, and **5)** how he cross-validates information. For the hacker's intelligence **analysis**, it is helpful to know: **1)** how he processes, filters, ascribes meaning to, and uses information [JDD01], and **2)** what he knows, understands, and believes about the computer system and its personnel. There are specific **vulnerabilities** in the hacker's intelligence process that are helpful to know: **1)** the single sources of information that he relies upon, as deception is easier when the ruse will not be cross-validated, **2)** the information he uses that is superficial and easily misrepresented, as with a ping scan, **3)** the investigations he performs when he is naive and thus easily duped, as during his initial network reconnaissance, and **4)** the intelligence processing of the hackers' automated agents, such as worms, as their simplicity and determinism may be easily duped.

Outsider hackers (non-insiders) are almost always **naive** about the networks they hack. A hacker's experience and skills are often asymmetric with the experience and skills needed for the network he is hacking. For example, hackers typically have never legitimately worked in a network or organization like the ones they are hacking. Script-kiddies usually hack from home

or school and have not worked in a large company. The Chinese government reportedly has a force dedicated to hacking U.S. Department of Defense (DoD) computers [Ger99]. These hackers may have never legitimately worked on large-budget systems like the ones they are hacking, nor have they worked in the U.S. DoD. Further, international hackers' culture and language are usually asymmetric with the culture and language of the network they are hacking. Such asymmetries result in a naiveté that is vulnerable to deception. In contrast, insiders often work with specific knowledge of assets and vulnerabilities. This tends to make their work more directed and focused than outsiders. Also, insiders have more intelligence sources than outsiders, and they may be able to collect intelligence via permissible computer use.

An example of exploitable naiveté is provided by Matt Singer, a young man who reportedly hacked full-time while living on welfare [FM97]. Matt's brother cautioned him about getting caught, but Matt replied that he was telnetting through so many networks that it would be impossible for his end-victim to trace his connection. However, in his youthful naiveté, Matt did not realize that his connections were being secretly recorded by a system administrator at one of his initial telnet connections.

3.3.4 How hackers think

An understanding of how hackers think also reveals vulnerabilities to deception and how those vulnerabilities can be exploited. The key elements of the hacker's thinking are his: **1)** intentions, **2)** perceptions, **3)** decision-making process, and **4)** his psychological vulnerabilities to deception.

The hacker's **decision-making process** is both a means and an end of deception:

- the means of inducing the target-action is misinforming the target's decision-making process, and
- one use (end) of deception is thwarting attacks by degrading the hackers' decision-making process, as will be explained in section 5.3 (page 48).

The hacker's decision-making process will be influenced by such things as: **1)** the information available to him, **2)** his skills and techniques, **3)** the maturity of his judgment, **4)** his perceived risk, and **5)** his willingness to take risk. For instance, a hacker who has invested much time in penetrating a network will act cautiously to avoid loosing the unauthorized access he has acquired. When using deception to deter hacking, the deceptions do not have to be entirely convincing if the hacker is acting cautiously. For example, dubious exaggerations of intrusion-detection capabilities can deter hackers if they think it is too risky to call the defender's bluff.

The hacker's **perceptions** of the network can be used to deceive him. For example, one of the easiest ways to deceive is to present the target with what he expects [Heu81]. Honeypots on a DMZ will be most believable if their vulnerabilities are consistent with what hackers expect. If the network is famous for its strong security, then hackers will be suspicious of a honeypot with gross vulnerabilities. Subtle vulnerabilities will be more believable there.

The hacker's particular **psychological vulnerabilities** to deception can also be used to advantage. The hacker Matt Singer was reportedly obsessive, compulsive and undisciplined in his hacking [FM97]. Such shortcomings significantly limit a hacker's ability to carefully and critically examine ruses. A number of books provide insights into how hackers think: chronicles

of prolific hackers [FM97, Sto89], the aforementioned findings from the Honeynet Project's surveillance [Hon01], and a sociologist's study of hacker culture [Tho02].

3.4 Operations requirements

Requirements are developed to ensure the deception operation effectively contributes to the **overall CND** effort. The deception operation's primary requirements include costs and effectiveness. Like all security measures, **cost requirements** are imposed by limited security resources. The deception operation's **effectiveness** can be assessed in a number of ways, including the number, percentage, or type of attackers thwarted **(**though only rough estimates may be possible). For instance, a honeypot with deceptive content is expected to regularly provide useful intelligence about hackers on the network's DMZ. Another measure of effectiveness is the **certainty of success**. There is often uncertainty about whether the target will observe the deception, interpret it as intended, and then take the intended action. Among honeypots, these uncertainties vary greatly, depending upon how the honeypot is constructed and used. The **ability to assess** the deception operation's effectiveness can also be a requirement. Often, it will be difficult to assess the deception operation's contribution to CND. For example, when deception is used for deterrence, it may be impossible to know how much hacking was actually deterred.

To meet the deception operation's overall requirements, there can be requirements for individual parts of the operation. For example, to meet overall cost-requirements, it may be necessary for the operation's deceptions to be **reusable** against the same target. Some honeypots use simple ruses that are easily detected by the target, but the honeypots can be reused by simply changing their IP address. There are requirements related to the intended **target action**, including the likelihood that the target will take the action, and also the duration of the action. For instance, if a honeypot is to prevent attacks by wasting the hacker's time, then the target must be induced to engage the honeypot for substantial periods of time. There are requirements for the deception story, and they are discussed in section 4.1.4 (page 30).

3.5 Operations management

This section explains three requirements for managing the deception operation: centralized control, operations security, and risk analysis. Deception operations often require **centralized control** to ensure that the various elements involved in the deception are portraying the same story [JDD96]. Also, deception planning must be **coordinated** with real systems and operations to ensure the deception does not conflict with them, and vice versa. For example, to thwart attacks, all of a network's public Internet-servers are to falsely identify themselves as running Solaris, when in fact they run Windows. Deception planners must ensure that all of the servers' system administrators portray a consistent story.

Deception operations need their own **operations security** (OPSEC) [Dew89]. Deception operations often need to be hidden from the target and from friendly personnel who lack a need to know. Decisions to hide a deception operation from friendly personnel must be balanced with the possibility of inadvertently deceiving them. Many deceptions become ineffective when the target discovers the deception, e.g., a honeypot web-server. Some deceptions become ineffective simply by the target knowing they exist, even if no deployments are discovered. For example, if insider hackers learn that hardware keystroke-loggers are being used, the hackers can simply use their own keyboards. Also, the target may be able to deduce the existence of sensitive

information just by knowing a deception operation exists, e.g., costly deception operations accompany valuable assets [JDD96].

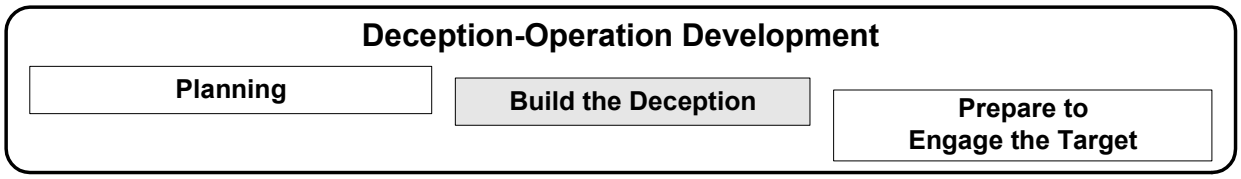
The uncertainties of deception operations require that **risk analysis** be performed throughout the operation. For deception operations, the **causes of failure** can be divided into two categories: **1)** those resulting from detection by the target, and **2)** those resulting from inadequate design or implementation by the deceiver [USA88]. The deception operation can fail at any point in the deception process illustrated in Figure 1 (page 7), e.g., the deception story is not observed by the target, the target does not interpret the story as intended, the target does not take the intended action, or the feedback channel doesn't work. Generally, the more complex the deception operation is, the more likely it is to fail. For example, Lance Spitzner observes that the more realistic a honeypot is, the more complex it will be and thus the more likely it is own security will fail [Spi03]. Also, like any software, the deception operation's software can have programming or configuration bugs. For instance, the network intrusion-detection system (NIDS) named Snort has a feature that can be used to deceive OS fingerprinting [SNO00]. The feature is called *Flexible Response,* and its documentation warns, "be very careful with Flexible Response. It is quite easy to get Snort into an infinite loop" [RG03].

Failure may enable the target to conduct a **counter deception-operation**. If the target discovers the deception story, he can pretend to be deceived, and then exploit the original planner's predictable response. The target will use the deception process to conduct his own deception operation, and much of it is ready-made by the original planner, who waits attentively and responds predictably. For example, in responding to an on-going incident, honeypots are deployed in tactical locations, to detect attacks and gather intelligence. Upon discovering the honeypots, the hacker can attack them to create diversions and to supply false intelligence regarding his capabilities and intentions.

For deception operations, failure can adversely effect computer security via **unintended target actions**. Deception can inadvertently induce additional attacks by the hacker. When complex honeypots fail, they can provide a platform from which the hacker can carry out additional attacks. If deception makes the hacker uncertain about the evidence he's left, he may respond by erasing the hard drive, as Berferd attempted on Bill Cheswick's honeypot [Che92]. In honeypot experiments led by Eugene Schultz, concern was raised that attackers would respond vengefully after learning they were deceived [SRH00]. Deception can inadvertently induce the hacker to improve his defenses and thereby defeat computer-security measures. For example, to deter attacks, a network's detection capabilities are exaggerated. However, instead of being deterred, the hackers use greater stealth and thus inadvertently evade detection mechanisms that would normally catch them.

For deception operations, failure can **adversely effect computer security** directly. The deception operation may prove less useful than anticipated, e.g., costs exceed actual benefits. Also, the deception operation can produce *unintended effects*. In deception terminology, **unintended effects** refers to the consequences of inadvertently deceiving neutral or friendly parties. For example, a network's DNS server crashes and a resourceful user runs a port-scanner to locate the network's public FTP server. Instead, he inadvertently locates a honeypot FTP server that contains deceptive information.

4 Building the deception



This section presents principles and techniques for building the deception: the deception story is constructed to induce the target action; feedback reveals when the story is received and the target-action taken, and the termination plan ensures the controlled and orderly release of information related to the deception.

4.1 The deception story

The deception operation's objective is to induce a specific target-action that benefits CND. The **desired perception** is what the target must believe in order for it to take the intended action. The **deception story** is an outline of how the computer system will be portrayed so as to cause the target to adopt the desired perception. This section presents principles and techniques for developing the deception story, and they are illustrated by an example deception operation.

In general, determining the desired perception can be difficult, as it requires an understanding of how the target works and thinks. Generally, it is much easier to **reinforce an existing belief** than to establish a new one [Heu81]. For example, a computer-savvy hacker will reasonably expect that a high-volume web site uses multiple web servers, load balancing, and a multi-tiered architecture. Also, to ensure the target-action is taken, the desired perception should make the target believe the target-action is in his **best interest**. Ideally, the target will perceive the intended action as compelling, and alternative actions as untenable.

Building and deploying the deception story is like writing and performing a play, though deception poses the added challenges of an adversarial audience whose attendance is uncertain and remote, and whose response is difficult to discern [DH82b]. Often, only part of the story needs to be implemented. The rest of the story is **notional**, implied by both what is real and what is implemented.

This section's example deception operation is descriptively named the **intellectual-property (IP) deception operation**. The purpose of the operation is to protect a company's intellectual-property database. The IP database is a collection of trade secrets, recorded in various formats such as MS Word and AutoCAD. The database is maintained by the company's IP department, and the department's director and his assistant have exclusive access to it. The IP department's intranet web-site describes the department, and the procedure for employees to submit trade secrets. Submissions are made by copying files to a shared folder on the assistant's workstation, and the assistant stores the files on the IP database. The database is kept on a single computer, and the computer is on a private LAN that can only be accessed by the director and his

assistant. This example is illustrated in Figure 2.

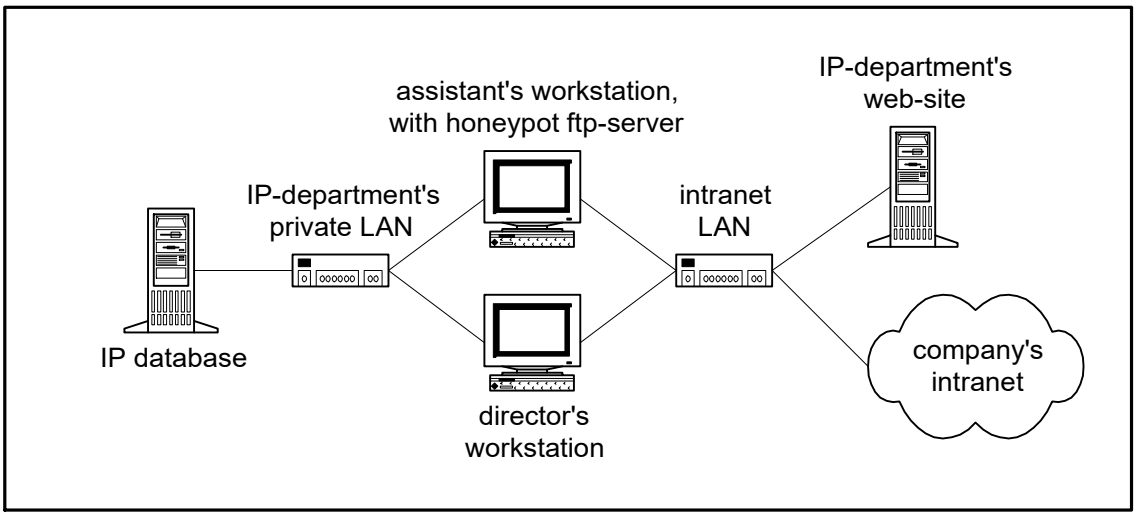


Figure 2 : The intellectual-property (IP) deception operation

The *target* of the IP deception operation is hackers who attempt to gain unauthorized access to the IP database. The intended *target-action* is one that reveals the target's presence and intent, but does not compromise computer security. The *deception exploit* is attack detection and the recording of forensic evidence. The *desired perception* is an exploitable vulnerability that provides access to the IP database. The *deception story* is a vulnerable FTP server on the assistant's workstation. The FTP server will appear to be a particular make and version that has a buffer-overflow vulnerability.

The next four sections describe additional principles and techniques for building the deception story, as summarized in the following table:

Principles and Techniques for Building the Deception Story			
4.1.1 Designing the story (pg. 18)	***4.1.2 Presenting the story to the target*** (pg. 21)	***4.1.3 Implementing the story*** (pg. 24)	***4.1.4 Operational requirements*** (pg. 30)
• Essential design-criteria • Key design-principles • Tactics and costs	• Affecting the target • Fitting the broader context • Revealing the story	• Determining what to implement • Exploiting the target's intelligence sources • Realism	• Time-related requirements • Adaptability and reuse • Potential problems

4.1.1 Designing the story

This section presents principles related to the deception story's design: 1) the essential design criteria, 2) a summary of the key design-principles and 3) making the story tactically effective and cost effective.

4.1.1.1 Essential design-criteria

The deception story's essential design-criteria [JDD96,DH82b,USA78] are that it be:

- **plausible:** the story must be plausible from the target's perspective. Consequently, it should appear *appropriate* from both an engineering and operations perspective. Also, it must appear to be something the defender is *capable* of doing. The story should be *consistent* with real systems and operations, as well as being internally consistent.
- **receivable:** The story must be something the target's intelligence is capable of receiving and interpreting as intended.
- **verifiable:** If the target will verify the story through multiple intelligence sources, then the story should be portrayed through more than one source. For example, to avoid honeypot web sites, a target can verify web sites he discovers by searching for links to them from real web-sites.
- **efficacious:** For the story to be efficacious**,** it must be received, and it must effectively induce the desired perception and target-action.
- **implementable:** The story must be something the deception planner is capable of implementing.

The IP deception operation illustrates those design-criteria: the deception operation's story is *plausible* to insiders because the IP assistant is *capable* of installing an insecure FTP server. File transfers are *consistent* with the assistant's job responsibilities, as described on the IP web-site. Also, the FTP server's vulnerability is *consistent* with other security problems on the intranet. The vulnerable FTP server is visible to a port scan, which is the target's expected means of *receiving* intelligence. The IP web-site's description of the assistant's file-transfer responsibilities helps to *verify* the story. The story is potentially *efficacious* as the FTP server's vulnerability is the target's only apparent means for accessing the IP database. The story can be *implemented* using a COTS (commercial off-the-shelf software) honeypot, such as Specter.

4.1.1.2 Key design-principles

In the present section on building the deception story (section 4.1), numerous principles are described. The most important among them are summarized here, to consolidate them for use in deception planning. Each principle is cross-referenced with the upcoming section where it is further explained.

- **Inducing the target-action:**

The target-action is easier to induce if it something the target is predisposed to doing, such as: **1)** something he is already planning to do, **2)** something he normally does, or **3)** something he wants to do (section 4.1.2.1, page 21).

- **Making the story believable:**

In general, it is easiest to persuade the target to believe something he already expects (section *4.1.2.3*, page 24). Also, it can be easy to deceptively portray things that are normally **hidden** from the target. Often, the target only expects to find limited information about something that is hidden, in which case that is all that needs to be portrayed (sections *4.1.2.1* on

page 21, and *4.1.2.3* on page 24).

- **Preventing the target from uncovering the deceptions:**

The deception story's **falsehood** should be kept to a minimum (section 4.1.2.2, page 23). The truth is much stronger than a lie, and it can be difficult to maintain a lie over time. Also, minimizing falsehood makes the deception story easier to implement. Some techniques for minimizing falsehood are: **1)** make the story simple, **2)** weave the story into the truth (section 4.1.2.2, page 23), **3)** provide no more detail than is necessary (section 4.1.3.3, page 29), and **4)** impersonate things that are normally concealed from the target, as he will only expect to see bits and pieces of information about them (section 4.1.2.3, page 24).

Another way to prevent the deception story from being uncovered is to minimize the target's **scrutiny** of the deceptions. Three techniques for doing this are: **1)** the deceptions can be communicated to the target via his weakest intelligence capabilities (section 4.1.2.1, page 21). If the target cannot examine the deception closely, he will be less likely to detect it. **2)** Deceptions can be communicated to the target when he has little time to scrutinize them (section 4.1.3.3, page 29), and **3)** The deceptions can portray things of which the target has little understanding (section 4.1.3.2, page 27).

- **Ensuring the target receives the story:**

In the course of hacking, the target **eagerly seeks** particular information; this presents an opportunity for using the target's intelligence sources to communicate the deception story to him (section 4.1.3.2, page 27).

4.1.1.3 Tactics and costs

The deception story should be designed to afford **tactical advantage** and **low cost**. Examples of tactical advantages are thwarting the hacker as soon as possible, limiting damage, and obtaining accurate intelligence. For the IP deception operation, the honeypot FTP server is put on the assistant's workstation rather than on the IP database. This allows attacks to be detected before they reach the database.

An obvious way to keep costs low is to make the deception story **simple**. Complex deception stories can be both expensive and error-prone. Another way to reduce costs is to base the deception story on **standardized systems** and protocols, e.g., FTP. Deceptions based on proprietary applications can be costly if the deception must be custom made. Also, there will be limited opportunity for using such deceptions elsewhere. In contrast, when deceptions are based on standardized systems, COTS deceptions may be available (e.g., commercial honeypots), and there is greater potential for reusing the deception. Further, standardized systems tend to change less than proprietary applications. A third way to reduce costs is to minimize the deception story's **falsehood**, as will be discussed later (section 4.1.2.2, page 23).

For the IP deception operation, the deception story is closely tied to the IP department's procedures and systems. This helps reduce the story's falsehood. However, if the real procedures and systems change, the deception story may also need to change. Basing the story on a vulnerable FTP server is economical, for several reasons: the FTP protocol is unlikely to undergo changes that affect the deception operation; inexpensive honeypot FTP servers are

commercially available, and such a honeypot could be used in the network's other deception operations.

4.1.2 Presenting the story to the target

Principles and Techniques for Building the Deception Story			
4.1.1 Designing the story (pg. 18)	***4.1.2 Presenting the story to the target*** (pg. 21)	***4.1.3 Implementing the story*** (pg. 24)	***4.1.4 Operational requirements*** (pg. 30)

This section presents principles and techniques for presenting the deception story to the target:

- affecting the target so as to induce the intended action,
- fitting the story into its larger context (page 23), and
- revealing the story in a convincing manner (page 24).

4.1.2.1 Affecting the target

Screenwriters craft their stories for specific audiences (e.g., teenagers) and for specific purposes (e.g., ticket sales). So too, the deception planner makes the deception story for the target and for the purpose of inducing the target-action. This section presents principles for crafting the deception story so that it effectively induces the target to perform the intended action.

For the story to be effective it must be **received** by the target and **interpreted** as intended. Further, the story will best ensure the target-action if that action appears **compelling** to the target, and if alternative actions appear dangerous, unattractive or impossible. Also, the target-action is easier to induce if it something he is **predisposed** to doing, such as: **1)** something he is already planning to do, **2)** something he normally does, or **3)** something he wants to do. At a minimum, the target-action must be something the target is **capable** of doing.

The deception story should fit in with the target's prior and future **courses of action**, i.e., the target's course prior to encountering the story, his likely responses to the story, and then his subsequent course of action. If the story is accepted, additional deception may be needed to sustain the target's erroneous beliefs.

In the IP deception operation, the target is hackers who are seeking to steal intellectual property. It is expected that the target will first locate and study the IP department's web-site. From it he will learn that the director and his assistant maintain a repository of intellectual property, and that their computers are promising pathways to the repository. It is anticipated that the hacker will scan these computers for vulnerabilities, and then attempt to exploit a vulnerability that provides easy and stealthy root access. Both of the computers are kept very secure by their users and by the IT department. The honeypot FTP server is expected to be the only vulnerability that the target encounters, making it compelling to attack.

To affect the target, the deception story must be designed to supply the **target's**

intelligence, and it must be designed to affect the **target's thinking** and decision-making. Section 4.1.3 (page 24) describes how to develop the deception story relative to the target's intelligence. Some principles for influencing the target's thinking are described here.

