2

DECEPTION

CHAPTER OUTLINE

Create a highly controlled network. Within that network, you place production systems and then monitor, capture, and analyze all activity that happens within that network Because this is not a production network, but rather our Honeynet, any traffic is suspicious by nature.

The Honeynet Project[1]

The use of deception in computing involves deliberately misleading an adversary by creating a system component that looks real but is in fact a trap. The system component, sometimes referred to as a *honey pot*, is usually functionality embedded in a computing or networking system, but it can also be a physical asset designed to trick an intruder. In both cases, a common interface is presented to an adversary who might access real functionality connected to real assets, but who might also

[1]The Honeynet Project, *Know Your Enemy: Revealing the Security Tools, Tactics, and Motives of the Blackhat Community*, Addison–Wesley Professional, New York, 2002. (I highly recommend this amazing and original book.) See also B. Cheswick and S. Bellovin, *Firewalls