One way to prevent the deception story from being uncovered is to minimize the target's **scrutiny** of the deceptions. A technique for doing this is to portray things of which the target has little understanding. For example, a deception operation's targets know PCs well, but not mainframe systems. If the deception story can be portrayed in the context of a mainframe system, the targets will be less able to detect disconfirming evidence.

In crafting the deception story, it is easy to **miscalculate** how the target will interpret the story. Some of the causes of this miscalculation are explained by Robert Jervis' research on misperception [Jer68].[12] Among all people, there is a **tendency to misperceive** the actions or intentions of others. The tendency is due to the limitations of human mental capacity, and in particular, weaknesses of perception. The tendency is a human phenomenon that is predictable in a statistical sense, though it is not predictable in the sense that it occurs in the same way to all people, and at all times.

An understanding of human misperception is useful to deception planners in two ways. First, the target's tendencies to misperceive can be used to deceive him. Secondly, the deception planner is **vulnerable to misperceiving** how the target will interpret the deception story. Jervis identifies three types of misperception that the deception planner is vulnerable to making. By understanding these misperceptions, the deception planner can better avoid them. Jervis states the misperceptions as hypotheses:

Jervis' **Hypothesis #14** states: *actors tend to overlook the fact that evidence consistent with their theories may also be consistent with other theories*. This hypothesis is especially applicable to the design of deception stories. Careful thought should be given to the target's **possible interpretations** of what is portrayed to him. For example, when deceptively showing something that is expected to be hidden, the entire thing does not have to be shown, only its expected indicators. However, in general, the less information presented to the target, the more interpretations that can be made of it. For instance, a mainframe can be hidden from OS fingerprinting tools (e.g., NMAP) by making the mainframe look like an uninteresting computer, such as a print server. The mainframe simply returns the signature packets that fingerprinting tools expect from a print server. However, other OSs may have the same signature as the print server, including OSs that the hacker prefers to attack.

Hypothesis #6 states: *when someone spends a great deal of time making a plan, they tend to think that the message they wish to convey about the plan will be clear to the receiver*. The sender is very aware of the message he wishes to convey, and he wrongly assumes it will be equally obvious to others. The sender doesn't realize that the message is clear to him because of his familiarity with the plan and message. The sender knows what to look for, and he wrongly assumes others know this as well. As when a teacher instructs his students, the deception story's designer must be aware of **what the target does not know**.

Hypothesis #13 suggests: *it is hard for an actor to see that issues important to himself are not important to others*. The deception planner must try to see the network and story through

[12] Dr. Jervis addresses misperception in international relations. However, his work is cited in primary military and intelligence deception sources [DH82b, Heu81], including the U.S. Army's battlefield deception manual [USA88].

the target's eyes. What is important to the planner may not be important, known or understood by the target. The IP deception operation suffers from this problem. It was assumed that all targets have the skill to identify and exploit the vulnerable FTP server. However, low-skilled insiders will not have these skills. They will only be able to exploit simple vulnerabilities such as weak passwords and misconfigured file-shares.

4.1.2.2 Fitting the broader context

[The deception] must be brought into harmony with the overall situation. WWII deception planner[13]

To be plausible, the deception story must fit within the broader context in which it resides. Three techniques for fitting the broader context are presented: **1)** making the deception story consistent with real systems, **2)** weaving the deception story into the truth, and **3)** using patterns to advantage.

Deception stories occur amidst real systems and their operations, e.g., real networks and real data-processing. To be believable, the deception story should be **consistent** with the real systems with which it is intertwined. Making the story consistent with real systems may require assistance from personnel who work with the systems. In the IP deception operation, the vulnerable FTP server is consistent with other FTP servers and vulnerabilities on the network.

There are several other principles to be observed in making the deception story consistent with real systems: **1)** it is possible for the deception story to be exposed by inconsistencies between itself and real systems. **2)** Compared to deceiving outsiders, deceiving insiders requires greater consistency between the deception story and its broader context. This is due to the insiders' intimate knowledge of systems and operations. **3)** Coordinating the story with systems personnel poses an operations-security risk, as it increases the number of people who know about the operation. **4)** To make the story fit the broader context, it may be necessary to deceive friendly personnel. For example, to make a honeypot web-site believable, real web-sites can be linked to the honeypot. Consequently, there is potential for friendly personnel to follow the links to the honeypot and be deceived by it.

Perhaps the most useful and powerful technique for designing the deception story is to **weave the story into the truth** [DH82]. This reduces the amount of falsehood in the deception story, which makes the story easier to implement and less vulnerable to discovery. Also, by meshing the deception story with real systems and operations, the real systems and operations help to verify the story. Further, once the target has verified much of the story, he will be inclined to ignore or explain away story details that are inconsistent, and that would otherwise reveal the deception. In the IP deception operation, the deception story is a small extension to the real IP systems and operations. The story is implicitly verified by information on the real IP web-site, and by the real workstation that runs the FTP honeypot.

Another useful and powerful technique for designing the deception story is **minimizing the falsehood** in the story. Lies tend to be weak, as they can easily and permanently fail under the light of truth. Also, lies tend to be expensive to implement, and difficult to maintain.

For computer security, old or **salvaged computers** can be a useful means for implementing deceptions. These are real systems that were part of real operations. If not too outdated, they can play an inexpensive and convincing role in deception stories.

---

[13] [DH82]

Various **patterns** in the network can affect the deception story [USA78]. There are patterns in system design, e.g., all of the web servers run Linux. There are also patterns in operations, e.g., the file servers are backed-up during third shift. The deception story may need to conform to existing patterns, e.g., if all web servers run Linux, a honeypot web-server that runs Solaris may create suspicion. Alternatively, intentionally breaking a pattern can be useful. For example, a subnet's workstations all run a secure Solaris implementation. A honeypot workstation is put on the network, and to draw attention to it, it runs an insecure Linux implementation.

Patterns in the use of deception operations can also affect the deception story. If the target is aware of the network's prior deception operations, he's not likely to believe recently-used deceptions, nor often-used deceptions. However, patterns in using deception operations can be used to advantage, e.g., if deception operations are not usually reused, then the target may be vulnerable to a reused deception because he doesn't expect it.

4.1.2.3 Revealing the story

This section describes several ways to reveal the deception story to the target in a convincing manner.

Deceptive showing can be relatively simple when it portrays things that are normally **hidden** from the target, or otherwise inaccessible to him. In such cases, the target will only expect limited and superficial indicators, so that's all that needs to be shown. Showing more may make the target suspicious or reveal the deception.

Usually, the easiest way to deceive is to show the target what he **expects**. If the target expects to find a small number of indicators, that's what should be shown. If the target expects something will be difficult to do, the deception story's realism will be enhanced by presenting the target with difficult tasks [BW82].

In the IP deception operation, the IP database is one of the company's major assets. Hackers will expect the database to be highly secure and difficult to access. The deception story would not be very believable if it portrayed a fake IP database that is easily accessed and compromised.

A technique for revealing the deception story is to provide the story in bits and pieces and then let the target **piece the story together** by inference [USM89,Dew89]. The technique is especially useful when the target only expects limited indicators. Also, the technique is consistent with the target's intelligence activities, as they normally acquire information in bits and pieces. A weakness of the technique is the risk of misinterpretation, as small amounts of information can be interpreted by the target in a variety of ways.

As a general principle, the target will be suspicious of deception stories that portray great **windfalls** for him, especially if he suspects deception is being used [CIA80]. For example, in World War II, there were several instances in which real battle-plans fell into enemy hands, but these windfalls were rejected as deceptions. On the other hand, if the deception story is **too subtle**, the target may not see the message or act upon it.

4.1.3 Implementing the story

Principles and Techniques for Building the Deception Story			
4.1.1 Designing the story (pg. 18)	***4.1.2 Presenting the story to the target*** (pg. 21)	***4.1.3 Implementing the story*** (pg. 24)	***4.1.4 Operational requirements*** (pg. 30)

Implementing the deception story involves constructing its deceptions, both showing the false and hiding the real. The deception story must be designed with its implementation in mind, as the implementation significantly effects the deception operation's tactics and costs. In the IP deception operation, the planner started with the idea that a simple honeypot could be used to detect attacks. The story was then built around the honeypot. Alternatively, the planner may start with a story that will powerfully manipulate the target, and then look for ways to implement it. Generally, the planner will compose the deception by considering both the story conveyed and the opportunities available for implementing it.

The following sections present principles and techniques for implementing the deception story:

- determining what parts of the story to implement,
- exploiting the target's intelligence sources (page 27), and
- providing the appropriate degree of realism (page 29).

As just discussed, these principles and techniques also need to be considered when designing the deception story.

4.1.3.1 Determining what to implement

This section describes how to determine the parts of the deception story to be implemented. Usually, only **parts** of the deception story will need to be implemented. Some of the story will be tied to the truth and portrayed by real systems and operations. Some of the story can be notional, implied by the parts of the story that are real and that are implemented.

To determine what parts of the story to implement, one must understand how the target receives the deception story, and what he expects to see, as will be discussed here:[14]

- **How the target receives the deception story:**

The target's **intelligence capabilities** determine how he receives the deception story. The deception planner will need to understand these capabilities, and especially the collection systems and techniques the target is likely to use. They are the planner's means for presenting the deception story to the target. The planner will also need to know the intelligence capabilities that the target most relies upon, and the target's process for verifying his intelligence discoveries.

Hackers' intelligence capabilities **vary greatly**. Consequently, the deception story may have to be presented in more than one way if it is to be received by all of the deception

[14] This section is based on the U.S. Joint Forces' deception-manual, and in particular, its process entitled "Identify the Deception Means" [JDD96].

operation's targets. For example, worms work mechanically, script-kiddies use publicly available tools, state-sponsored hackers may have proprietary tools, and insiders have access to the physical premises. Most hackers' intelligence collection includes open-source intelligence[15] (e.g., public web sites), as well as network access. Insiders and social-engineers will have access to network personnel. Outsiders who penetrate the network's perimeter will have the same network access as insiders.

For the IP deception operation, the target's intelligence collection is expected to begin with the IP web-site. It will be followed by a port-scan of the two IP workstations. The target will then investigate the listening ports for vulnerable servers. Most likely, the target will not expect to be deceived nor detected, so he will trust what sees, and he will act boldly and quickly.

- **What the target expects to see:**

The deception planner must determine the things the target would expect to see, if the deception story was true. The systems portrayed by the deception story have indicators that the target expects to see. The set of indicators for a particular system form its *profile*. For example, the profile of a default OS-installation consists of its OS fingerprint and its default set of network servers. Knowing something's profile often requires detailed knowledge of it, and assistance may be needed from the personnel who work with it, e.g., network and database administrators. In addition to identifying the things the target expects to see, it is also helpful to identify the parts of the story that the target does not expect to see, e.g., the things or indicators that he expects to be hidden or inaccessible.

For the IP deception operation, the target will expect the IP department's workstations to be secure, but he will look for accidental vulnerabilities. The target will use a port-scanner to find network servers. He'll examine the servers for exploitable vulnerabilities, such as the buffer-overflow vulnerability on the FTP server *WUFtp Version 2.5.0*. The target will expect an FTP server to present its login interface. A buffer-overflow attack normally crashes a server, or it provides access to a root shell. The target will not expect the IP database itself to be easily accessible on the intranet, due to its value and security.

- **Determining the parts of the deception story to implement:**

Having determined how the target discovers the deception story, and what he expects to see, the planner can then determine the parts of the story to implement. There's usually no need to implement things the target does not expect to see. As was mentioned, parts of the story may be notional, and parts may be represented by the real systems that are intertwined with the deception story. If implementing the story proves too costly or impossible, the story will need to be revised.

For the IP deception operation, the real IP web-site and workstations will portray themselves. An FTP honeypot will impersonate a vulnerable FTP server, as the target expects. However, the honeypot does not need to simulate, nor provide, root-shell access. After several failed attacks, the target will simply give up, and attribute the failures to system idiosyncrasies. Nothing needs to be portrayed regarding the IP LAN. The target's knowledge of it is

[15] Open-source intelligence is intelligence obtained from publicly available information.

speculative, and he will expect it to be hidden and inaccessible.

The next two sections discuss further aspects of the deception story's implementation: exploiting the target's intelligence sources, and providing the appropriate degree of realism.

4.1.3.2 Exploiting the target's intelligence sources

Provided the enemy has an efficient intelligence service, provided he is capable of reacting to what he sees, or thinks he sees, he can apparently be taken in again and again. G. Barkas[16]

I wanted to watch the cracker's keystrokes. . . The best solution was to lure him to a sacrificial machine and tap the connection. . . [We] did construct such a machine, ***[but] never managed to lure anyone interesting to it****.* Bill Cheswick[17]

When implementing the deception story, the planner's **goal** is that the story be received by the target, believed, and interpreted as intended. Such manipulation of an adversary can be very difficult and problematic. Fortunately, the target provides an opportunity the planner can exploit to achieve his goal: in the course of hacking, the target **eagerly seeks** particular information, and this presents an opportunity for using the target's intelligence sources to communicate the deception story.

The last section described how the target's intelligence capabilities determine the parts of the deception story to be implemented. This section examines the target's **intelligence sources** and how they can be exploited to implement the deception story. Simply put, an *intelligence source* is something that is used by the target to learn about the network. The U.S. DoD defines an intelligence source as *"the means or system that can be used to observe and record information relating to the condition, situation, or activities of a targeted location, organization, or individual. An intelligence source can be people, documents, equipment, or technical sensors"* [JDD01].

Hackers' intelligence sources take a variety of forms, including port-scanners and network sniffers. A web-site can be either an intelligence source, or the object of an attack, depending upon how the target uses it. In the IP deception operation, the IP web-site is one of the target's intelligence sources. The target gets information from the site, but he doesn't attack it. Some of the most useful intelligence sources are network-administration tools such as traceroute and ping. Also, any network client can be used as an intelligence source, e.g., its telnet, FTP, and web clients. In social engineering, insiders are used as unwitting intelligence sources. A description of hackers' intelligence sources can be found in books on hacking techniques, such *Hacking Exposed*, by McClure, et al [MSK99]. Section 5.2.1 (page 45) describes McClure's model of the hacking process, and the deception opportunities it presents.

The implemented parts of the deception story **must be observable** by the target's intelligence sources, e.g., a port scan. Otherwise, the target cannot receive the story. Further, the implemented story must be presented in the target's observation arenas, those places he is likely to see. Otherwise, the target will not receive the story. McClure's hacking model also reveals the hackers' primary observation arenas. In the IP deception operation, the story included a

[16] [Bar52]
[17] [Che92], emphasis added

vulnerable FTP server on the IP workstation because it is something the target can see, and is likely to see. Designing the deception story requires an understanding of the target's intelligence sources and observation arenas.

Although the deception story must be observed by the target, it is often advantageous if he only sees the story dimly. When the target cannot examine deceptions closely, he is less likely to detect them. One way to reduce the target's scrutiny is to present the story via his weaker intelligence capabilities, e.g., those that do no provide detail.

Several **potential problems** should be considered when implementing the deception story. First, the deception planner must consider the likelihood and consequences of inadvertently deceiving friendly personnel or systems. The planner may have to forgo some opportunities for implementing the story, lest friendlies be deceived. Secondly, there are four types of failure in communicating the deception story [DH82b]. Information presented to the target can be:

- modified before it reaches the target, e.g., a firewall inadvertently drops packets sent to the target,
- received by the target, but not interpreted as intended,
- received by the target, but dismissed as a deception, or
- not received by the target, e.g., the target does not look in the expected places.

If the target will **verify** the deception story using multiple intelligence sources, then the story should be portrayed through more than one source [JDD96]. For example, to avoid honeypot web-sites, a target can verify web-sites by ensuring they have reasonable host names. The target will use two sources: a port scan to find web-sites, and the DNS to find their hostname. The story must, therefore, take into account all of the adversary's intelligence sources, and the story may need to be made available through all or many of those sources. Single-source inputs that provide the entire story may not be convincing. When portraying things that the target expects to be hidden, multiple conduits can be used, with each providing the target a small piece of the deception story.

A CIA deception study offers **Jones' Lemma:**[18] *deception becomes more difficult as the number of intelligence sources available to the target increases. However, within limits, the greater the number of controlled intelligence sources the greater the likelihood the deception will be believed* [CIA80]. Two related observations can be made: as more intelligence sources are used to portray the story, the likelihood of it being received is increased. Secondly, if the planner can control most of the target's intelligence sources, then the planner lessens the target's access to disconfirming evidence [DH82].

Several factors influence the **choice of intelligence sources** used to portray the story [DH82b,Wha69]: **1)** how plausible the source is, as a means of communicating the story to the target, e.g., if the source is a public web-site, the hacker may be suspicious of confidential information there, **2)** the degree of trust, or credibility, that the target places in the source, e.g., hackers know ping scans are not an accurate source because many computers do not reply to pings, **3)** the planner's control over the source, e.g., one of the target's primary sources is a database, but its administrators do not have resources to support deception operations, **4)** the likelihood that the target will use the source.

[18] Professor R.V. Jones was one of the key figures in British scientific intelligence during World War II [CIA80].

4.1.3.3 Realism

[Deception] stories ought to be as near the "real thing" as is safely possible. WWII deception planner[19]

For each part of the deception story that is implemented, the deception planner will need to determine its degree of realism. The **realism needed** is a function of: **1)** the target's intelligence capabilities, and **2)** the time the target has available to analyze the situation and take appropriate actions [FN95].

Often, minimal realism is needed for deceptions that the target has little time to observe and analyze [FN95]. Simple and cheap simulations can be used; there are no strict security requirements, and there is no need for complex feedback. One can expect that the deception will work, but should not be largely dependent upon its success. A good example of when hackers have little time to observe is during extensive port scanning. When many ports are to be scanned, each scan must be quick, and thus superficial. Such scans are easy to deceive, and the deception is fairly reliable. For example, ping scans can be easily and reliably deceived by fake echo-reply packets.

In general, it is best to design the deception story so that the amount of realism needed is **kept to a minimum**. In creating a lie, the more information that must be provided, the more opportunity there is for the lie to be revealed, and the more difficult it will be to implement the deception. One way to reduce the needed realism is to portray general types of things rather than specific things.

For example, a honeypot computer will be made to look like a network's backdoor for maintenance personnel. The honeypot will be implemented, but not the network, as its existence will be inferred by documents on the honeypot. Creating these documents will be easier if a general network can be portrayed rather than a specific type of network. For instance, to successfully portray a mainframe network, it is easier to portray a general network than a bank's network. Portraying a bank requires specific banking information; some targets may find such a backdoor implausible at a bank, and impersonating a bank may create legal problems or other unintended consequences. In general, the deception story should be designed so that it requires the least amount of detail that is needed to induce the target-action.

The target's ability to examine and scrutinize the story is determined by his intelligence abilities, opportunities and time. From the perspective of simulating computer systems, the target's intelligence ranges between two extremes of scrutiny: **1) superficial probes**, such as ping scans and the examination of servers' banners, and **2) interactive access**, where the target personally works with a system's processing and content, e.g., web-browsing and database queries. Superficial probes can be performed by either the target or an agent, e.g., a worm. When such a probe is received, a computer system can be simulated by simply returning the expected superficial reply, e.g., for a TCP port scan, only the second TCP-Open packet needs to be returned.

The target's *interactive access* is with an application or system program. When deceptively portraying the program, varying degrees of realism are possible. The degree of realism needed depends on the target's degree of interactive access. If the target closely examines the program and its content, a very high degree of realism will be needed. It will

[19] [DH82b]

probably be necessary to implement the actual program (e.g., an Apache web-server) rather than simulate it. Implementing actual systems can be very expensive, especially if detailed deceptive-content is required, e.g., meaningful fake web-pages.

Honeypots are categorized according to the degree of hacker interaction that they support [Spi03]. Greater interaction requires greater realism, which incurs greater costs. For example, a honeypot server is used to detect hackers' attempts to login via weak passwords. The server can easily be simulated if it just presents a login interface and refuses all logins. Simulating accepted logins will clearly be more difficult.

Deceptive content is not only expensive to create, but it must continually be updated to be kept current. For example, a honeypot mail-server will need deceptive content if the deception is to be maintained after the target breaks into the server. A very high degree of realism is required if the target includes insiders, as they know the mails' context. To keep the deception current, new emails must continually be created. Manually creating a plethora of realistic email is prohibitive under most circumstances. Symantec's ManTrap can generate fake e-mail for honeypot mail-servers, but users will need to determine if its realism is adequate for their deception operation.

Although it is very expensive to create deceptive content from scratch, it can be very easy to create deceptive content by **mirroring** an existing system. For instance, a honeypot workstation can be created by mirroring a real workstation. If necessary, the copy can be altered to protect sensitive information. Mirroring is not practical under all circumstances, though. The deception may be revealed if the target discovers both the real system and its copy. Also, removing sensitive information may be difficult, and there is a risk of accidentally leaving some on the copy.

The deception story's realism is also determined by the target's ability to recognize truth and falsehood, and thereby **detect deception**. An insider's knowledge of systems and operations greatly aids him in detecting deception. Also, targets can verify a story via multiple sources and communication channels. Verification can be performed intentionally, to detect deception, or unintentionally during the course of hacking. To ensure the deception story is convincing and realistic, the story may have to be portrayed using several of the target's intelligence sources, as described in the previous section. In summary, the amount of realism required depends upon how closely the target will examine the deception, and also, his ability to detect it.

4.1.4 Operational requirements

Principles and Techniques for Building the Deception Story			
4.1.1 Designing the story (pg. 18)	***4.1.2 Presenting the story to the target*** (pg. 21)	***4.1.3 Implementing the story*** (pg. 24)	***4.1.4 Operational requirements*** (pg. 30)

The deception story is deployed in a dynamic environment: the story has timing requirements, its environment changes, and the operation may not go as planned. The deception story's *operational requirements* are concerned with accommodating such dynamics. These

requirements are presented here, along with principles and techniques for fulfilling them:

- time-related requirements,
- adaptability and reuse (page 32), and
- potential problems (page 32).

4.1.4.1 Time-related requirements

The deception story is portrayed to an adversary over time. This section discusses some of the deception story's time-related requirements. For example, incident-response counters an active attack, so it has **deadlines** for the deception operation's deployment, the target-action, and the action's exploit. The deployment deadline will limit the time available for developing the story, and the deadline for the target-action will limit the types of stories that can be used. A deception story is expected to unfold in a particular **order**. The order is contingent upon the target's activities, and their occurrence may be uncertain. In the IP deception operation, the target is expected to scan the IP workstation, discover the honeypot FTP server, and then break into it.

It can be helpful to lay out the deception story's **timeline**: 1) the target's likely courses of action, before, during, and after he engages with the deception operation, 2) when the target encounters the story and the time it takes him to collect and analyze intelligence about it, 3) when elements of the story must be displayed, and once displayed, when the elements must be updated to reflect current conditions, 4) how long the target interacts with the deception, 5) when the target performs the intended action, 6) when the target's action will be exploited, and 7) when the target might uncover the story's deceptions. The timeline can use particular times (e.g., a time of day), or just a relative sequence of events, e.g., a port-scan precedes a buffer-overflow attack.

Timing requirements are affected by the **tempo** of CND and the target's offense, e.g., for the tempo of incident response is much faster than the tempo of long-term static defenses. During incident response, the tempo of CND may afford little time for developing and deploying the deception operation. However, the faster the target's offensive tempo, the more vulnerable he is to deception. An increase in the target's tempo will generally decrease his time for intelligence and for scrutinizing deceptions.

It can be helpful to know when the target will be **most vulnerable** to deception. Naiveté makes one vulnerable to deception. Generally, the less the target knows about the network, the more vulnerable he is to deception. In the IP deception operation, the target is deceived and detected before he breaks into the IP LAN (Figure 2, page 18). While he is trying to get into the LAN, the hacker is naive, and he can be deceived by a simple ruse. Another important timing requirement is **how long** the target must be deceived. If the deception operation's objective is to occupy the target and waste his time, then the deception must be believable for long periods. If the deception operation's objective is just to detect attacks, an easily discovered ruse can be effective, as long as the attack gets detected. An example of such a ruse is honeypots that superficially impersonate servers. To keep the target from uncovering the deception story, it can be helpful to **minimize the target's interaction** with the story, and especially with its deceptions.

A CIA deception study offers another time-related deception technique [CIA80]. The **sequencing rule** states that: *deception [operations] should be sequenced so as to maximize the*

persistence of the incorrect hypothesis(es) for as long as possible. In other words, "red-handed" activities should be deferred to the last possible instant. The deception planner's red-handed activities are those that reveal the deception to the target. For instance, the deception is often revealed when the deception planner exploits the intended target-action. In the IP deception operation, the target will learn about the deception when evidence from it is used against him, e.g., to prosecute him, or to thwart his attacks. By deferring such red-handed activity, the honeypot can continue to detect attacks by the target, and by his accomplices. Another way of following the sequencing rule is to defer risky portions of the deception story that might reveal the ruse.

4.1.4.2 Adaptability and reuse

The deception story should be **flexible,** so it can be **adapted** to a changing environment and to unforeseen circumstances [DH82,USA78]: **1)** as network technology and operations change, so must the deception story if it is to be believable. **2)** A deception story that portrays current or on-going situations must be continuously revised. This is one of the difficulties of maintaining fake content on servers, e.g. the fake email on honeypot SMTP-servers. **3)** A deception story may need to be revised based on feedback about the target's actions. Feedback can reveal unexpected problems or failures. It can also reveal unexpected successes that the deception planner can exploit.

In the IP deception operation, the honeypot server uses FTP. The FTP login-interface changes little over time. However, the honeypot server displays the banner of a make and version that is vulnerable to attack. Eventually, the banner will need to be updated to display a more recent vulnerable server.

Often, the deception story will need to be **reusable**, as it would otherwise be too costly or ineffective tactically. Some deceptions will need to work repeatedly for the **same target**. In the IP deception operation, the honeypot FTP server will identify itself as a version with an exploitable vulnerability, although it won't have the vulnerability. Consequently, the target's attacks will fail, however, they should not fail in a way that reveals the deception. Further they should fail in a way that encourages the target to continue attacking, as he may not leave sufficient evidence on his first attack.

Many deceptions will ultimately be revealed to the target, but they will need to work against **subsequent targets**. In the IP deception operation, the honeypot deception will be revealed when its evidence is used to prosecute the target. However, the deception can be reused if subsequent attackers are unlikely to know about prior attackers. In contrast is a honeypot web-server that resides on a famous network's DMZ. If a script kiddy discovers the deception, he's likely to boast of his discovery on hacker forums, thus revealing the deception to the world.

Reuse presents a difficulty for developing deception-based products, such as commercial honeypots. Being part of a product, the deception is publicly known. However, the deception cannot be identifiable when it is deployed. To solve this problem, the ManTrap and Specter honeypots support the creation of customized deceptions that are not easily identified.

4.1.4.3 Potential problems

Deceptions may **fail** for many reasons [JDD96]. The target may not receive the story, may not believe the story, be unable to act, be indecisive when the story is believed, act in

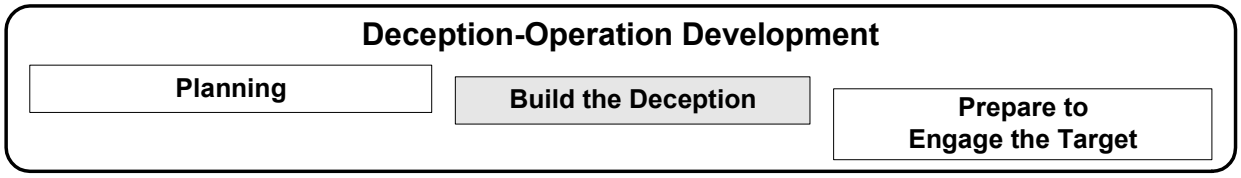
unpredicted ways, or he may discover the deception. Earlier sections described some of the deception operation's potential problems (sections 3.5 on page 15, and 4.1.3.2 on page 27). This section addresses some additional potential problems related to the deception story.

It can be helpful to analyze the deception story's **vulnerabilities to discovery** by the target: **1)** the deception story may have single points of failure, where the discovery of a single deception will reveal the whole deception operation, **2)** a highly-effective deception technique is weaving the deception story into the real network, however, unexpected conditions in the real network may reveal the deception [JDD96], **3)** deceptions that are easily subjected to the target's intelligence will be especially vulnerable to discovery.

It can be helpful to anticipate what the target will do **after uncovering** a deception. His discovery of deception may terminate the deception operation or prevent it from being reused. Also, if the target suspects deception, but does not find it, his uncertainty may cause him to act unpredictably, or even dangerously. For instance, when a hacker is uncertain about the evidence he's left on a computer, he can simply erase the entire hard drive.

Another type of potential problem are the **unintended consequences** following both successful and unsuccessful deception operations. These consequences include both target actions and third-party actions. In the IP deception operation, the target will eventually give up his attacks against the honeypot FTP server. A determined target may then try to break into other computers on the IP workstations' subnet, in the hope of finding another path to the IP database. These attacks will need to be anticipated and preparations made for them.

4.2 Feedback



Deception Maxim #8, The Importance of Feedback: A scheme to ensure accurate feedback increases the chance of success in deception. CIA deception study[20]

The crucial effect for which deception aims occurs in the inaccessible mind of the opponent. Daniel and Herbig[21]

Build the Deception
Deception story
Feedback
Termination plan

Feedback is the second component addressed for building the deception. The above quotes indicate the importance of feedback. Though, it should be noted that Daniel and Herbig overstate its importance. Deception's most crucial effect is the target action, and fortunately, this can often be observed.

Feedback mechanisms are a major component of the deception operation. For the deception operation to be successful, not only must the target be deceived, but there must be

[20] [CIA80]
[21] [DH82b]

adequate feedback for monitoring the target's reception of the deception and his reaction to it. Two types of feedback are used in deception operations [JDD96]. **Operational feedback** identifies what deception information is reaching the target. **Analytical feedback** identifies what actions the target is taking because of that information. This section describes: 1) feedback's uses, 2) the sources of feedback, 3) deceptions that require little feedback, and 4) the interpretation of feedback.

In summary, for CND deception operations, the availability of feedback varies between extremes. If the hacker operates in a **controlled** environment where feedback mechanisms can be installed, then almost all of the hacker's activity can be monitored and recorded, including activity conducted through SSH connections. Honeypots create such a controlled environment, and several honeypot systems have been developed for obtaining feedback. In contrast, if the hacker operates freely in the network, and feedback is obtained via **ad hoc investigation**, then discovering the hacker's activity can be very difficult and time consuming. Further, if the hacker uses covert and encrypted channels, or if he is otherwise stealthy, then ad hoc investigations may produce little or no feedback.

Feedback seeks to know the target's actions and his thinking. By nature, a weakness of deception is that the deception planner can only speculate about the target's thoughts and the deceptions' influence over them. However, there are principles that can be used to interpret feedback and to avoid misperception. Given the cost and uncertainty of obtaining feedback, the deception operation should be designed to minimize the amount of feedback needed, e.g., by using deceptions that require little feedback.

4.2.1 The uses of feedback

Timely target feedback is an essential element of deception operations [DH82,FN95]. Feedback makes it possible to: **exploit** the intended target-action, exploit unforeseen target vulnerabilities, and increase the operation's lifespan. Feedback also makes it possible to **detect problems and failures** (described earlier in sections *3.5* on page 15, and *4.1.4.3* on page 32). For instance, the target may not receive the story; he may discover the deception, or he may act unpredictably. By detecting such problems, it may be possible to make adjustments and continue the operation (described earlier in section *4.1.4.2* on page 32). Alternatively, problems may require the deception operation to be terminated. Feedback on the target's reaction to the deception is critically important if the planner's own position would be vulnerable if the target is not successfully deceived.

Yet another use of feedback is detecting the target's **counter-deception** (described earlier in section *3.5* on page 15). If the target discovers the deception operation, he can use its feedback channels to portray his own deception story [DH82]. When feedback channels can be exploited like this, they should be used cautiously and examined critically. Concealing the feedback channels from the target can prevent him from exploiting them.

4.2.2 The sources of feedback

The opportunities for using deception are strongly influenced by the availability of feedback. As will be described here, more feedback is available for insiders than for remote hackers. Also described are the opportunities and limitations of two techniques for obtaining feedback: incident response (IR) investigation and honeypots.

Usually, the only sources of feedback about **remote hackers** are surveillance of their network connections and surveillance of the hosts to which they connect. Usually, the remote hacker cannot be physically observed, nor his communication with other networks. If the remote hacker's actions are few or stealthy, there may be little or no feedback available. In some instances, deception's usefulness will be hampered by limited opportunities for feedback. There is more opportunity for getting feedback about **insiders**, as they can be physically observed, and all of their communication can potentially be monitored, including email and telephone calls.

The tools and techniques for **IR investigation** can be useful for obtaining feedback. For example, commercial IR tools for recording computer use include hardware keystroke-loggers and *spyware* that monitors computer users' activity. In general, IR investigation is very costly and time consuming.[22] For deception operations, it is not practical to just allow the target to hack, and then use IR investigation as an ad hoc means for obtaining feedback. Instead, the use of IR tools and techniques should be designed into the deception operation, to detect and monitor the target's actions.

An FBI sting provides an example of using IR investigation tools in deception operations [Lin01]. The FBI needed evidence of two Russians' hacking, so the FBI created a fake computer-security company, and through it, they offered the hackers a partnership in the business. The hackers visited the company's U.S. office, and while there, they used the company's computers. From these computers, the hackers logged-in to their own computers in Russia. However, the FBI had installed keystroke loggers on the office computers, and they recorded the hackers' passwords. The FBI used the passwords to access the hackers' computers and thereby obtain the evidence they sought. In the deception operation, the keystroke logger served two purposes: it provided feedback, and it was used to exploit the intended target-action.

Several books describe IR investigation tools and techniques [Cas00, KH02,MP01], and a summary of the techniques is provided here. The primary means for **monitoring and recording** hacker activity include: 1) network monitoring, 2) host monitoring (e.g., spyware), 3) the logging capabilities of hosts, networking systems, security systems, and servers, and 4) checksums for files (e.g., Tripwire). The primary means for **discovering** evidence of the hacker's activity are: 1) examining file systems for files created, modified, or deleted by the hacker, and 2) examining running processes and active logins. IR investigation requires an understanding of hacking tools and techniques, and many books describe them [ANO97,MSK99].

Another source of feedback is **people**. System administrators or end users may see evidence of the target's actions. Deception planners can work with appropriate personnel to make them aware of the type of information that is being sought. However, disclosure of the deception operation should be balanced with the consequent risk of the operation being revealed to the target.

Feedback is a major component of **honeypot** design and practice. Lance Spitzner's honeypot book describes five elements of feedback collection, and they are summarized here: data capture, data collection, alerts, response, and analysis [Spi03]. **Data capture** involves recording the hacker's activity. The amount of data to be captured varies, depending upon the honeypot's use. In general, Lance recommends, *"configure your honeypot to capture as much*

[22] For instance, in an IR experiment, it took investigators, on average, 34 hours to investigate one half-hour of hacking [Dit01].

information as possible. You will be surprised how often the seemingly obscure information turns out to be of great importance." Honeypot data-capture techniques include those used in incident response, as well as techniques specifically for honeypots. Data capture is a key selling-point for commercial honeypots like ManTrap and Specter. The Honeynet Project has developed about a dozen freeware tools for honeypot data capture [Hon04]. Sabek is their primary tool, and its purpose is to capture all of the hacker's activities, including keystrokes, file uploads, and passwords. Sabek covertly sends the data to a server where it is collected. Both Sabek and ManTrap capture keystrokes at the kernel level, making it possible to record encrypted connections such as SSH.

Honeypot **data collection** involves logging and managing the large volume of data that is often captured. The Honeynet Project has developed tools for collecting and correlating data from multiple honeypots. Honeypot **alerts** should be reliable, prioritized, and provide clear and critical information. **Responding** to the hacker's activity requires remote access and control of the honeypot. **Analysis** of the feedback varies, depending on the honeypot's purpose, e.g., detecting attacks or collecting intelligence about hackers.

4.2.3 Deceptions that require little feedback

The need for feedback varies among deceptions. As possible, the deception operation should be designed to minimize the amount of feedback needed. When a deception's effect on the target is certain and **predictable**, then feedback can be simple and of low concern. For example, camouflage clothing, and fake ping replies both have predictable affects on the target's observation. Predictable deceptions tend to be simple, and deployed in environments where the target has little time to scrutinize them [FN95]. The deception may predictably effect the target's intelligence-collection abilities, or his decision-making processes. For instance, camouflage clothing has a predictable effect on the human eye, when the camouflage is viewed at a distance and under the pressures of battle. Fake ICMP echo-replies have a deterministic effect on the output of the hacker's ping scan.

Another case in which feedback is of low concern is when the deception's effect on the target **does not alter** the deception planner's course of action. For example, a diversion can be used to draw attention away from a committed action. This technique is common in sports, as with fake handoffs in football. In computer security, diversions may be helpful for battling active hackers during incident response

4.2.4 The interpretation of feedback

When [a remote client] fetches the 'passwd' file during a long session, it is not always clear that he has evil intentions. Sometimes they are just checking to see if any transfer will work. Bill Cheswick[23]

This section presents principles that can be used to interpret feedback and to avoid misperception. Feedback is used to learn how the target receives and accepts the deception story, and the actions he takes as a result. The target's **actions** can be observed, but what goes on in his **mind** can only be inferred from his actions [DH82]. The crucial elements of the deception operation that reside in the target's mind are: **1)** his perception and acceptance of the deception story, and **2)** the story's effects on his decision-making process.

[23] [Che92]

There are indicators that reveal the deception's effects on the target, and one of the best indicators is **changes** in the target's actions [JDD96]. The target's **intelligence interests** and activities can provide indications of the receipt of deception information. The target's **reaction to deception** can provide indicators of whether he is believing the deception story. For instance, a target is deceived by a honeypot, but later avoids all other honeypots, indicating he has discovered the deception and learned how to detect it. On some occasions it is possible to intercept the target's **personal communications** [FN95]. They can provide useful feedback on the target's thoughts, as well as his actions. For example, the Honeynet Project is a consortium that researches honeypots and that gathers intelligence on hackers. Their honeypots have captured extensive hacker communication, and their book records over a hundred pages of hackers' IRC communication [Hon01].

Feedback can easily be misinterpreted. There are human tendencies for misperceiving the motives and actions of others, and the deception planner is vulnerable to such misperception when analyzing feedback. (Misperception was discussed earlier in section *4.1.2.1* on page 21.) Jervis' hypotheses on misperception reveal ways that feedback can be misperceived [Jer68]. By understanding these misperceptions, the planner will be less likely to make them.

As cited earlier, Jervis' **Hypothesis #14** states: *actors tend to overlook the fact that evidence consistent with their theories may also be consistent with other theories*. This hypothesis is especially applicable to the analysis of feedback [CIA80]. The deception planner receives feedback about the target's actions. From the feedback, the planner forms speculative explanations about how the deception story is perceived by the target and how it is influencing the target's thinking. Usually, several plausible explanations will be consistent with the feedback. Forming the most likely explanations requires skill and wisdom. For example, in the IP deception operation (section 4.1, page 17), a target might visit the IP department's web-site and then later attack the FTP honeypot. Although it appears the attack was motivated by the deception story, it may be just opportunistic and unrelated to the story.

Hypothesis #11 states [24]: *when an actor tries to influence others, and the others behave in accordance with the actor's desires, the actor will tend to overestimate the degree of his influence; but when the others' behavior is undesired, it is usually seen as being caused by the others' own personal motives*. Simply put, the actor will tend to take credit for success and to deny responsibility for failure. The first part of the hypotheses can be explained by the actor's desire for gratification, and, conversely, the second part can be explained by a reluctance to admit failure. Both parts of the hypothesis can also be explained by the actor's familiarity with his influence on the other and his lesser familiarity with other influences.

The hypothesis is illustrated by the prior example of the IP deception operation. The planner is likely to take credit for motivating the attack on the honeypot when, in fact, the attack is opportunistic and unrelated to the story. Conversely, if the target does not attack the honeypot, the planner may blame the target's obtuseness, rather than acknowledge the story's ineffectiveness.

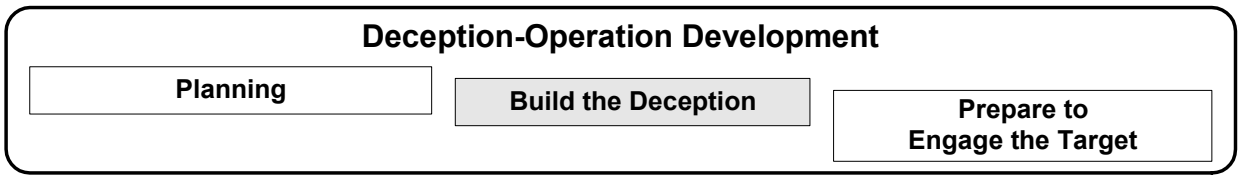
Hypothesis #9 states: *actors tend to see the behavior of others as more centralized, disciplined and coordinated than it is*. Jervis offers some causes for this phenomenon: **1)** when patterns are perceived, there is a tendency to attribute more causes than there really are, **2)** actors

[24] A subset of Jervis' *Hypothesis 11* is presented here.

are hesitant to admit (or even see) that particular incidents cannot be explained by their theories, **3)** actors see others as more internally united than they in fact are, and **4)** actors generally overestimate the degree to which others are following a coherent policy. When analyzing feedback about hackers, the hacking process' uncertain path should be kept in mind. Much of a hacker's activity involves exploring and investigating, making his course of action uncertain, and more opportunistic than planned. Also, for many hackers, youthful naiveté or muddled thinking can result in actions that are unpredictable, inefficient, or nonsensical.

A good example of this hypothesis is provided by Cliff Stoll's famous investigation [Sto89]. In tracing a hacker's connections, Cliff learned the hacker used the Westar satellite one day and the Comsat satellite the next. Cliff initially thought he was dealing with "*one elusive hacker—switching communication satellites from day to day.*" However, Cliff soon learned he had overestimated the hacker's skill-- the internet used both satellites, and the hacker had no choice in which one carried his data.

4.3 The termination plan



Planning the termination of a deception operation requires the same care and attention to detail that went into planning the deception's execution. U.S. Joint Forces deception manual[25]

Build the Deception
Deception story
Feedback
Termination plan

The termination plan is the third component addressed for building the deception.

A **plan** is needed for terminating the deception operation after the deception story has been received by the target [JDD96]. Termination planning ensures the controlled and orderly release of information related to the deception. The plan includes termination for both successful and unsuccessful outcomes of the operation. The plan should identify what information about the deception will be disclosed, what will be hidden, and when. A cover story may be needed to prevent disclosure of the deception. Deceptions for computer security are, by nature, amenable to cover stories, as will be explained.

When the deception operation is terminated, the operation's existence, and its deceptions, usually need to be **hidden** from the target. This allows reuse of the deception story and its deception techniques. If the deception has been disclosed to the target, termination planning can reduce the amount of disclosure. For instance: **1)** the target may prematurely discover the deception, making termination necessary, and **2)** for successful deception operations, the deception is often disclosed when the target-action is exploited. For example, when an attack is detected via deception, the deception may be revealed if the attack is promptly blocked.

[25] [JDD96]

4.3.1 Using cover stories to hide the deception

A **cover story** can be used to hide the deception, during termination. Cliff Stoll's investigation provides a good example [Sto89]. Cliff spent months secretly monitoring a hacker's extensive penetration of military computer systems. On one occasion, the hacker started downloading sensitive data, and Cliff was compelled to terminate the hacker's connection. However, if Cliff promptly and abruptly terminated the connection, he would risk revealing his surveillance, and that would be a major tactical loss. Cliff spontaneously created a clever cover story that induced the hacker to terminate the download himself. By jingling his keys across the communication line, Cliff created line noise that sporadically corrupted the data transfer. The download became so slow that the hacker gave up on it.

Termination cover stories are an extension of the deception story. They can be preplanned for the deception operation's successful and unsuccessful outcomes. When termination is unforeseen, ad hoc cover stories can be used. Accommodating ad hoc cover stories is another reason to make the deception story flexible and adaptable, as described in section 4.1.4.2 (page 32) [USA78]. Computer-security deceptions are often amenable to termination cover-stories, especially if termination can be attributed to other security operations, or to the dynamics of normal system-operations, as will be described.

Usually, hackers expect that they may be detected or blocked by a variety of network defenses, at any time. The defenses are often concealed, and they usually don't involve deception. The hacker's expectation of being detected, and of being denied access, can be used to create cover stories that attribute the deception operation's counter-actions to other security measures.

For example, in the IP deception operation (section 4.1, page 17), the target is detected when he accesses the honeypot FTP server. If the target learns his attack was detected, he is likely to attribute the detection to standard security measures such as the routine examination of log files or misuse detection by IDSs. In the worst case, he will only suspect deception to be one possibility among several. When the deception story portrays vulnerable systems, as honeypots do, actions that thwart the hacker can often be attributed to other security measures.

Termination cover-stories can also be based on the dynamic nature of the network. In most networks, systems are continually being added, removed and changed, in support of on-going operations. Also, systems are regularly taken down for maintenance, and there are a steady supply of errors that temporarily disable systems. When termination involves shutting-down systems, it may be possible to portray the shutdown as a part of normal operations.

For example, a honeypot impersonates a temporary network backdoor for use by network maintenance personnel. If the honeypot has to be shut down, the target may perceive the backdoor's closure as part of its temporary existence. When Cliff Stoll degraded network performance by jingling his keys, the hacker likely attributed the problem to normal network problems.

4.3.2 Revealing the deception during termination

Controlling the exposure of a deception, or deception operation, is unavoidable at times [JDD96]. When disclosure is unavoidable, it may be possible to use the deception's disclosure to advantage. For example, the network-wide use of deception can be exaggerated. The secrecy

that normally surrounds deception operations helps to make such exaggeration believable, as the hacker expects little evidence [Han85]. One successful deception, well known by the target, may be sufficient evidence for making the exaggeration believable. Exaggerating the use of deception can deter hacking, and it can impair the hacker's decision-making process by making him more uncertain.[26] In addition, even when disclosure is avoidable, it may be useful to intentionally reveal the deception during termination, in order to deter or impair hacking.

Another principle for terminating the deception operation is the sequencing rule that was described earlier (section 4.1.4.1 on page 31). It states that deception operations should be sequenced so as to maximize the persistence of the incorrect hypothesis(es) for as long as possible. In other words, "red-handed" activities should be deferred to the last possible instant.

[26] Using deception to impair the hacker's decision-making process is discussed further in section 5.3 on page 48.

5 Uses of deception in computer-security defense

This chapter provides an overview of the uses of deception for computer-security defense. It focuses on perceptions and actions that can be induced in hackers and used to advantage. Many examples of actual deception-measures are given. In early 2003, we conducted a survey of the use of deception in computer security. The examples represent most of the deception uses found in the survey. However, only noteworthy honeypot examples are included here as their uses are ubiquitous.[27]

There are various models that explain how computer-security works, e.g., defense can be modeled by the categories of detection, prevention, response, and intelligence. Such models of computer security can be very useful for identifying opportunities for using deception. This chapter presents several computer-security models and how deception can be used to enhance them. The models are listed in the table of contents, below. Usually, deception is not a primary computer-security effort, such as detection, or prevention. Rather, deception augments or enhances primary computer-security efforts.

5.1 Models of computer-security defense

Models for Illustrating the Uses of Deception		
Computer-Security Defense	***The Hacking Process***	***Hacker Decision-Making***

Fundamentally, deception can be used:

1) **defensively,** to enhance or provide computer-security, e.g., steganography[28] provides confidentiality, or
2) **offensively**, to thwart a particular aspect of hacker activity, e.g., honeypots such as LaBrea can thwart port-scanning.

For a specific security measure, deception can be: 1) its **primary** security mechanism, e.g., as in honeypots, or 2) a **supporting** security mechanism, e.g., hiding a keystroke logger.

[27] Additional information on honeypots is provided in chapter 6 (page 53).

[28] Stegonography is the hiding of secret messages within another seemingly innocuous message or carrier [JDJ01]. For example, text documents can be hidden within digital pictures, and without noticably altering the image.

There are **four basic categories** of computer-security defenses: detection, prevention, response, and intelligence [Spi03]. Deception can be used to enhance each of them:

1) **Detection:** One of the most useful applications of deception is detection. Many deceptions are temporary, and this limits the use of deception for computer-security's long-term static defenses. However, for detection, a short-lived ruse can be sufficient, provided that detection occurs before the ruse is discovered. Deception is especially promising for detecting hackers when they collect intelligence, as hackers can be lured by deceptive intelligence-indicators that are simple and inexpensive; also, the hacker is naive about the network when he collects intelligence, and his naiveté makes him vulnerable to deception. For example, port-scans can be effectively deceived by honeypots that superficially emulate servers. Specter is one such honeypot, and it is reportedly easy to use [Spi03]. Examples of detection follow, and many more examples are given in the upcoming sections on the other models:

   a) Detection is one of the primary uses of honeypots, as up-coming sections will describe. **b)** Unauthorized access can be detected by placing identifying information among data that a hacker is likely to access. The identifying information is sometimes called a *honeytoken*. Unauthorized access is detected by devices that watch for the honeytoken. For example, a password file can contain a user ID and password for a fake account. An alarm is set-off when the account is accessed. **c)** A similar technique is used to detect unknown viruses. The security-product RegRun creates "bait" files (e.g., winbait.exe) that are likely to be infected by viruses. If the bait-files change, the user is alerted. **d)** *Digital watermarking* is used to protect copyrights and intellectual property. Digital watermarks are commercial applications of steganography, and they are used to trace, identify and locate digital media across networks [JDJ01].

2) **Prevention:** Deceptions are often not reliable enough to be used by themselves as prevention measures, as ruses tend to be temporary and their effects uncertain. This is especially true when more permanent and certain solutions are available, such as encryption. However, temporary and uncertain deceptions can be used to enhance prevention measures. They can prevent or degrade attacks by causing them to be: less effective, less likely to occur, less certain to succeed, or more difficult to accomplish. Deception can also prevent or degrade hackers' intelligence collection and analysis, as will be described with the up-coming incident hacking-model. Examples of preventing attacks are listed here:

   Honeypots can be used to help prevent attacks by: **a)** wasting hackers' time on expendable resources, **b)** impeding the hacking process, **c)** providing time for response, and **d)** deterring attacks by reducing the hacker's will to attack. However, in a detailed analysis of honeypots' ability to prevent attacks, Lance Spitzner concludes that honeypots add little value to prevention [Spi03]. He thinks that the resources would, instead, be better used on traditional prevention measures. Lance does note that honeypots may be useful in preventing attacks on high-value resources. Lance's analysis seems most applicable to single-computer honeypots. In contrast, Honeyd and LaBrea are honeypots that create numerous imposter systems, making it difficult to locate real systems.[29] Honeypots such as these may prove useful for preventing attacks and for thwarting intelligence efforts.

   Deception can be used to **enhance authentication**. There is a long-known ruse for

[29] Fred Cohen reports similar prevention effects in his deception research [CMS01,Coh00].

deceiving brute-force login attacks. When an invalid password is entered, the user is prompted for another password. After a few failed attempts, the login process will continue to prompt for passwords, but it ignores them. The brute-force attack is neutralized and occupied indefinitely, thus wasting the hacker's time, detecting the attack, and providing time for response.

Deception can **provide confidentiality** by hiding. Although the effectiveness may be uncertain, the deception can still be useful. **a)** Steganography hides the existence of data. **b)** *Chaffing-and-winnowing* provides confidentiality via deception [Riv98]. A stream of clear-text packets are sent. Some of the packets are real, but the others are imposters. Packet checksums are valid for the real packets, but invalid for the imposters. The sender and receiver share a secret key that is used to calculate the checksums. **c)** Deceptive hiding is used to conceal the existence of computer-security systems that perform secret surveillance on computer users. Such systems include *spyware* and *anti-spyware.*[30] **i)** Spyware monitors and records a computer user's activity; spyware is described in the next section. **ii)** Anti-spyware is used to detect spyware. Products include SpyCop and SpyGuard. Anti-spyware hides from spyware.

Deceptive showing is used to **protect privacy**, a type of confidentiality.[31] **a)** The ident service reveals the user and process that are associated with a network port. Some servers, e.g., IRC, require ident responses from their clients. There are several ident servers that provide fake responses. **b)** Spam-bots are tools used by spammers to search web-sites for email addresses. Sugarplum is an automated spam-bot poisoner that feeds large quantities of realistic and enticing, but otherwise useless, data to spam-bots. The intention is to foul spammers' databases, and to discourage the return of spam-bots to the victim web-site.

3) **Response:** Responding to an incident involves: a) detection, b) investigation, c) isolation and containment, and d) network monitoring [MP01].[32] We'll refer to these activities as *response*. The environment of response presents unique and promising opportunities for using deception: **a)** for an on-going incident, response is a battle with a specific person or small group. In this environment, deceptions can be designed to exploit individual hacker's vulnerabilities, and to avoid their strengths. For defensive measures other than response (e.g., detection), deceptions are not usually designed to counter individuals. Instead, they counter hackers in general, or broad groups of hackers such as script-kiddies. **b)** Incidents tend to be relatively-short one-time events, and there is a high degree of interaction between the hacker and responder. This provides opportunity for using deceptions that are short-lived or non-reusable. **c)** Response addresses a real and present adversary, and he brings high risk and potential loss. Relative to non-response conditions, deception can be very appealing to the defender due to his weakness and desperation, and his willingness to spend, and do, whatever is necessary to prevail. **d)** The ad hoc and creative aspects of response are

[30] Network intrusion-detection systems often hide by using receive-only network connections. This hiding is more likely to be **denial** than deception.

[31] Two of the best-known privacy mechanisms use denial instead of deception. Onion routing is an anonymous routing protocol [RSG98]. Chaum uses a device called a mix, to send untraceable email [Cha81]. Onion routing and mixes hide by using encryption and by avoiding correlation vulnerabilities.

[32] The category of response consists of incident-response and prosecution. Incident response is addressed by this paper, but not prosecution.

conducive to using deception. A famous computer-security incident shows how useful deception can be for response:

Deception was pivotal in solving Cliff Stoll's famous investigation [Sto89]. Cliff spent four months tracking KGB-sponsored hackers who were breaking into U.S. military computers. By planting fake documents on various subjects, Cliff learned the hackers were seeking nuclear secrets. However, Cliff had to get the hacker's physical address, and his earnest efforts to work with the FBI, CIA and NSA were proving futile. He was facing defeat and the loss of months of work. In this desperate state, Cliff's girlfriend suggested a ruse that was pivotal in solving the case. A fake internal government-memo was planted on a computer, offering nuclear secrets via postal mail. The KGB took the bait, and the FBI apprehended the KGB's agent at the receiver's postal address.

Software decoys are a deception-measure used for response. These decoys are embedded within real applications, e.g., a working file-server. The decoy uses standard intrusion-detection techniques to detect attacks. Once an attack is detected, the decoy responds deceptively, e.g., by simulating a successful attack. The deception is used to gain information about the attack. Also, the deception can limit the attack by introducing delays or by re-routing it to a honeypot. Software decoys are being researched at the Naval Postgraduate School, and prototypes have been developed [RMA02].

Response uses computer-security measures from the three other categories: detection, prevention, and intelligence. These categories' uses of deception are described in the present section, and they are largely applicable to response. The upcoming section on the incident hacking-model describes additional deceptions that are applicable to response.

4) **Intelligence:** Deception can cause hackers to reveal their capabilities, intentions, and actions. Deception can also be used to protect intelligence-collection systems, e.g., to hide honeypots and spyware.

Current intelligence describes the present situation, and *basic intelligence* describes long-term conditions. Honeypots can be used for both current and basic intelligence. The Honeynet Project is a consortium that produces intelligence on hackers, gained mostly from honeypots [Hon01,Spi03]. They find that honeypots provide a number of intelligence uses: **a)** as an early warning system for similar real systems, **b)** to capture step-by-step attack information, **c)** to read encrypted sessions, by intercepting keystrokes, **d)** to observe hackers' activity after they have broken in, in order to: discern motive, identify accomplices, and identify other compromised systems, and **e)** to accurately distinguish hacker activity amidst the overwhelming volume of legitimate activity.

Current intelligence is produced as part of the response and detection categories, and their use of deception is described above. The upcoming section on the incident hacking-model also describes deceptions for obtaining current intelligence. Counter-intelligence involves obtaining intelligence about the hacker's intelligence activities. Deception is useful for counter-intelligence, as described above for detection.

5.2 Models of the hacking process

Models for Illustrating the Uses of Deception		
Computer-Security Defense (pg. 41)	***The Hacking Process*** (pg.44)	***Hacker Decision-Making*** (pg. 48)

Deception can be used to thwart hacker activity. These uses are shown within the context of two models of the hacking process: the incident hacking-process and the single attack.

5.2.1 Model of the incident hacking-process

The hacking process for computer-security incidents is shown in Figure 3; it applies primarily to unauthorized access over a network. From the hacker's perspective, each step of the process involves chiefly offense, defense or intelligence, as labeled in the figure.

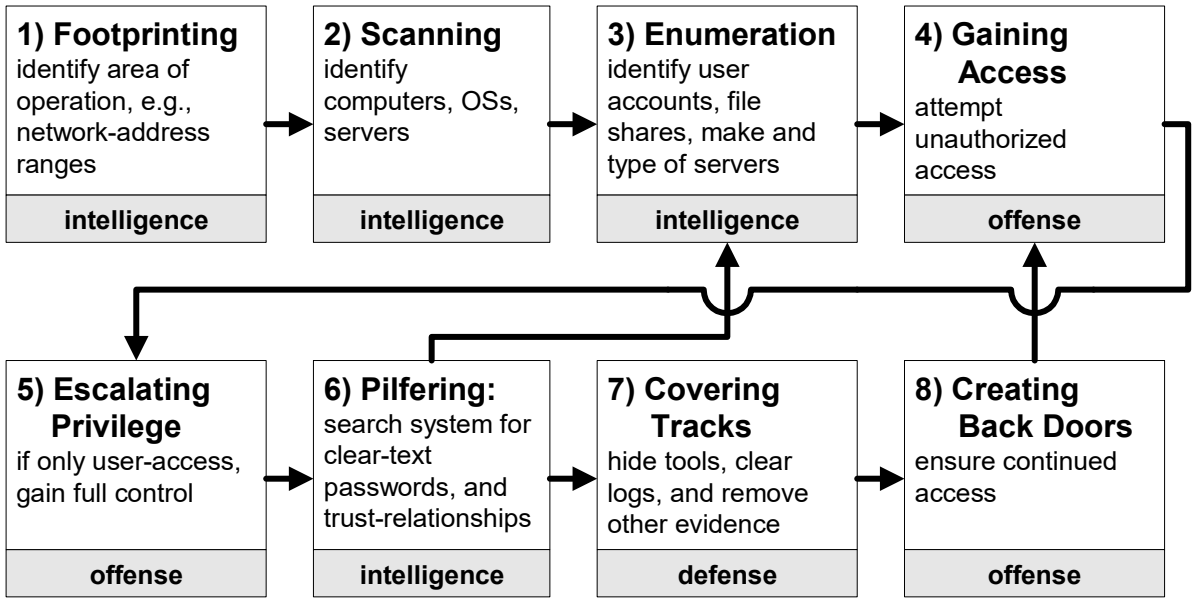


Figure 3 : Model of the incident hacking-process (adapted from *Hacking Exposed* [MSK99])

There are opportunities for deceiving the hacker, in each step of the process:

1) **Footprinting: a)** Hacker's often use publicly-available information to learn about an organization and its network. False open-source information can be provided, e.g., on web-sites, to detect and confuse footprinting. **b)** Deception has been incorporated with networking systems to misrepresent topologies: *Fake AP* is a tool that creates thousands of fake wireless access-points, for the purpose of confusing hackers and hiding true access points. Various tools have been developed for deceiving traceroute. They log traceroutes and create fake hops, presumably to detect and confuse hackers and to waste their time.

2) **Scanning:** Deception can be very useful against scanning because: **a)** scanning techniques and tools are few, simple and well understood, **b)** scanning tools operate deterministically,

c) when worms scan, their ability to analyze and adapt is limited and deterministic, **d)** most scans perform a superficial investigation that is easily deceived, **e)** hackers scan to learn, and their naiveté makes them vulnerable to deception, **f)** the defender often has a good understanding of what hackers are seeking and what they expect to find. Examples follow:

a) *Low interaction honeypots* superficially impersonate servers, e.g., Specter and BOF. They are particularly useful for detecting and confusing scans, and for gathering intelligence from the scan. **b)** The LaBrea and Honeyd honeypots use ARP spoofing to impersonate numerous computers, and thereby hide real systems from scanning. LaBrea further degrades scanning by sending special packets that trick scanners into operating very slowly. **c)** NMAP and Queso are hacker tools that fingerprint the OS of network-attached computers. There are several systems for deceiving such tools. Kernel patches are available for changing Linux's network fingerprint. Also, the SNORT IDS has a plug-in (FlexResp) for fooling OS fingerprinting.

3) **Enumeration:** This step involves scanning for particular computer-security vulnerabilities. Deception is very useful against such vulnerability-scans, for the reasons just given for scanning. Also, many computer-security vulnerabilities and exploits are well-known, and the euphoria of discovering exploitable vulnerabilities can make the hacker less critical and thus vulnerable to deception. Examples follow:

a) A common ruse used by honeypots is impersonating vulnerable servers, for the purpose of detection and gathering intelligence. **b)** Similarly, honeypots can impersonate numerous instances of a real server. If the real server is vulnerable to attack, the imposters will substantially delay the hacker in locating it. In the mean time, attacks against the imposter-servers will be detected, the vulnerability will be revealed, and the delay will provide time to secure the real server. **c)** A long-used ruse is to deceptively alter a server's identifying banner. Hackers, including their tools and worms, examine banners in their search for particular vulnerable servers. **d)** Forescout is an intrusion detection product that replies to vulnerability scans with fake vulnerabilities, for example, vulnerable file shares. Hackers are positively identified when they later try to exploit these fake vulnerabilities.

4) **Gaining access:** There are classes of honeypots, *medium and high interaction honeypots*, that are designed to be broken into, for the purpose of detecting and preventing attacks and for collecting intelligence.

5) **Escalating privilege:** At AT&T, Bill Cheswick used a honeypot during incident response to learn about a hacker [Che92]. The hacker was given access to a user account, and his attempts to gain root access were recorded.

6) **Pilfering:** This is another step that offers abundant opportunities for deception, as hackers' pilfering efforts are well understood and somewhat predictable, and the hacker's euphoria of discovery can make them vulnerable to deception. For example: **a)** one of the best ways for a hacker to access additional systems is to obtain and *crack* a system's password file. The Specter honeypot emulates several OSs, and for each, it has a set of fake passwords that are available for "capture" by hackers. The deception detects hackers, reveals their intentions, and wastes their time. **b)** Pilfering can be detected by placing honeytokens among data that

is likely to be pilfered. (Honeytokens were described earlier.) For instance, the honeytoken could be the name of a file, and NIDSs can monitor for transfers of that file. **c)** Network sniffing is a means of pilfering. The Sniffer Detector is a honeypot system that uses honeytokens to detect sniffers [Gru98]. The Sniffer Detector sends plain-text root logins across a network. A hacker who has sniffed this information will be detected when he attempts to access those accounts.

7) **Covering tracks:** Deception can be used to hide logs and other records of forensic evidence. **a)** Log files can be hidden by creating duplicate log files and storing the duplicates in unconventional places. **b)** *Activity and event monitors* (a.k.a. *spyware*) are a class of security tools that record computer usage, for the purpose of detecting and understanding misuse [Ban02]. These tools run on the computer that is being monitored, and thus are vulnerable to being discovered by computer users. Activity and event monitors use deception to hide their processes and data from computer users [Gla02]. The spyware can be easily found using anti-spyware (described earlier) or Internet-posted instructions. However, the hiding is effective against users who aren't explicitly looking for spyware or who don't know how to find it.

8) **Creating backdoors:** **a)** Among the most common backdoors are the remote-access servers that were described earlier (section 2.2.1, page 5). Well-known servers include BackOrifice, NetBus and Sub7, and there are numerous fake versions of them. These fake servers offer varying degrees of impersonation, and most of them log access. **b)** Other deceptive systems can be easily converted to impersonate backdoors, including PortSentry and the Deception Tool Kit. **c)** The Fake Backdoor System is a Perl script for impersonating a simple bash-shell backdoor. Detection is a key component for all of these fake backdoors.

5.2.2 Model of a single attack

The prior hacking process-model is for incidents, and they consist of one or more attacks. A model for how a single attack works is shown in Figure 4. As illustrated in the figure's top row, an *Attacker* uses *Tools* to gain *Access*, and this produces *Results* that are intended to achieve the attacker's *Objectives*. The figure's center column shows that *Access* is gained by exploiting one of three types of *vulnerabilities*. The *unauthorized access* (*or use*) is to *processes*, or to *files* (or *data-in-transit*) *through processes*.

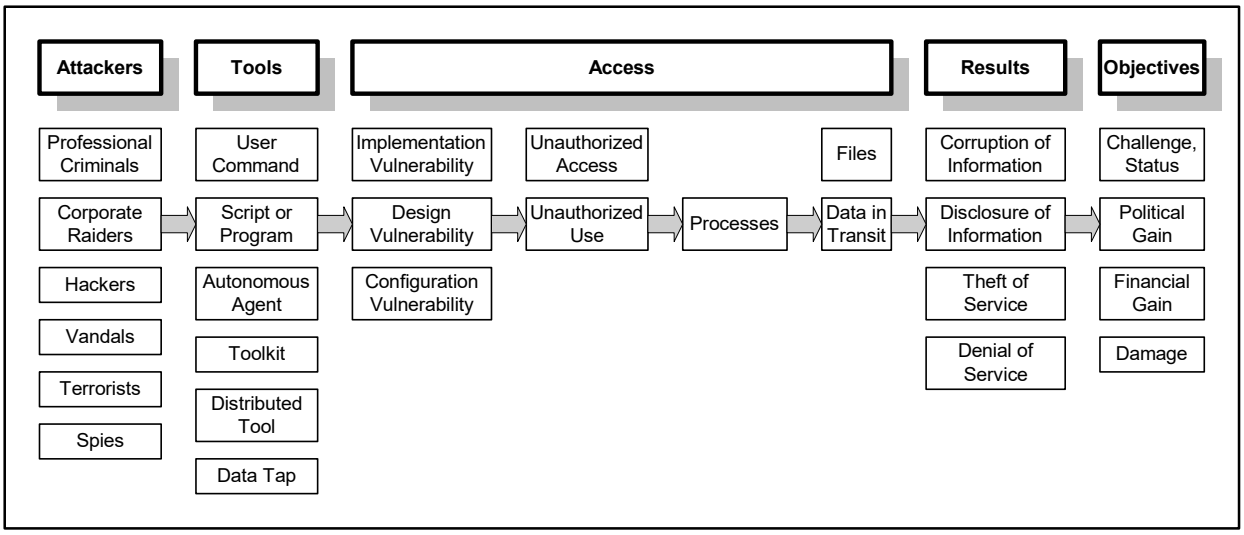


Figure 4 : Model of a single attack (adapted from Howard [How98])

The hacker is potentially vulnerable to deception at each step of the attack. A USMC deception-manual offers a use of deception that is applicable here: "we must force the enemy to make a mistake; a specific mistake that aids operations planned by us" [USM89]. For example, *autonomous agents* (e.g., worms) like Slapper-B scan for *implementation vulnerabilities* in servers. The scan examines each server's banner for its make and version. Such scans can be defeated by deceptively misstating a server's make and version.

There are many models of how computer-security works, and each potentially reveals uses for deception. General models have been presented, but more detailed models can be used. For instance, the famous paper "Smashing the Stack for Fun and Profit" is a model for buffer-overflow attacks. and it can be studied for uses of deception. The next section presents models adapted from military deception.

5.3 Models of hacker decision-making

Models for Illustrating the Uses of Deception		
Computer-Security Defense (pg. 41)	***The Hacking Process*** (pg. 44)	***Hacker Decision-Making*** (pg. 48)

Deception can be used to attack the hacker's decision-making process, in two related ways: the hacker's **will to attack** can be lessened so that, ideally, he decides not to attack, or at least, he decides to invest less resources in the attack. Secondly, the hacker's decision-making can be **degraded**, so that he is slowed down, made less effective, or countered by computer-security measures. For example, situations that are favorable to the hacker can be made to appear unfavorable. Deceptive OS fingerprints are one way to achieve this effect, as they can

make the hacker's preferred targets (e.g., Linux) appear to be something else (e.g., Windows). This section explains how deception can be used to thwart hacker decision-making. Two models provide the framework for the explanation: the OODA model and the OPSEC model.

5.3.1 The OODA model

The **OODA loop** is a model from military-theory [DD97] that is also useful for computer-security. It views conflict as time-competitive cycles of observation, orientation, decision, and action (OODA), as shown in Figure 5. First, each party to a conflict *observes* himself, his surroundings, and his enemy. Second, based upon those observations, the combatant produces a mental image of the situation and gains situational awareness. This *orientation*, or understanding, becomes the foundation upon which to erect a plan. Generally, the better the orientation, the better the plan. Next, based upon this orientation, the combatant *decides* upon a course of action and develops a plan for it. Last, the combatant *acts*, or puts the decision into effect. This is the execution phase where the decision and plan are implemented. Since this action has changed the situation, the combatant again observes, beginning the cycle anew. This cycle is known as the OODA loop. All combatants' OODA loops occur simultaneously, and they interact with each other.

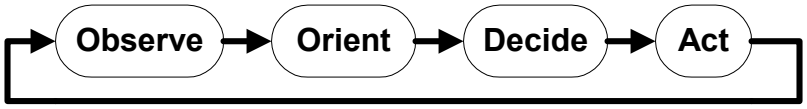


Figure 5 : The OODA loop.

A hacker has his own OODA loop for a hacking incident, and he has additional OODA loops for various parts of the incident, such as his intelligence activities and single attacks. A principle of tactics is cycling through one's own OODA loop faster than the enemy can cycle through his. This is referred to as *operating within the enemy's OODA loop*. It can be accomplished by speeding up one's own OODA loop or by slowing down the enemy's.

The OODA model reveals uses for deception. The first two stages of the OODA loop, *observation* and *orientation*, make up the combatant's perception. Deceptions can adversely affect an enemy's perception in two ways [CIA80,USA88]: **1) "ambiguity-increasing" deceptions** (*A-type deceptions*) increase doubt and uncertainty in the target's mind. They lower his likelihood of forming a correct perception by taking away needed information or by adding superfluous information. **2) "Misleading" deceptions** (*M-type deceptions*) reduce uncertainty in the target's mind by having him become convinced of a particular falsehood. In practice, the effects of a deception will lie somewhere on a continuum between A and M-type deceptions.

Many honeypots are A-type deceptions. For example: **1)** A network has two VPN devices that provide tunnels though its perimeter. A honeypot impersonates a third VPN device. Hackers suspect one of the VPNs is a honeypot, but they are uncertain of its identity among the three. **2)** Honeyd and LaBrea create numerous imposter computers on a LAN broadcast-domain. They make the hacker uncertain of the real computer's identity.

As illustrated by the honeypot examples, A-type deceptions affect perception by: **1)** overloading the target's intelligence capabilities, **2)** decreasing his intelligence accuracy, **3)** creating confusion, and **4)** increasing his perceived risk. A-type deceptions slow the target's OODA loop by impairing his perception and by causing indecision and timidity. A-type deceptions also degrade the effectiveness of the target's OODA loop. Impairing his perception causes him to misallocate resources in time, place, quantity, or effectiveness. A hacker's resources include such things as his: time, skills, tools, computers, and compromised network assets. M-type deceptions can also slow the hacker's OODA loop and degrade its effectiveness.

In general, M-type deceptions are preferable to A-type deceptions [Dew89,USA88]. A- and M-type deceptions have different effects on the target's decision making. Ideally, deception aims to mislead the hacker into a specific and predictable course-of-action or inaction that can be exploited [Dew89]. M-type deceptions have the greatest potential for causing this to happen. By making the target certain of particular falsehoods, it limits, in his mind, the number of appropriate actions. Ideally, only a single course of action will appear appropriate to him. For example, if a hacker has been tricked into believing that a particular computer has the assets he seeks and a vulnerability he can exploit, then he will predictably attack that computer.

Uncertainty has the opposite affect on the target. It tends to increase, in his mind, the number of appropriate actions. Also, uncertainty tends to make the best choice unclear and the decision-making subjective. Relative to M-type deceptions, A-type deceptions tend to induce less predictable target-actions.

In spite of their limitations, A-type deceptions offer some unique benefits: **1)** when the target is made uncertain, it may not be possible to predict what he **will do**, but it may be possible to predict what he **cannot do**. For example, when honeyd is used to impersonate many computers, the hacker cannot accurately target real computers. **2)** When the target possesses several elements of the truth, it can greatly limit the use M-type deceptions. This can occur with computer-security defenses that are static, long-lasting, and have publicly-accessible interfaces. An example is public servers on the DMZ. In such cases, it may be possible to use A-type deceptions instead [USA88].

There is another way in which deception can impair the target's OODA loop. If a hacker knows that deception is an effective part a network's defenses, the very use of deception can make the hacker uncertain [USM89] and aid computer security: **1)** the uncertainty produces the effects of A-type deception, as previously listed [DH82b]. **2)** The hacker's deception-detection efforts will slow him down. Also, faulty or uncertain detection can cause the hacker to misinterpret normal system behavior as deceptions or to engage in unnecessary counter-measures [RR02]. **3)** The hacker is vulnerable to exaggerated claims of deception use [Han85]. For instance, a well-known organization could exaggerate its deception-use via deceptive postings on hacker bulletin-boards. When hackers encounter less deception than reported, they are likely to attribute the scarcity of evidence to the deceptions' stealthiness. **4)** The hacker may reject true windfall discoveries, believing them to be deceptions. For example, a hacker stumbles across a true computer-security blunder such as a vulnerable server, but he rejects it as deception because it is "too good to be true". During World War II, there were several cases in which true battle plans fell into enemy hands, but the enemy rejected the plans, thinking they were deceptions [CIA80].

5.3.2 OPSEC models

Operations-security (*OPSEC*) is a military concept that is also applicable to civilian organizations [OPS03]. *OPSEC* is a process that is used to deny adversaries information about one's own capabilities and intentions. Deception is a means for OPSEC. OPSEC deceptions can cause hackers to work ineffectively, and they can prevent hackers from taking advantage of favorable situations [Dew89].

The OPSEC process identifies, controls and protects information about systems and operations. There are three steps in the OPSEC process: **1)** the organization's *critical information* is identified. It is the information that, if available to an adversary, would harm the organization's ability to effectively carry out its operations [NNS03]. **2)** The critical-information's vulnerabilities, threats and risks are identified, and **3)** countermeasures are devised to protect the critical information. Countermeasures include defenses such as classifying information, physical security, communications security, computer security, and deception [Cam96].

OPSEC can be used to protect computer systems and operations by: **1)** identifying the critical information that hackers need to effectively carry-out attacks, and **2)** limiting the hacker's ability to collect or derive this information. An example of critical information is the network perimeter's external entry points, such as VPN interfaces. OPSEC deceptions can: **1)** thwart the hacker's intelligence collection and analysis, or **2)** provide false information that degrades the hacker's decision-making processes.

There are two noteworthy hiding techniques that are useful for OPSEC: **1)** when a system can't be hidden, deception can be used to hide some of its attributes, and **2)** a system can be hidden by implementing it in an unconventional way. The latter deception can be strengthened by also deploying a conventional-looking dummy system. For example, a real SSH server could be configured to listen on an unconventional port (i.e., any port other than 22), and on the same computer, a honeypot SSH server could be configured to listen on the conventional port 22.

OPSEC can be used to protect computer systems and operations, as shown in the table below.[33]

Deceptive Measures for Securing Critical-Information about Computer-Systems	
Hiding prevents the hacker from deducing:	*Showing* persuades the hacker to deduce:
What it is	It is something else
What it does	It does something else
How it works or operates	It works or operates differently
When it works or operates	It works or operates at a different time
Where it is	It is somewhere else
Where he (the hacker) is	He (the hacker) is somewhere else
The defender's knowledge of the hacker's	The defender's knowledge of the hacker is

[33] Table adapted from Daniel [Dew89] and Handel [Han85].

tools, techniques, capabilities and intentions	greater, or less, than it actually is
How successful the hacker's operations are	His operations are more, or less, successful than they actually are

Many of this chapter's deception examples provide OPSEC, including: fake wireless access-points, fake replies to traceroute, honeyd's impersonation of computers, and misleading banners for servers. The OPSEC process enables the planner to use such deceptions effectively. It systematically identifies the network's most critical information and the opportunities for using deception to secure it.

6 The means of deception

The *means of deception* are the ruses and techniques used to deceive. More plainly, they are the ways to trick people. The deception planner needs a toolbox of deception means. We have researched this topic, but the results are not written-up, due to finite resources. Some of the key points from our research are listed here:

- Computer-security deceptions:

Honeypots are the most prevalent use of deception for computer-security. Lance Spitzner recently published an excellent and thorough book on the subject [Spi03]. The Honeynet Project's web site is another excellent source of information (http://www.honeynet.org).

We conducted an extensive survey of the use of deception for computer security. Most the examples in this paper came from that survey. We also developed a taxonomy that identifies the major categories of deception devices.

- Hiding and showing:

Bell and Whaley developed a very insightful taxonomy for hiding and showing [BW82]. It is cited by several papers, and it is very helpful for understanding how hiding and showing work. However, we have identified several shortcomings in the taxonomy. We are developing a revised taxonomy.

- Standard ruses:

There is a set of timeless ruses that have general applicability, for example, diversions [USA88]. Many of them can be used in computer security.

- Psychological vulnerabilities to deception:

Jervis' hypotheses for misperception were described earlier (section 4.1.2.1, page 21). Being human, the target has the same tendencies to misperceive. These tendencies make the target vulnerable to deception, and the deception planner can exploit them to deceive the target [Heu81].

7 Honeyfiles

Honeyfiles are an intrusion detection mechanism based on deception. Specifically, a honeyfile is a bait file that is intended for hackers to open, and when the file is opened, an alarm is set off. For example, a file named *passwords.txt* could be used as a honeyfile on a workstation. Hackers who gain unauthorized access to the workstation will be lured by the file's name, and when they open the file an alarm will be triggered.

Honeyfiles are implemented as a file server enhancement, and the file server's users can make any of their files a honeyfile. Alarms are sent by e-mail directly to the user, and services can be used to securely forward the e-mail by phone or pager. With honeyfiles, detection mechanisms can be very effectively deployed, as they are placed by the end users who are intimately familiar with the network's file spaces. In addition, when an alarm is sent, those end users can easily and effectively interpret it.

Honeyfiles can be used to detect unauthorized access to computers with a file space mounted from a file server, a standard practice in all but the smallest of organizations. By implementing the alarm system on the file server, honeyfiles provide defense in depth for the file server's clients. Also, in protecting the clients, honeyfiles can detect unauthorized access gained through unknown attacks, as well as unauthorized access gained through unintended file-access permissions.

When effectively deployed, it will be difficult for hackers to avoid honeyfiles, and honeyfiles show potential for avoiding some of the problems frequently encountered by network intrusion-detection systems (NIDSs), such as high false-positive rates and also high false-negative rates for unknown attacks. Honeyfiles offer several additional benefits, such as the opportunity to increase a network's internal security without impairing its normal operations. In addition, the honeyfile system can be used to detect unauthorized access to real files (in addition to bait files), and this provides some substantial advantages over alternative systems like Tripwire.

A prototype honeyfile system has been implemented on the Network File Server (NFS), and it has been tested by subjecting it to hackers. Honeyfiles, and the prototype, are further described in the following sections:

7.1 How honeyfiles work

This section describes the use of honeyfiles, and the system that implements them. Honeyfiles' basic functionality is described, and then their uses and benefits. Next, some essential honeyfile enhancements are presented, and the section is concluded with an analysis of honeyfiles' limitations.

7.1.1 The basic function

Honeyfiles are implemented by a *honeyfile system*, and it provides the necessary file-system and alarm functions. The file-system functions are implemented as an enhancement to a network file server, as illustrated in Figure 6. The system's components are numbered in the figure and their descriptions follow.

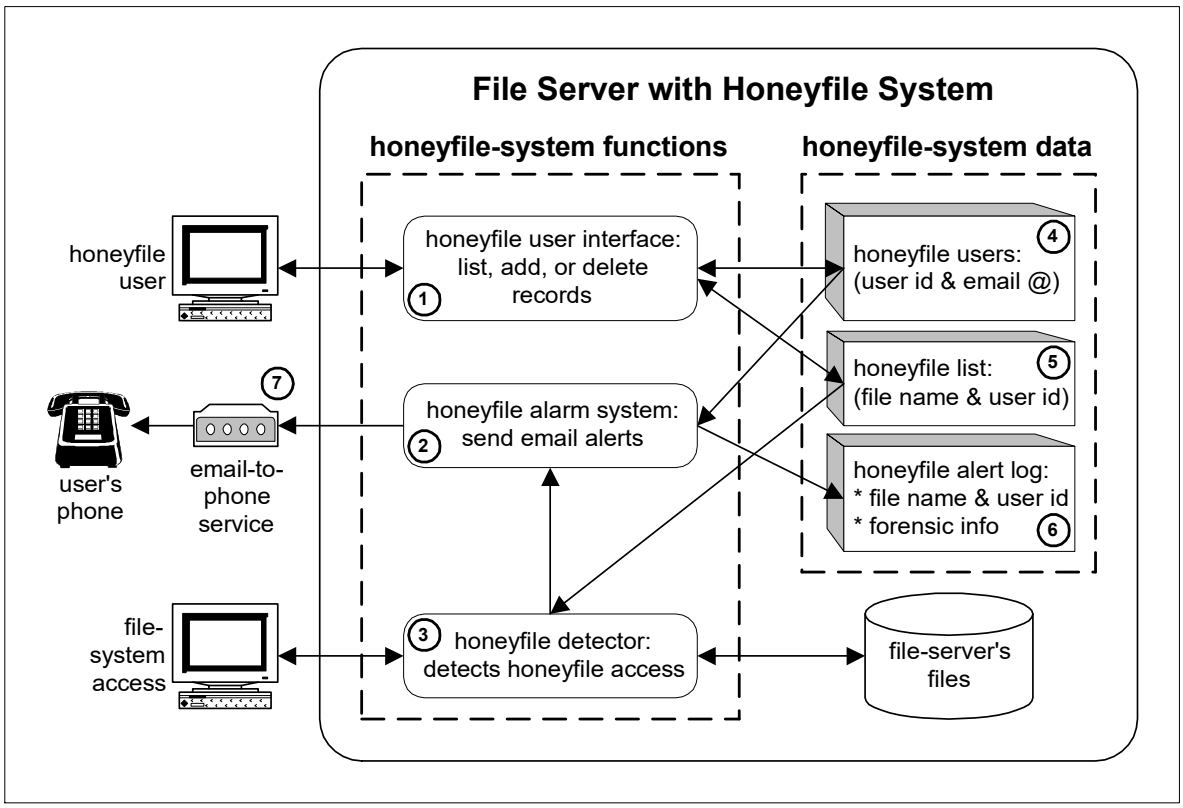


Figure 6 : The honeyfile system

With few exceptions, any file within the user's file space can be a honeyfile. The honeyfile system provides an interface whereby file-server users specify their honeyfiles (1).[34] Honeyfile alarms are sent as email messages, so the user also provides an email address to be used.[35] The email messages are called *email alerts*, or simply *alerts.* To detect access to honeyfiles, the honeyfile system monitors all file access on the file server (3). When a honeyfile is accessed, an alert is sent (2), and it is logged (6). The alert includes the name of the opened

[34] A file records the system's honeyfiles (5). Each record contains a file name and user ID.
[35] A file records the system's users (4). Each record contains a user ID and email address.

honeyfile, and forensic information for incident response, such as the IP address of the computer that opened the file.

The network can be configured for the email alerts to be sent in a secure manner. Ideally, they will be sent to an automated service that will call the user's cell phone and read the email message (7). This ensures secure delivery of the alert should the user's mail client also be compromised. Phone delivery also enables the user to be notified while away from his computer.

Honeyfiles can detect the hacker's investigation and copying of files, including:

- the hacker's personal perusal of the file space. Hackers can be tricked into opening files with alluring names that indicate the file is of value.
- the hacker's use of search tools to find particular types of files, e.g., file names containing the string "password". These tools can examine file names or contents. Honeyfiles can be created to match common hacker searches.
- the hacker's use of tools like *tar* and *zip*, to copy and steal the contents of entire directories. Such copying can be detected by placing honeyfiles in directories that are likely to be stolen, and the honeyfile's name will blend in with the other files, e.g., "sysrun1.dll".

There are four types of files that are generally of interest to hackers, and that can often be used as honeyfiles:

- files with information about accessing and using other systems, such as password files (*passwords.txt)*, user manuals (*customer-accounts-system.pdf*), and security documentation (*vpn- instructions.doc)*,
- system or application programs that the hacker may run, but that authorized users would not run, such as the gcc compiler,
- files that contain evidence of the attack, such as log files, and
- files that contain information of use other than hacking, such as credit card numbers, intellectual property, expected stock market prices, and military intelligence.

Honeyfiles should be named and located in such a way that its owner will not be inclined to open them accidentally. One technique is to give a honeyfile a name that appears unusual only to its owner. The unusual name can help jog the owner's memory and recognize the honeyfile. For example, a honeyfile password file could be named *complete-passwords.txt*. Its owner has no partial password files, so the prefix "complete" will help him recognize the honeyfile. However, to a hacker, the prefix is not suspicious, and in this case, it even makes the file more desirable. Additionally, the real password file can be hidden by encrypting it, and giving it an uninteresting name and location, e.g., *c:\driver-downloads\printer.sys*.

Honeyfiles can contain deceptive content, such as fake user-IDs and passwords. Deceptive file-content can take on a plethora of uses and forms, and it can be used independently of honeyfiles. In order to concentrate on central honeyfile functions, this paper does not address deceptive content in honeyfiles. Instead, it focuses on honeyfile deceptions involving just file system information, i.e., the file's location and its directory entry, including its name.

7.1.2 Uses

Honeyfiles' primary uses, as a computer security measure, are as follows:

- Honeyfiles can detect unauthorized access to computers:

The primary strength of honeyfiles is their ability to detect unauthorized access to computers whose file-space resides on a file server. For example, a workstation stores the user file-space on a file server, and the workstation automatically mounts the file-space at boot time. If a hacker breaks into the workstation, his presence will be detected if he opens a honeyfile within the user file space. It is common practice for computers to automatically mount a file space from a file server, so for many organizations, honeyfiles can be used to protect a large portion of their computers.

- Honeyfiles can detect unauthorized access to file servers:

In all but the smallest of organizations, most of the organization's files reside on file servers, and honeyfiles can be used to protect them. In general, honeyfiles detect unauthorized access to the file spaces on a file server. Further, the detected access can reveal the file server's compromised security mechanism, including: **1)** compromise of the file space's user ID and password, **2)** compromise of weak or defective authentication mechanisms on the file server, e.g., NFS' notoriously weak authentication, and **3)** exploitation of errors made in granting file-space permissions, e.g., accidentally making the file space "world readable".

- Honeyfiles can be used to protect a wide variety of files and computer systems:

A honeyfile can be almost any file stored on a file server. In addition to regular data files, they can be files used by application programs, such as folders within a mail client. For example, a company's executive email may reveal future stock prices, making the information extremely valuable to hackers. Security personnel can work with the executives to place honeyfiles within their mail clients. Honeyfiles can also be desktop shortcuts to programs that entice hackers who have gained remote access via programs like BO2K or Windows Terminal Server. For instance, a workstation at a bank could use a honeyfile shortcut *vault-camera-01*, and many hackers would irresistibly open the application.

Honeyfiles can be used by almost any file system user. In addition to protecting someone's file spaces and computers, honeyfiles can be used to protect application programs. For example, a web-server's cgi-bin directory can be populated with empty honeyfiles named after notoriously vulnerable scripts. Alerts will be generated when hackers attempt to run those scripts, and the web-server connections can be identified by correlating the alerts with entries in the web server's connection log.

It is even possible for some of a computer's system files to be honeyfiles. Although most system files must reside on the local computer, some can be links to files on a file server. For example, system programs that would only be used by hackers can be replaced with symbolic links to honeyfiles on a mounted file system. For instance, if a workstation will not use the gcc compiler, */usr/bin/gcc* can be replaced with a symbolic link to a honeyfile. The honeyfile does not have to be the gcc compiler, and it could even be a deceptive program that generates

confusing compiler error-messages. Although some hackers may notice the symbolic link and suspect a honeyfile, it can still catch some hackers.

Another way for a computer's system files to be honeyfiles is to mount them from the file server. For example, in Unix, the */etc* directory contains essential system configuration files. A Unix computer can start with a local copy of the /etc directory, but in the boot process, the directory can be mounted from a file server. Within /etc, honeyfiles can impersonate unused files that hackers are likely to access. For instance, */etc/hosts.equiv* specifies computers that can login without a password. Many installations don't use the file, but hackers may access it to learn trust relationships or when they start servers that read the file.

- Honeyfiles can enhance a network's internal security without impairing normal operations:

Networks typically use a relatively low level of internal security, as additional security is burdensome and costly. For example, extra access controls make resource sharing difficult, and making IDSs more sensitive increases their false alarm rates. Honeyfiles can provide a means of increasing internal security without impairing operations. Honeyfiles have little adverse affect on legitimate computer use, and the honeyfiles' false alarm rate is expected to be much lower than standard network intrusion-detection systems (NIDSs). If honeyfiles are effectively deployed in a network. it would be difficult for a hacker to not be detected. Also, honeyfiles can be an effective deterrent for insider hackers because they, like all other network users, will have been informed of the honeyfiles' availability and use.

- The honeyfile system can be used to detect unauthorized access to real files, and it offers substantial advantages over alternative systems like Tripwire.

Honeyfiles are deceptive bait files created to detect a hacker's presence. In addition to detecting access to deceptive honeyfiles, the honeyfile system can be used to detect access to real files. For example, when a workstation user leaves work for the day, he could use the honeyfile system to set alarms for all of his files. Also, real files that are of high value, and infrequently opened, can be alarmed simply to monitor their use, e.g., a real password file. During incident response, if the hacker's intentions are known, the files he seeks can be alarmed. Similarly, if a former employee is suspected of having created "backdoors", his entire file space can be alarmed to detect his unauthorized access.

The honeyfile system can be easily extended to provide a powerful alternative to Tripwire. Tripwire is used to detect changes in files. It creates and stores checksums for files and then periodically recalculates the files' checksums to detect changed files. The honeyfile system can provide this function for a file server. For selected files, the system would send an alert when the file is changed.

For detecting changed files on a file server, this honeyfile system enhancement provides two substantial improvements over Tripwire: **1)** the honeyfile system detects changes when they occur, whereas Tripwire detects changes during its periodic, and often infrequent, execution, and **2)** the honeyfile system is simpler than Tripwire. The honeyfile system resides on the file server and an end user only has to specify the honeyfiles. With Tripwire, the user must either run Tripwire himself, or grant file access to a separate machine that runs Tripwire. If the user runs Tripwire, he must securely store the binaries and checksums.

- Honeyfiles can provide the security functions of detection and deterrence, and they can support incident response.

In addition to detecting attacks, honeyfiles can deter attacks. If hackers know honeyfiles are being used, it can dissuade them from hacking, and it can slow them down by making them cautious and uncertain. Honeyfiles are also useful for incident response. Investigators can view all of the alerts for a network, and collectively, they may reveal a hacker's capabilities, intentions, or courses of action.

7.1.3 Benefits

Honeyfiles' benefits stem mostly from: **1)** decentralized deployment: the network's end users create and place alarms, and **2)** centralized implementation: the alarm mechanism resides on the file server rather than on its clients.

- Honeyfiles can be used to detect unauthorized access gained through unknown attacks, as well as unauthorized access gained through unintended file-access permissions:

Honeyfiles detect the hacker after he gains unauthorized or unintended access. The detection mechanism is independent of the specific techniques used to gain access. Honeyfiles can even reveal file and computer access that is contrary to a user's intentions, as occurs, for example, through the accidental granting of file sharing permissions. Generally, non-deceptive detection mechanisms are very weak at detecting unknown attacks, as well as discrepancies between a user's access rules and his intentions.

- By enabling end-users to create alarms, the detection mechanisms can be effectively deployed and the alerts effectively interpreted:

If honeyfiles are created and placed well, it can be difficult for hackers to avoid them, resulting in a low false negative rate. End users are intimately familiar with the data they create and manage. Honeyfiles make it possible for users to create and place alarms where they are most needed and where they will be most effective. With some basic instruction on security and honeyfile tactics, users can effectively deploy honeyfiles. Also, end users can evaluate and improve their alarms' effectiveness because they receive alerts directly. Further, end users can adapt their honeyfile use as the network and its threats change.

Honeyfile users can accurately discern between true and false positives because they create the honeyfiles and receive the alerts. For instance, if a user accidentally opens a honeyfile, the resultant alert can be recognized as a false positive. If an alert is sent when the user is not accessing his file space, the alert can be recognized as a true positive.

- Honeyfiles support defense-in-depth for the file server's clients:

Honeyfiles provide the file server's clients with an alarm system that resides outside of the client itself, and this adds a layer of depth to the client's defenses. When a hacker breaks into a client, the honeyfile's alarm mechanism is on the file server, not on the client. If the honeyfile's alarm mechanism was on the client, the alarm would be vulnerable to attack or

detection by the hacker, especially when the hacker has "rooted" the client computer. Alerts are sent by e-mail, and they can be made to travel over a secure channel. Further, the e-mails can be sent to a service that forwards the e-mail message by phone or pager.

- Honeyfiles are an effective deception because they can be easily created, require little falsehood, and involve little risk. Also, hackers can be highly vulnerable to honeyfile deceptions.

A honeyfile is integrated with a real file space, and this real context makes the honeyfile deception easy to create and difficult to detect. Also, honeyfiles themselves involve little falsehood—just a directory entry. Further, honeyfiles involve little risk. If detected, they may still be useful against other hackers, but if necessary, they can easily be replaced. Honeyfiles can fail to detect attacks, but the failure does not increase the network's vulnerability.

Honeyfiles take advantage of several deception vulnerabilities in most hackers' intelligence collection and analysis: **1)** when hackers initially access a file space, they must search it in order to locate valuable data. If the hacker's search can be anticipated, honeyfiles can be placed where he is likely to encounter them. **2)** The hacker's limited knowledge of the file space makes it difficult for him to discern what truly belongs there, and his naiveté makes it easy to create deceptive honeyfiles. **3)** It can be very difficult for the hacker to detect a honeyfile. The honeyfile deception is created using a small amount of information, i.e., the file's directory entry, and usually, there is no way for the hacker to cross-verify the information, and **4)** In most instances, if the target wants to know a honeyfile's contents, his only option is to open the file and trigger the alarm.

- Implementing the alarm system on the file server makes honeyfiles available to almost all network computers:

Honeyfiles can be created by any computer that uses the file server. Honeyfiles can be used by computers with a wide variety of operating systems and file systems, without having to enhance the computers' many distributions, e.g., Windows and Unix. Also, having a single alarm system makes it easier to train users.

- Implementing the alarm system on the file server centralizes security management functions.

Having a single alarm system makes the system's maintenance and defense easier, as the system resides in one place, rather than on each of the client computers. Further, having a single alarm system makes it easier for network security personnel to monitor the alarm system's overall use and effectiveness, e.g., the honeyfile system can list its users and honeyfiles, and the log file records all alerts.

- Honeyfiles show potential for avoiding some of the problems encountered by network intrusion-detection systems (NIDSs).

NIDSs are typically very weak at detecting unknown attacks, whereas honeyfiles can detect unknown attacks and even access gained through unintentional file-access permissions. Also, NIDSs can generate an exorbitant volume of false alarms, and investigating them is costly

and unproductive. In contrast, honeyfiles show the potential for having a much lower false alarm rate. Further, with NIDSs, false alarms are often investigated by a centralized security group that does not work directly with the protected data, making investigation difficult. In contrast, honeyfile users can accurately and easily dismiss many false alarms because of their familiarity with the protected data. In addition, false alarms provide feedback that enables users to reduce them, e.g., a honeyfile can be renamed so the user is less likely to accidentally open it.

Honeyfiles make it possible for alarms to be deployed by the personnel who create and manage information assets. They are very familiar with the assets, and they also have a vested interest in the asset's security. In contrast, when NIDSs are deployed by a centralized security group, it can be difficult for them to accurately understand the network's changing information assets.

7.1.4 Enhanced functions

An earlier section described basic honeyfile functions, and this section describes some enhancements that greatly improve honeyfile use. These improvements have to do with maintaining realism, controlling alarms, and enabling incident response.

Operational systems change over time, and so too must most honeyfiles if they are to be believable. A file's MAC times record when it was created, last modified, and last accessed. Honeyfiles that portray in-use files must have their MAC times periodically updated. The honeyfile system can solve this problem by periodically updating MAC times, within user-defined parameters. For example, a honeyfile named passwords.txt might have its access time updated three times a week, and the modified time updated once a week.

If deceptive content is being used, it may also need to change over time. Although deceptive content is not addressed here, there is a noteworthy technique for automatically updating a file's deceptive content. The honeyfile's contents mirror a source file that his hidden from the target, and the honeyfile system periodically updates the honeyfile from the source. For example, an executive's mail client contains a honeyfile mail-folder named *leveraged-buyout*. The executive stores email about the leveraged buyout in two other mail folders, and those folders are hidden by giving them deceptive and uninteresting names. One of these folders contains highly sensitive correspondence. The other folder contains the remaining correspondence, and it is the honeyfile's source. Insider information about leveraged-buyouts can be extremely lucrative, and such a honeyfile can be an effective way of detecting hackers who are seeking that information.

The need for deceptive content can be avoided by simply not allowing a honeyfile to be opened or modified. The honeyfile system would allow its users to mark a honeyfile as *not accessible*, and then deny attempts to open or modify the file. The returned error code can deceptively state that access was denied due to file permissions, and hackers who do not check the file's permissions are likely to believe the error message. Regardless of whether the hacker believes the error code, an alert will be generated and the access detected.

Honeyfile use can also be improved by providing controls for selectively generating alerts. Some processes and users must be permitted to open honeyfiles without setting off alarms, such as tape-backup processes and the root user. Also, honeyfile end-users need the ability to temporarily suspend their honeyfile's alerts, as was described earlier.

Another type of honeyfile enhancement is functions for aiding incident response. For some honeyfiles it can be useful to do more than send an alert when they are opened. The file server could automatically take preventive measures such as blocking access from suspect computers, increasing its logging, or even shutting down.

7.1.5 Limitations

Honeyfiles' primary limitations are as follows:

- Some files and file-spaces are used in ways that are not well suited for honeyfiles:

Honeyfiles are appropriate for file spaces that are accessible to one person or a small group. If many people legitimately access the file space, it can be difficult to inform them of honeyfiles, false positives can be excessive, and alerts can be difficult to investigate. Also, files that are often opened do not make good honeyfiles, due to the volume of alerts generated.

There are search tools for finding files that contain a target string. The tools scan directories, and open its files. Using the tool, it is easy for a legitimate user to generate false alarms by inadvertently opening a file space's honeyfiles. Users can craft searches to avoid honeyfiles, and also, the honeyfile system could accommodate searches by enabling users to temporarily suspend their honeyfiles' alerts. However, in some cases, honeyfiles will not be viable due to unavoidable and frequent false alarms from file searches.

- Honeyfiles have tactical weaknesses that limit their use:

Like most deceptions, honeyfiles provide uncertain effectiveness against an individual attack. Many other security measures, such as strong encryption, are much more certain. However, as mentioned earlier, honeyfiles also have redeeming uncommon abilities, such as detecting unknown attacks.

Effective use of honeyfiles requires some knowledge of the hacker's behavior and interests. If little is known about potential hackers, honeyfiles may not be viable. Also, honeyfile use will be limited if the target does not tend to explore the file system, or if he does not use files on the file server.

There are some circumstances in which honeyfiles can be defeated. There are ways in which a hacker can identify real files, and if he opens only them, he will avoid honeyfiles. For example, a hacker can use a keystroke logger to learn what files are being used, and then open only them. Another honeyfile vulnerability is overloading of the alert mechanism. A hacker can use a file search tool to open all the files in a file space. This will trigger all of the file space's honeyfiles, and thereby overload response capabilities.

- Honeyfiles require that users have adeptness with computers and knowledge of security:

Effective honeyfile deployment requires user participation, and it requires users to have some security savvy as well as adeptness with computers. Not all users will have the time or skills needed to use honeyfiles. However, security personnel can provide some simple training that will be sufficient for many users, including honeyfile techniques for countering the network's particular threats and vulnerabilities. In addition, honeyfile users need to know

enough about computers to discern between true and false alarms, and to avoid setting off alarms, e.g., through the use of search tools.

7.2 Using honeyfiles

A honeyfile is a part of a deception operation, usually a small scale deception operation. Earlier chapters presented principles and techniques for developing deception operations, and they are largely applicable to honeyfile use. From these principles and techniques, those most pertinent to honeyfile use are presented here. Deception-operation planning is discussed first, followed by building the deception.

7.2.1 Planning

This section describes the essential elements of honeyfile planning:

- **Deception objective:**

The *deception objective* defines the deception operation's desired outcome, and it includes both the *target action* and the *deception exploit* (section 3.2, page 11). The purpose of the deception is to induce a specific target action. For honeyfiles, the target action is opening an alarmed file. The ultimate goal of the deception operation is exploiting the target action. For honeyfiles, the deception exploit is both the email alert and the incident response process that the alert initiates.

Incident response capabilities are needed for handling the unauthorized access detected by honeyfiles. Honeyfile users must know how to respond to detected attacks, and basic instructions can be provided in the email alert. Users perform the initial response (e.g., evaluate the alert and turn off the compromised computer), and security personnel will further handle the incident. For instance, the file-server's logging can be turned on, and compromised computers can be denied access to the file server.

- **Target identification and analysis:**

Effective deception operations require an understanding of how the hacker works and thinks (section 3.3, page 11). Security personnel can gain such understanding for honeyfile use, and they can convey the information to honeyfile users. The honeyfile *target* should be identified as specifically as possible for the network. In general, the target is hackers whose unauthorized file-server access can be anticipated.

In using honeyfiles, it is necessary to know how the target accesses the file space. Hackers' general access techniques were described earlier (section 7.1.1 on page 55), and honeyfile use can be improved though more specific information about the network's likely hackers.

Using honeyfiles against insiders can be more difficult than against outsiders. Insiders start with more knowledge about how systems work and about network assets. Insider's familiarity reduces their need for collecting intelligence, and it increases their ability to detect deception. However, their familiarity makes them more predictable, as they can go directly to

primary assets, and this predictability can be used in creating honeyfiles. For instance, at a company, all the employee's email clients are configured to use the same directory and start-up icons. However, the executives' email clients are configured differently, and honeyfiles impersonate the normal configuration.

In addition, a honeyfile deception relies primarily upon file names, so the target's language skills must be considered. For instance, foreign hackers may readily recognize dictionary words, but not abbreviations, so *passwords.txt* would be recognized, but not its abbreviated name, *pwd.txt*.

7.2.2 Building the deception

The deception is built by devising the deception story, feedback channels, and termination plan. Their use with honeyfiles is described.

7.2.2.1 Creating deception stories

Honeyfiles create a deception story for the target (section 4.1, page 17), and some principles and techniques for creating the story are presented.

- **Designing the story:**

The goal of the deception story is to induce a *desired perception* in the target, and the goal of the desired perception is to induce the *target action* that will be exploited. For a honeyfile, the desired perception is that the target consider the file to be of enough value to open it. The deception story can portray the perception to the target himself, or to the tools he uses for searching and copying files.

The deception story is constructed primarily by the file name, and the context in which the file resides. The context includes directories and files within the files space, as well as the computers that access the file space, e.g., workstations or servers.

Honeyfiles can be created by any network user. Security personnel can benefit from an in-depth understanding of deception stories, but most end users only need the basic deception story principles from sections 4.1.1.1 (page 19) and 4.1.1.2 (page 19).

Prior sections described several ways to create honeyfile deception stories. One of the primary techniques can be described as *showing the false and hiding the real***.** In particular, the honeyfile impersonates a real file, and the real file is hidden, e.g., a real password file is hidden, and the honeyfile *passwords.txt* is created. Two addition techniques for creating honeyfiles are:

- *The honeyfile pool:* multiple instances of a real file are created, and all but one of them are honeyfiles. For example, multiple instances of a password file are created, and named passwords-01.txt, passwords-02.txt, and so on. The real file *hides in plain sight*. The files' timestamps should not reveal the real file. One solution is to give all files the same timestamp. The honeyfile system could provide support for doing this automatically, e.g., by using hard file links, as described in section 7.4 on page 67.
- *Exploiting file-access techniques used by viruses and worms*: honeyfiles can be created for files that viruses and worms typically access. The security tool RegRun reportedly

does this. RegRun creates a bait file for viruses called *winbait.exe*, and it periodically checks the file for changes.

- **Presenting the story to the target:**

The story must affect the target as intended (section 4.1.2.1, page 21), and this includes the target receiving the story and interpreting it as intended. For a honeyfile to have its intended affect, it must be named and placed in accordance with how the hacker works and thinks. For example, when a hacker explores a file space, he's likely to start at its root, so honeyfiles are more likely to be discovered if placed near the root.

Also, the honeyfile must fit within the broader context in which it resides (section 4.1.2.2, page 23). Honeyfiles that portray windfalls for the hacker will only be believable in certain contexts, e.g., the network security director would not store his password file unencrypted, but an ordinary user might. It is possible to create a context for the honeyfile by creating whole directories and additional files, although such deceptions should be regulated by the principle of minimizing falsehood.

- **Implementing the story:**

It is essential that the deception story be seen by the target, and the primary way to do this is to communicate the story through the target's intelligence sources, i.e., the techniques he uses for learning about the network (section 4.1.3.2, page 27). Honeyfiles can exploit the hacker's typical means for finding and using files, as described earlier.

The degree of realism needed for the deception story depends upon the target's intelligence techniques and resources (section 4.1.3.3, page 29). Generally, honeyfiles that the target personally examines require more realism than those that he accesses in bulk through tools. For instance, if the hacker uses a tool to search an entire file space for all ASCII files containing the string "password", the honeyfile could have any name and reside in any directory.

- **Operational requirements:**

The deception story's operational requirements have to do with using the story in a changing environment (section 4.1.4, page 30). Honeyfiles that portray current operations will need to be regularly updated. For instance, when a honeyfile portrays an actively used file, its timestamp needs to be updated, and this can be done automatically, as discussed earlier. Over time, a file space's honeyfiles may need to be replaced, due to changes in the file space or its security risks. For example, new projects may be started, resulting in new directories and files, and new security threats and vulnerabilities may arise. Another cause for change in honeyfile use is practical experience, which instructs both security personnel and end users. Further, if a hacker opens a honeyfile, the end user will need to consider replacing the honeyfile. If the hacker has detected the honeyfile, it will no longer detect him.

Honeyfile users should also anticipate and prepare for potential problems that may arise, especially hackers' discovery of the honeyfiles. For instance, what might an insider hacker do when he opens a honeyfile and thereby discovers it? Another potential problem is compromise of the honeyfile system itself, through compromise of the file server. Hackers could then know all honeyfiles, and users would be compelled to create new honeyfiles. However, a benefit of

honeyfiles is that their failure usually does not create significant problems.

7.2.2.2 Feedback

Target feedback is an essential and critical component of almost all deception operations (section 4.2, page 33). Target feedback is both a strength and weakness of honeyfiles. Alerts provide very reliable feedback on opened honeyfiles. However, honeyfiles provide no feedback when they are not opened, making it difficult to assess the incidence of false negative detection. The primary way to assess false negatives is to examine the honeyfile alerts received during known attacks.

7.2.2.3 Termination plan

Honeyfiles will be removed when they are detected or when they are no longer useful. There are several ways that a honeyfile can be detected. For instance, if the target knows honeyfiles are being used, then the target may detect them when he examines their contents. However, the target will be detected when he opens the file, and it often will not matter whether he discovers the deception. If necessary, the honeyfile can have fake content to prevent the target from discovering the honeyfile deception. For example, passwords.txt could contain passwords.

If the honeyfile is detected, it will no longer work against the target and the other hackers he informs. If the honeyfile is to continue detecting them, it must be replaced. In general, removing or replacing honeyfiles will not pose significant problems. A benefit of honeyfiles is that the deception can usually be changed without suffering adverse interactions with the target, as the targets' access to the file space is infrequent and short lived.

7.3 Managing honeyfile use

The honeyfile system's centralized implementation enables security personnel to easily and effectively manage the organization's use of honeyfiles. General principles for managing deception operations were presented earlier in sections 3.4 (page 15) and 3.5 (page 15). From these principles, those most pertinent to honeyfile use are presented here. The first section describes how to maintain effective honeyfile use, and the second section describes risk analysis.

7.3.1 Maintaining effective use

Security personnel can easily evaluate the overall use of honeyfiles and thereby maintain their effectiveness as a network security measure. Honeyfiles use a centralized alarm system that resides on the file server, so the existing honeyfiles and prior alerts are easily analyzed. The current honeyfiles can be listed, to see how they are currently deployed. If honeyfile use is required, compliance can be examined and enforced. If existing honeyfiles are not effectively deployed, security personnel can provide additional training and assistance to end users.

Honeyfiles' primary purpose is detection. Honeyfiles' overall detection effectiveness can be determined by examining:

- alerts sent, as recorded in the honeyfile system's log
- alerts reported by end users to security personnel, as possible security violations

- the findings from the investigation of reported alerts
- prior hacker activity that is known by means other than honeyfiles, e.g., incident response investigation.

This information can be used to evaluate failures in honeyfile alerts. False positives can be determined for both alerts sent to end users and those reported by end users to security personnel. Insights into false negatives can be gained by comparing honeyfile use with the known hacker activity that was not detected by honeyfiles. All alarms, both valid and invalid, can be examined to learn how to better use honeyfiles, and to evaluate their overall contribution to network security.

The honeyfile system makes it easy for security personnel to test users' responses to alerts. By simply opening honeyfiles, user responses can be evaluated. This test can be used for training and as part of the evaluation of honeyfiles' overall effectiveness. Also, if honeyfile use is required, the test can be used to evaluate and enforce compliance.

7.3.2 Risk analysis

Honeyfile failures were discussed earlier, from the end user's perspective. Failure also needs to be considered from the perspective of system-wide honeyfile use. The honeyfile system itself can potentially fail. For instance, the system could have bugs that allow a hacker to gain control of the file server, or the honeyfile system could have a vulnerability that enables hackers to detect honeyfiles.

Honeyfiles can inspire unintended target actions that have system-wide significance. For instance, hackers may try to defeat honeyfiles by breaking into the file server. If honeyfile use leads to increased attacks upon the file server, its security will need to be enhanced. Another potential cause of unintended target actions is deceptions that lead hackers to do more harm than they would otherwise. For example, if a honeyfile has deceptive content that exaggerates the network's assets, it can inspire additional hacking. Honeyfile users can be provided with guidelines for not inspiring harmful hacker activity.

7.4 Bobcat: a honeyfile prototype

A proof of concept honeyfile system was built to test both the implementation and use of honeyfiles. This prototype is called *Bobcat* and its requirements, design and implementation are described below. The prototype implements the basic honeyfile system that was described earlier (section 7.1.1, page 55).

The prototype was tested by subjecting it to hacking, as also described below. A deceptive network called a *honeynet* was built and the Bobcat honeyfile system was installed on it. The hackers' activity was observed and the honeyfile system's effectiveness evaluated.

7.4.1 Implementation requirements

The honeyfile system's primary function is to maintain a list of honeyfile names, and to detect when files with those names are opened. However, to prevent circumvention of the honeyfile system, additional types of file access must be detected. If only file open is detected, then a hacker could rename a honeyfile, and open it under the new name without being detected.

Alternatively, the hacker could create a hard link to the honeyfile, and open the linked file without being detected. Further, to make the honeyfile system robust, it should detect any access to a honeyfile, e.g., deletion, and not just open.

To meet these requirements, Bobcat detects the following file accesses, and they are referred to as *the basic file accesses*.

- **file use**: file open is detected
- **file creation and deletion**: a honeyfile does not have to exist on the file system. An alert will be sent when the file is created. This feature is useful for detecting the creation of common hacker files and programs.
- **changes to the file's meta information**, including the name, owner and group ID, permissions, and timestamps.

File links pose challenges for the honeyfile system. When a link is created, a *linked file* is linked to a *source file*. In Unix, there are two types of links: *hard links* and *soft links* (or *symbolic links*). For a hard link, the linked file's directory entry points directly to the source file. For a soft link, the linked file's directory entry points to a file that contains the path name of a file, and that named file is a linked file or a source file. Thus soft links can form a chain of links, but hard links can not.

Bobcat provides the following honeyfile support for links:

- **hard links**: the basic file accesses are detected for each linked file or source file that is a honeyfile. Detection is supported for the individual honeyfile itself. If the honeyfile has a source file or linked file associated with it, the associated file must be made a honeyfile if access to it is to be detected. Once a file is made a honeyfile, hard links to it can not be made and attempts to do so will be denied. This restriction prevents hackers from using hard links to circumvent honeyfile alarms.
  Hard links can be used to provide both alarmed and unalarmed access to a particular file. For example, the source file could be alarmed, but not the linked file. The source file would be bait for the hacker and the linked file used by the file owner.
- **soft links**: the basic file accesses are detected for a source file that is a honeyfile. This includes access made via the source file itself or a linked file. Honeyfiles are not supported for linked files that are soft links. If a linked file is made a honeyfile, the resultant alarms are not specified, although other permitted honeyfiles will not be adversely affected. The prototype does not support such linked files because they are not of central importance, and the solution appears complex.

It is possible for a honeyfile user to receive too many alerts. For instance, a hacker could intentionally overwhelm the user with a large volume of alerts. The honeyfile system should keep the number of alerts sent to a useable level. Bobcat does this by maintaining a minimum time interval between alerts, for each honeyfile user. All alerts are logged on the file server, so alerts that are not sent can still be obtained.

7.4.2 Design

Bobcat provides honeyfile support for the NFS file server. Bobcat is an enhancement to NFS itself. All NFS file accesses go through the NFS *permission* function, which verifies that the access is permitted. Honeyfile access is detected within the *permission* function, before the file access is verified. Consequently, alerts are sent for honeyfile access, regardless of whether the access is permitted by NFS.

Bobcat was originally designed to monitor the file server's local file-system for honeyfile access. It was assumed that network file systems, such as NFS and Samba, use the local file-system to access files. Honeyfile access was detected by monitoring calls to file open on the local file system. However, this didn't work for NFS, as it caches file handles for its file opens. Consequently, some NFS file opens do not use file open on the local file system. Bobcat now monitors the NFS permission function, as all NFS file accesses pass through there.

Bobcat's monitoring of file open was left in the prototype, as it could be used to support Samba, which appears to consistently use the local file system. Supporting Samba will require additional enhancements to the local file system in order to detect honeyfile accesses other than open, e.g., the renaming of honeyfiles.

7.4.3 Implementation

This section provides an overview of the Bobcat implementation, illustrated in Figure 7. Bobcat's components are numbered in the figure, and their descriptions follow. The components are divided among kernel functions, user-space functions and files. Bobcat is implemented on the Red Hat 9, 2.4.20-9 kernel. It uses the NFS shipped with the kernel and the Ext2 file system.

Bobcat uses a simple configuration mechanism that consists of two files, both in XML format. There is a list of honeyfiles (8), and each entry in the list contains a file name and notification ID. The notification list (9) stores the e-mail addresses used for alerts, and each entry in the list contains a notification ID and e-mail address. In the prototype, the files are manually edited. A production honeyfile system will need a GUI interface for end users.

For a running system, the list of honeyfiles is kept in kernel memory (2). It is loaded from the honeyfile list stored on disk (8). The honeyfile-list build daemon (5) is a user-space function that loads the honeyfile list at boot time, and the list can later be reloaded by sending the daemon a *sig –HUP* signal. The daemon builds the in-kernel list by calling the honeyfile-list functions that reside in the kernel (1). The functions can also be called from the operator console by using command-line utilities (4).

Bobcat's monitoring of honeyfile access occurs in the NFS *permissions* function (3). For each file accessed, the in-kernel honeyfile list (2) is checked to determine if a honeyfile is being accessed. If it is, then an alert is sent to the notification daemon (7). The daemon runs in user-space, and the alert is sent in a UDP packet via inter-process communication.

For a running system, the notification list is kept in user-space memory (6). At boot time, the notification daemon (7) loads the list from disk (9). When the notification daemon receives an alert from the permissions function, it sends the alert via e-mail and it logs the alert on disk (10). The notification daemon regulates the frequency of e-mail alerts for each notification ID, as described earlier.

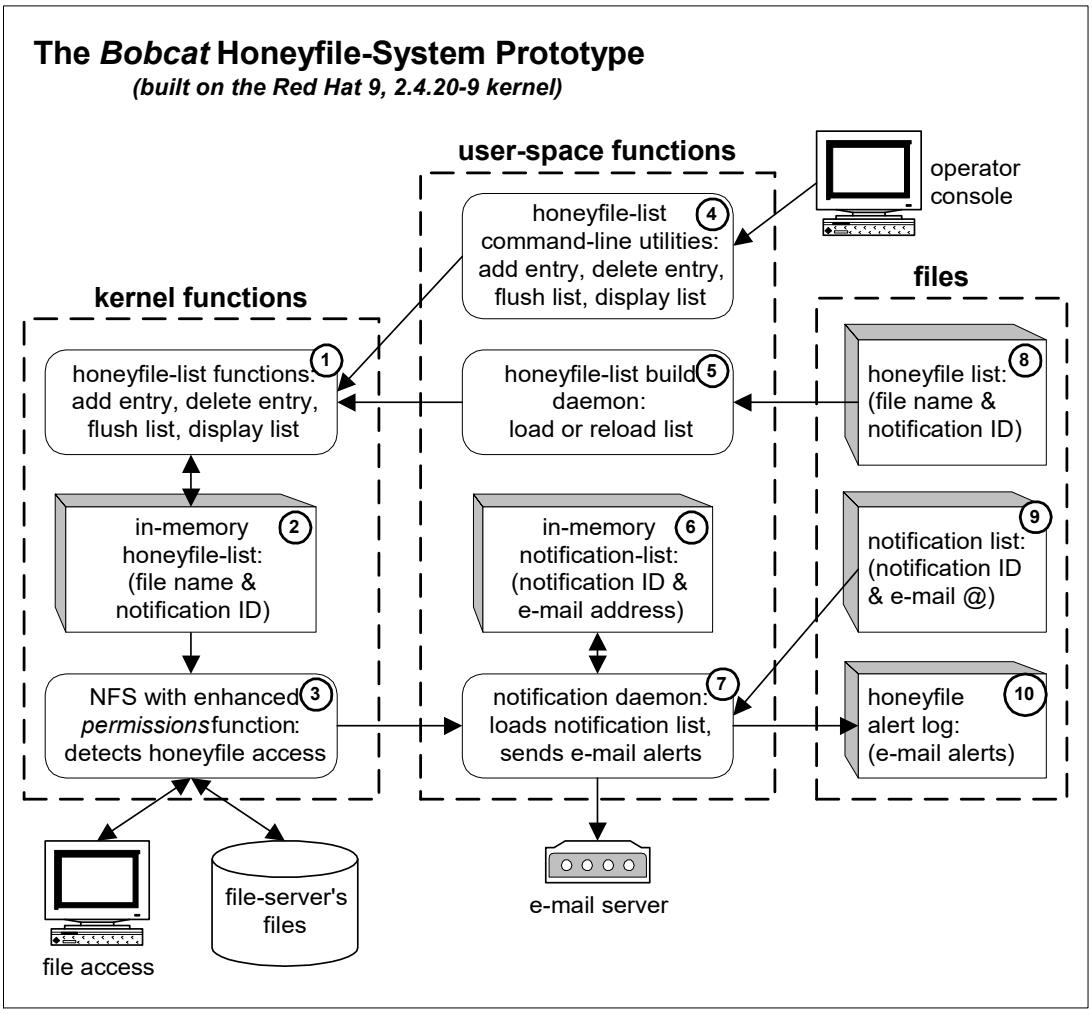


Figure 7 : The Bobcat Honeyfile-System Prototype

7.4.4 Field tests

The honeyfile test environment is described fully in the next chapter. The tests were still being conducted at the time of publication. Three hackers who tested the honeyfiles were interviewed. In summary, each of them were detected by at least one honeyfile. The primary lessons learned from the tests is that hackers may find the file space boring and not be inclined to search it diligently. This suggests that honeyfiles are more likely to be detected if they are near the file space's root, where the hacker will start searching. This section of the paper will be updated once the tests are completed and the results more fully analyzed.

8 Using honeynets to test security devices

This chapter describes the use of deception for testing a computer-security device, namely, the *honeyfile system* described in the prior chapter. A deceptive network, called a honeynet, was created to attract Internet hackers and monitor their activity. The honeyfile system was tested by deploying it on the honeynet and thereby subjecting it to hacking. The purpose of this research is to experiment with and understand: 1) honeyfiles, 2) the deception-operation process from earlier chapters, and 3) the use of honeynets to test computer-security devices.

We constructed a small honeynet, and several hacking incidents were observed. They were conducted by students from North Carolina State University who are skilled in computer security. Upcoming experiments will subject the honeynet to unknown Internet hackers.

The following sections describe:

- the honeynet implementation (page 71),
- the deception-operation principles used to design the honeynet (page 74), and
- the experimental results (page 75).

8.1 The honeynet

The honeynet for testing honeyfiles is shown in Figure 8 (page 73). The following sections describe the honeynet's deception objectives, deception story, and its implementation.

8.1.1 Deception objectives

The primary purpose of the honeynet is to test honeyfiles. Thus, the honeynet should represent a typical network setting for honeyfile use. Also, the honeynet should lure the honeyfile's typical hackers. The honeynet was designed to achieve these goals, with the available resources.

The honeynet's *deception target* is skilled script-kiddies. Experience reveals that honeynets connected to the Internet are usually attacked by script kiddies within a few hours or days [Spi03]. Targeting script kiddies ensures the honeynet will be hacked repeatedly within the time available. The skills required to break into the honeynet include locating a vulnerable network server, breaking into it using a publicly-available exploit, and using the server's Unix operating system to investigate the attached network. If less skill was required, then the honeynet's vulnerability might not be believable. Also, hackers without these skills pose less threat and are not as significant.

The *deception objective* defines the deception operation's desired outcome, namely, the *target action* and the *deception exploit*. For the honeynet, the target-action sought is typical network hacking. The deception exploit is learning about the effectiveness of honeyfiles and discovering ways to improve them.

For a deception operation, the *desired perception* is what the target must believe in order for him to carry out the target action. The target must perceive the honeynet as sufficiently interesting to break into and investigate. Also, the honeynet must be sufficiently obvious to be

discovered by targets within the time available. The desired perception is induced by the honeynet's deception story, which is described next.

8.1.2 The deception story

The honeynet portrays a deception story. In this story, the honeynet appears to be a back-door for a mainframe network, for use by maintenance personnel. The deceptive backdoor is implemented by actual computers, but the mainframe network is notional. [36] The deception story's computers are shown in Figure 8, and in the description below they are cited in *italics*.

The backdoor is the *outer SSH server* that resides on one of the network's DMZs, and the SSH server is protected by the *outer firewall*. From their home, maintenance personnel access the mainframe network by first connecting to the outer SSH server. From there, they connect through the *inner firewall*, to the *inner SSH server.* The outer SSH server has five fictitious user accounts. Each account's data space (/home) is mounted from the *file server* within the mainframe network.

The outer SSH server contains information that conveys the deception story. The server's login banner identifies it as a maintenance backdoor, to be used only in case of emergency. The banner further states that a firewall only permits connections to the outer SSH server, from the home IP address of approved personnel. The banner also contains a stern and legal-sounding security warning that was adapted from a bank's web site. As part of the deception story, the SSH server's user file-spaces have various mainframe computer manuals, as well as problem reports for a fictitious billing system. The user accounts' shell history-files portray users connecting to the inner SSH server and also downloading manuals.

To allow hackers into the network, the outer SSH server is vulnerable to remote attack, and exploits are readily available on hacker web-sites. By itself, this vulnerability would raise suspicion, as a real maintenance backdoor would be better secured. To make the vulnerability more believable, the server's banner states it is "heavily secured" and protected by a firewall. Hackers can attribute their unauthorized access to an exceptional streak of bad luck, for the defender. Once connected to the outer SSH server, a hacker can connect to the inner SSH server, but he can not break into it, i.e., it has no publicly announced vulnerabilities.

To test the honeyfile system, honeyfiles are placed in the user directories of the outer SSH server. A mainframe network was portrayed in order to make the honeynet sufficiently interesting for the hacker to investigate the user directories, e.g., in search of passwords to the inner SSH server.

[36] *Notional* elements of a deception operation do not exist, but are implied by implemented deceptions and by things that truely exist.

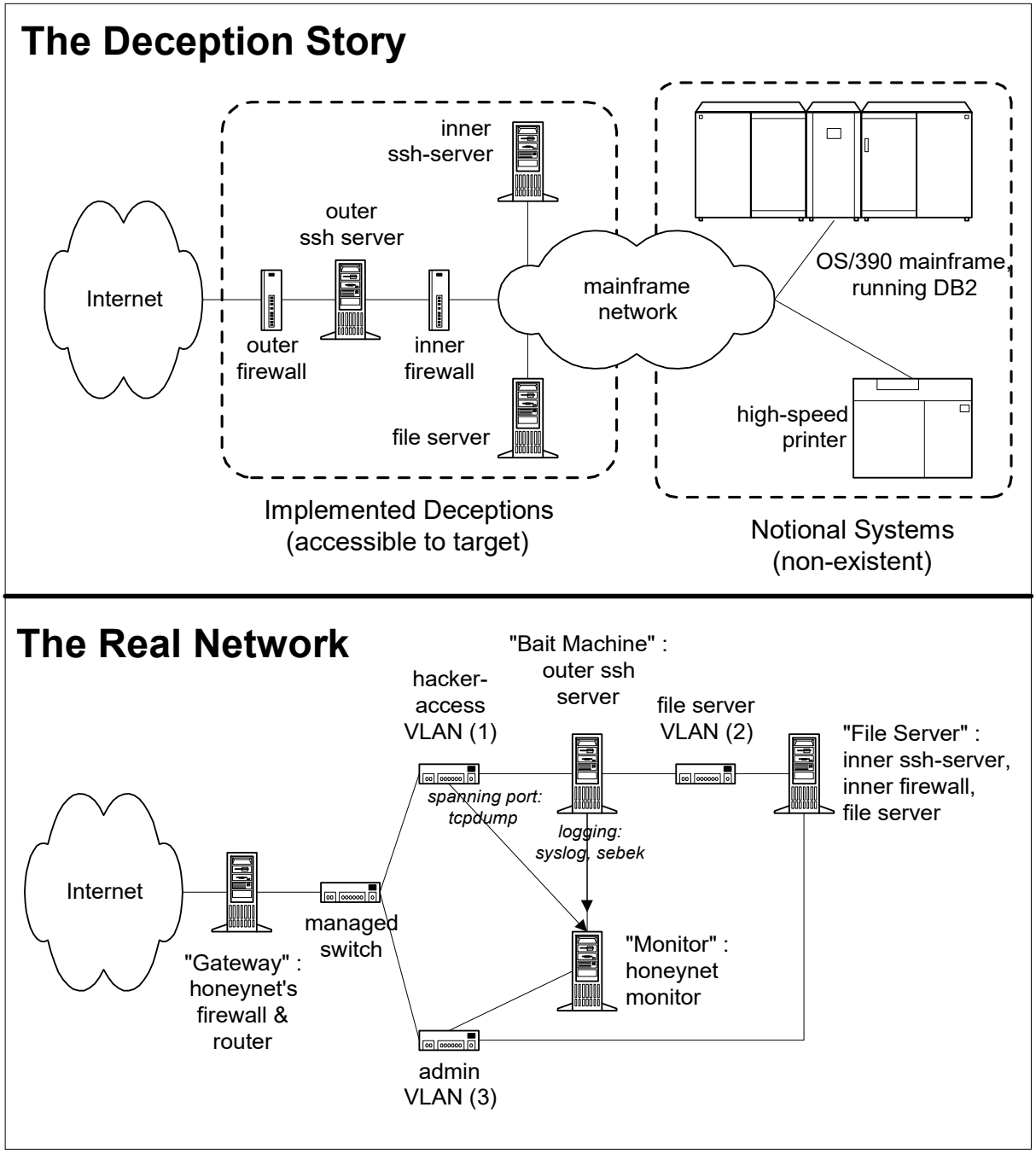


Figure 8 : The honeynet

The deception operation's termination plan is to simply block honeynet access once a target no longer provides useful information, or becomes a threat. When blocked, the target is likely to think he was detected by the backdoor's system administrators, and that its misconfigured firewall was fixed. All Internet access to the honeynet will be blocked initially, but it can be restored once unwanted visitors have lost interest. However, the firewall will permanently block the IP addresses of the targets that bring about termination.

8.1.3 Honeynet implementation

The honeynet is implemented by the real network shown in Figure 8. The real network's *gateway* is a RedHat 9 PC that implements the deception-story's outer firewall, and it provides network address translation (NAT). The honeynet uses a managed Ethernet switch that is divided into three VLANs, and a spanning port is used to sniff traffic, as shown in the figure. The honeynet uses three additional computers and each is a RedHat 9 PC. The real network's *file server* implements the deception-story's inner SSH server, inner firewall, and the file server. The file server runs the honeyfile system. Its email alerts are delivered as text messages to the honeynet operator's cell phone.

The real network's *bait machine* implements the deception-story's outer SSH server. The gateway's firewall does not allow connections to be initiated from the bait machine to computers outside of the honeynet. This prevents hackers from using the bait machine to attack other networks. On the bait machine, two logging facilities record the hacker's activity. A modified bash-shell logs all commands to a remote syslog server on the real network's *monitor*. The other logging facility is Sebek, a kernel module designed for use on honeypots. It logs almost all of a hacker's activity, including keystrokes, file uploads, and passwords. Sebek stealthily sends its log records to the *monitor*. Two logging facilities are used should one be discovered and disabled.

8.2 The deception operation

The honeynet deception-operation uses many of the principles and techniques presented in earlier chapters, and the most significant ones are described here.

The deception story is designed to reduce the risks of unintended target actions. In particular, the gateway's firewall prevents hackers from using the bait machine to attack other networks. However, hackers are likely to suspect a honeypot when they break into a computer that only allows incoming connections. The ruse of a secure maintenance backdoor was intended to make the firewall configuration plausible.

The deception operation is designed according to how the target, skilled script kiddies, works and thinks. The deception operation's target action is consistent with what the target is predisposed to do. Also, the vulnerable SSH server was chosen because it will almost certainly be detected by the targets' intelligence collection. There are two benefits in portraying a mainframe network: it is an appealing prize for the target, and secondly, script kiddies know little about mainframes, which makes it difficult for them to detect flaws in the deception story.

The deception story uses several techniques to minimize its falsehood, and this makes the deception easier to implement and less likely to be detected. The backdoor is a simple and isolated piece of a complex system, i.e., the mainframe network. The complex system's existence is implied, so it does not have be implemented. Also, the backdoor is something that would normally be hidden from the target, so he will not expect to find much information about the backdoor, and he will not expect that the story can be verified through other intelligence sources. By portraying a hidden piece of a complex system, only parts of the story needed to be implemented, and the target will piece together the rest. Falsehood was also minimized by not portraying the mainframe system's use. Portraying a specific use, such as a bank, requires

greater detail and makes the story more susceptible to detection.

The deception operation has two significant vulnerabilities to detection. Hackers may suspect a honeypot when they discover the gateway's firewall blocks outgoing connections. Secondly, hackers may discover the bait machine's logging facilities. The termination plan makes it possible to block targets if it appears they will detect the deception. After terminating the deception operation, it can be reused with the same IP address, or, to avoid prior targets, a different IP address can be used.

8.3 Experimental results

This section describes the experiments that were conducted, and our findings from them.

8.3.1 The experiment

The experiment involved building the honeynet and monitoring the honeynet's hacking incidents.

8.3.1.1 Building the honeynet

Building the honeynet provided some general insights regarding computer-security deception operations.

- System software and networking

Building the honeynet took much longer than expected, primarily due to our "rusty" system administration skills. With more experience and tools the honeynets could be built substantially better and faster. In general, building deceptive systems involves specialized skills. For neophytes, it may be economical to employ a skilled consultant.

The honeynet's Internet access was from an ISP's high-speed residential service. The ISP's security policy does not disallow honeynets, but we expected our honeynet would be forbidden if its hackers caused problems. We tried to avoid notice by the ISP's system administrators.

- Constructing deceptive content

Although the deception story uses little deceptive content, its construction proved challenging in several ways. The deception story's files need realistic file timestamps, to the extent that the target will examine them. Linux timestamps include access, modified and change times, and each may have to be set. [37] The Linux *touch* command can reset the access and modified times. We couldn't find a standard utility for resetting the change time, but it can be done.

Files created for the deception story must be given plausible timestamps. Also, when building or maintaining a honeypot, the files used will need their timestamps adjusted, and system log entries will need to be cleaned, such as shell history files, and login records. Our ad

[37] In Linux, the modified time is for the file contents, and the change time is for the file's meta information, such as its access permissions.

hoc treatment of timestamps and log entries proved difficult and error prone. Systematic procedures and tools would be very helpful.

As part of the deception story, five user accounts were created on the bait machine, and files were placed in three accounts to portray access to a mainframe network. The files included 25 mainframe manuals, four problem reports, and one password file. We fabricated the four problem reports and each contained a paragraph describing fictitious problems with mainframe operations. The password file contained a userid and password for the inner SSH server, but the password was invalid.

The real mainframe manuals were an inexpensive and convincing source of deceptive content. However, in spite of their brevity, writing the problem reports proved problematic. The problems encountered suggest two general principles. First, deceptive content can easily appear affected, perhaps because the author is consciously writing fiction for an audience. One safeguard is to have another person edit the fabricated content. Second, complex systems, such as a mainframe, often have their own specialized terminology. The terminology must be used correctly, as those who work with such systems can easily detect the non-standard diction of those who have not.

- Choosing an exploitable vulnerability

For most honeynets, its exploitable server is a key part of its deception story. The server must be chosen correctly as it can be very difficult to use a different exploitable server after the honeynet is constructed. Our deception story was designed around RedHat 9's SSH server, which had a recently-announced vulnerability. However, it wasn't until after building the honeynet that we discovered the vulnerability had no publicly-available exploits. This meant that no script kiddies could break into the honeynet, and we would have to use a different server.

There were two requirements for the server. First, the bait machine was already running RedHat 9, the latest version, and we did not have time install an older RedHat. Therefore, the server must have a publicly-available exploit that works against RedHat 9. Second, the server must fit plausibly within the deception story.

We found an SSH server with a publicly-available exploit that works against RedHat 9. However, the server and exploit were three years old. We expected the honeynet would not be attacked often by such an old exploit. Also, hackers might be suspicious of a three year old SSH server running on the most recent version of RedHat. We found other, more recent, vulnerable SSH servers, but their exploits did not work with RedHat 9.

We found that RedHat 9's SMB server has a recent exploit that is popular among script kiddies. However, after installing an SMB server we discovered our ISP blocks SMB access, making the server inaccessible to the Internet. Our project deadline was two weeks away, so we installed the SSH server with the three year-old exploit. To ensure the honeynet was hacked in time, we recruited penetration testers from among the skilled computer security students at North Carolina State University. Currently, the honeynet is only accessible to the testers, but it will be opened to Internet script kiddies later.

- Using surplus hardware and open-source software:

The honeynet has low performance requirements, so we built it using surplus equipment, for about $700. A switch with a spanning port was needed, so we bought an old 3Com SuperStack Switch on Ebay for $75. Maintaining the surplus equipment was costly, as we had to replace a hard drive, two power supplies, and two fans. We only used open source software, and it worked well.

8.3.1.2 Hacking incidents

The hacking tests were still being conducted at the time of publication. Three students completed the hacking and were interviewed. Overall the tests went well, and some useful lessons were learned. This section will be updated when the tests are completed and the results fully analyzed.

8.3.2 Findings

This section contains the findings after building the honeynet. As mentioned, the hacking tests were still being conducted at the time of publication. This section will be updated when the tests are completed.

8.3.2.1 The deception-operation process

This deception operation was designed using the principles and techniques presented in earlier chapters. They were helpful in three particular ways: 1) they enabled us to know and focus on key elements of the deception operation: the target, the deception objective, and minimizing the use of falsehood, 2) they provide an understanding of the overall deception process, and 3) they are an extensive checklist that ensures important elements of the deception operation are not overlooked.

One principle of deception operations is that there is a natural tendency for deception planners to be captivated by clever tricks, and to neglect to consider whether the tricks effectively achieve the deception objective. Knowing this principle was very helpful in recognizing when our ruses were delightful, but not the best way to achieve the deception objective.

8.3.2.2 Honeyfiles

Honeyfiles were used as an alarm mechanism for the honeynet itself, and this application of honeyfiles helped to confirm their usefulness. An alarm was needed to indicate the moment a hacker broke into the honeynet. Using the honeyfile system we placed alarms on the ls command and on the banner file displayed for login bash-shells. These alarms were almost certain to be triggered shortly after a hacker broke in. Further, we don't know of any better ways to make such an alarm, though they may exist.

8.3.2.3 Using honeynets to test security systems

In using the honeynet to test a computer-security system, several advantages and limitations came to light. The primary advantage is obvious: the honeynet recruits hackers who voluntarily, though unwittingly, perform testing. The testing is performed in a controlled environment with extensive feedback. Using real, but unknown hackers is a useful means for revealing faulty and overlooked assumptions, something that is very difficult for the system

developer to do himself.

In addition to its use with hackers, the honeynet provides a test bed in which the security device's testers can exercise its code. The honeynet also provides hands-on experience with deploying the security system. Another advantage of honeynets is that the test environment can be very realistic. For example, in our experiment, a network backdoor would be an excellent application for honeyfiles.

In addition to their strengths, honeynets also have some significant limitations for use in testing security systems. First, honeynet hackers are mostly script kiddies, just one type of hacker, and often the least threatening. Second, large honeynets are very expensive, due to the costs of creating deceptive content. Thus, in practice, the honeynets used for testing will be relatively small. Third, a honeynet's firewall typically must prevent its hackers from connecting back out to other networks. However, this firewall policy may reveal the honeynet to the hacker. Finally, there is potential for retribution from hackers who learn they were used as unwitting and unpaid testers. Also, there could be legal problems in so using hackers. However, when real hackers can not provide a useful test, penetration testers can be employed instead, as we had to do with our experiments.

8.4 Conclusion

The hacking tests were still being conducted at the time of publication. This section will be updated when the tests are completed. The following conclusions are based on our findings after building the honeynet.

Honeynets show much promise as a means for testing security devices. They provide a realistic setting in which hackers can test the device. The testing can be performed unwittingly by real hackers or by those recruited for the task. Testers from outside the development organization may reveal the developers' faulty and overlooked assumptions. Such errors are a common source of security vulnerabilities, and they are very difficult for developers to find themselves. The honeynets' primary limitation is that their expense limits their scale. However, costs could be lowered through further research and the development of tools and techniques.

9 Conclusion

Summarized here are: **1)** deception's promising uses, **2)** opportunities and challenges for the wide-spread use of deception, and **3)** topics for future research.

After years of research and development, computer security remains an error-prone task and, in some respects, a loosing battle. Computer security's chronic problems call for wholly new approaches. Deception works in a fundamentally different way than conventional security. Conventional security tends to work directly with the hacker's actions, e.g., to prevent them or to detect them. Deception manipulates the hacker's thinking to make him act in a way that is advantageous to the defender. Being fundamentally different, deception can be strong where conventional security is weak, as with honeyfiles. Deception is not always useful, but it can be an effective way to help secure conventional security's inherent vulnerabilities.

Some of the promising uses of deception identified in this paper are: **1)** deception can hide information needed for hacking. Currently, it is far too easy for hackers, and worms, to scan networks and find vulnerabilities. Deception can make that substantially harder, as the honeyd tool demonstrates. **2)** Deception can increase a network's internal security without making it harder for users to do their work. Honeyfiles are a good example. **3)** In creating deceptions, false content is very expensive to generate, e.g., documents or email. An inexpensive solution is mirroring existing content, e.g., a user's email folders. **4)** Honeyfiles are a simple, cheap and effective means of detecting attacks, and **5)** honeynets can be used to test security devices. Subjecting the devices to hacking is an effective means for revealing the developers' faulty assumptions.

Currently, deception is little used for computer security. There are several possible reasons: 1) a lack of deception skill among security personnel. 2) deception planning requires creativity, and not everyone is creative and 3) there are not many deception tools. Fortunately, there are solutions to these problems: 1) instruction in deception, such as this paper's, can provide the understanding needed for using deception. 2) Many companies out-source their network security, and this could include deception operations, and 3) deception tools and techniques are becoming increasingly available.

There are several fruitful topics for follow-on research: **1)** deception planners need an understanding of the means of deception. Our research on this topic is outlined in chapter 6 (page 53), and it is near completion. **2)** Honeyfiles appear promising, and a prototype system was built. A fully-functional system could be implemented within a commercial or open-source file server. Also, it could be implemented as a file-server front-end. **3)** Deception's greatest use may be in hiding the network information that is needed for hacking. The topic is ripe for both analytical and experimental research. **4)** Deception is an essential element of hacking. Counter-deception research would be useful for defenders. Research can include analysis and a survey of current practice. **5)** Further empirical ("hands-on") research is needed to learn the principles and techniques that are specific to computer-security deceptions.

Sources

This paper is derived from the military and intelligence deception literature. The primary sources, in descending order, are:

- U.S. Joint Forces deception manual: *Joint Doctrine for Military Deception* [JDD96]
- "Propositions on Military Deception", by Daniel and Herbig [DH82b]
- *The Art of Deception in Warfare*, by Dewar [Dew89]
- U.S. Army deception manual: *FM 90-2 Battlefield Deception* [USA78,USA88]
- CIA deception study: *Deception Maxims: Fact and Folklore* [CIA80]
- "Tactical Deception in Air-Land Warfare", by Fowler and Nesbit [FN95]
- "Cognitive Factors in Deception and Counterdeception", by Heuer [Heu81]
- USMC deception manual: *FM 15-6 Strategic and Operational Military Deception* [USM89]
- *Military Deception in Peace and War*, by Handel [Han85]

The general deception sources, in descending order, are:

- *Cheating and Deception* [BW82]
- "Hypotheses on Misperception" [Jer68]

The primary computer-security deception sources, in descending order, are:

- *Honeypots : Tracking Hackers* [Spi03]
- security web site: http://www.packetstormsecurity.org
- security web site: http://www.securityfocus.com

Bibliography

[ANO97] anonymous, *Maximum Security: A Hacker's Guide to Protecting Your Internet Site and Network*, Sams.net Publishing, 1997.

[Bar52] Barkas, G. *The Camouflage Story*, Cassell & Co. Ltd, 1952.

[Ban02] Bannan, K. "Watching You, Watching Me", *PC Magazine*, Ziff Davis Media, http://www.pcmag.com/article2/0,4149,396467,00.asp, July 1, 2002.

[BW82] Bell, J., B. Whaley. *Cheating and Deception*. Transaction Publishers, 1982.

[Cam96] Cameron, D. *Information Control in the Information Age*, The OPSEC Journal, 3rd ed., OPSEC Professionals Society, 1996.

[Cas00] Casey, E. *Digital Evidence and Computer Crime*, Academic Press, 2000.

[Cha81] Chaum, D. "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms", *Communications of the ACM*, 24(2), February 1981.

[Che92] Cheswick, B. "An evening with Berferd: In which a cracker is lured, endured and studied," *Proceedings of the Winter USENIX Conference*, January 1992.

[CIA80] *Deception Maxims: Fact and Folklore*, Deception Research Program, Office of Research and Development, Central Intelligence Agency, 1980.

[CMS01] Cohen, F., I. Marin, J. Sappington, C. Stewart, E. Thomas. "Red Teaming Experiments with Deception Technologies", internet-published manuscript: http://www.all.net, November 2001.

[Coh00] Cohen, F. "A Mathematical Structure of Simple Defensive Network Deceptions", *Computers & Security*, 19(200):520-528, August 2000.

[Dew89] Dewar, M. *The Art of Deception in Warfare*, David & Charles, 1989.

[DH82a] Daniel, D., K. Herbig, editors. *Strategic Military Deception*, Pergamon Press, 1982.

[DH82b] Daniel, D., K. Herbig. "Propositions on Military Deception", in [DH82a].

[DD97] Doctrine Division. *MCDP 1-3 Tactics*, http://www.doctrine.usmc.mil, U.S. Marine Corps, MCCDC, 1997.

[DD00] Doctrine Division. *MCWP 2-14 Counterintelligence*, http://www.doctrine.usmc.mil, U.S. Marine Corps, MCCDC, 2000.

[Dit01] Dittrich, D. "Results of the Forensic Challenge", The Honeynet Project, http://project.honeynet.org/challenge/results/index.html, 2001.

[FM97] Freedman, D.H. and C.C. Mann. *At Large: The Strange Case of the World's Biggest Internet Invasion*, Simon & Schuster, 1997.

[FN95] Fowler, C., R. Nesbit. "Tactical Deception in Air-Land Warfare", *Journal of Electronic Defense*, June 1995.

[Fre00] Freeh, L. "Statement of Louis J. Freeh, Director Federal Bureau of Investigation, before the Senate Committee on Appropriations", http://www.cybercrime.gov/docs.html, February 16, 2000.

[Ger99] Gertz, B. "Chinese hackers raid U.S. computers", *Washington Times*, May 16, 1999.

[Gla02] Glass, B. "Antisurveillance Tips & Tricks", *PC Magazine*, Ziff Davis Media, http://www.pcmag.com/article2/0,4149,396241,00.asp, July 1, 2002.

[Gru98] Grundschober, S. "Sniffer Detector Report", technical report, IBM Zurich Research Laboratory, 1998.

[Han85] Handel, M. *Military Deception in Peace and War*, Magnes Press, 1985.

[Heu81] Heuer, R. "Cognitive Factors in Deception and Counterdeception", in [DH82a].

[Hon01] The Honeynet Project. *Know Your Enemy : Revealing the Security Tools, Tactics, and Motives of the Blackhat Community*, Addison-Wesley, 2001.

[Hon04] The Honeynet Project. "Tools for Honeynets". Internet published http://www.honeynet.org/tools/index.html, 2004.

[How98] Howard, J. *An Analysis of Security Incidents on the Internet : 1989-1995*, PhD thesis, Carnegie Mellon University, August 1998.

[Jac75] Jackson, M. *Principles of Program Design*, Academic Press, 1975.

[JDJ01] Johnson, N., Z. Duric, S. G. Jajodia. *Information Hiding : Steganography and Watermarking - Attacks and Countermeasures*, Kluwer, 2001.

[JDD96] Joint Doctrine Division, *Joint Doctrine for Military Deception*, U.S. Joint Command, http://www.dtic.mil/doctrine, 1996.

[JDD01] Joint Doctrine Division, *DOD Dictionary of Military and Associated Terms*, U.S. Joint Command, http://www.dtic.mil/doctrine/jel/doddict/index.html, 2001.

[Jer68] Jervis, R. "Hypotheses on Misperception", *World Politics*, 20(3):454-479, April 1968.

[KH02] Kruse, W. and J. Heiser. *Computer Forensics : Incident Response Essentials*, Addison Wesley, 2002.

[Lic02] Lichtblau, E. "CIA Warns of Chinese Plans for Cyber-Attacks on U.S.; Defense", *Los Angeles Times*, pg. A.1, April 25, 2002.

[Lin01] Linn, A. "Bogus Company Used to Catch Hackers", Associated Press article, *The Las Vegas Sun*, May 5, 2001.

[Mon78] Montagu, E. *Beyond Top Secret Ultra*, McCann & Geoghegan, 1978.

[MP01] Mandia, K. and C. Prosise. *Incident Response : Investigating Computer Crime*, Osborne Press, 2001.

[MSK99] McClure, S., J. Scambray, and G. Kurtz. *Hacking Exposed : Network Security Secrets and Solutions*. Osborne/McGraw-Hill, 1999.

[Mur80] Mure, D. *Master of Deception*. William Kimber & Co. Ltd, 1980.

[NNS03] "The Five-Step OPSEC Process", National Nuclear Security Administration, U.S. DoE, Nevada Site Office, OPSEC Program Managers' Office, http://www.nv.doe.gov/opsec/fivestep.htm, 2003.

[OPS03] OPSEC Professionals Society. OPSEC Professionals Society web-site, http://www.opsec.org, 2003.

[RG03] Roesch, M., Green, C. *Snort Users Manual, Snort Release: 2.0.0*, http://www.snort.org, 2003.

[Riv98] Rivest, R. "Chaffing and Winnowing: Confidentiality without Encryption", MIT Lab for Computer Science, Internet published: http://theory.lcs.mit.edu/~rivest/chaffing.txt, April, 1998.

[RMA02] Rowe, N., B. Michael, M. Auguston, and R. Riehle. "Software Decoys for Software Counterintelligence", *IAnewsletter*, 5(1):10-12, Spring 2002.

[RSG98] Reed, M., P. Syverson, D. Goldschlag. "Anonymous Connections and Onion Routing", *IEEE Journal on Selected Areas in Communications*, 16(4):482-494, May, 1998.

[SNO00] Announcement for SNORT Release 1.6, http://packetstormsecurity.nl, 2000.

[SRH00] Schwabel, J., N. Rohring, M. Hall, E. Schultz. "'Lessons Learned' from Deploying a Honey Pot", *Information Security Bulletin*, 5(10):23-29f, December, 2000.

[Spi03] Spitzner, L. *Honeypots : Tracking Hackers*. Addison Wesley, 2003.

[Sto89] Stoll, C. *The cuckoo's egg : tracking a spy through the maze of computer espionage*, Doubleday, 1989.

[Tho02] Thomas, D. *Hacker Culture*, University of Minnesota Press, 2002.

[Tol75] Tolstoy, L. *Anna Karenina*, 1875.

[USA78] *FM 90-2 Tactical Deception*, U.S. Army, 1978.

[USA88] *FM 90-2 Battlefield Deception*, U.S. Army, 1988.

[USM89] *FM 15-6 Strategic and Operational Military Deception: U.S. Marines and the Next Twenty Years*, U.S. Marine Corps, 1989.

[Wha69] Whaley, B. *Stratagem : Deception and Surprise in War*, Center for International Studies, Cambridge, 1969.

[YWS00] Yuill, J., F. Wu, J. Settle, F. Gong, R. Forno, M. Huang and J. Asbery. "Intrusion-Detection for Incident-Response : using a military battlefield-intelligence process", *Computer Networks*, Elsevier, 34(4): 671-697, October 2000.

Authors

Jim Yuill is a PhD student in the Computer Science Department at North Carolina State University. This paper is part of his dissertation. Jim previously worked at IBM in operating systems development, for seven years. *jimyuill@gmail.com*

Fred Feer is retired from a career with the U.S. Army counterintelligence, CIA, RAND and independent consulting. The geographic area of prime focus is the Middle East. Deception has been an interest and area of professional specialization for over 40 years. *ffeer@comcast.net*

Dr. Dorothy Denning is a Professor in the Department of Defense Analysis at the Naval Postgraduate School. She has published 120 articles and four books. She is an ACM Fellow, and the recipient of several awards, including the National Computer Systems Security Award. In November 2001, she was named a Time magazine innovator. *dedennin@ nps.navy.mil*

Dr. Bowyer Bell was a leading authority on terrorism and the author of many books. He was a professor in the School of International and Public Affairs at Columbia University. His book *Cheating and Deception* [BW82] is a primary source for the USMC manual on strategic deception [USM89]. Sadly, Dr. Bell passed away before this paper was completed.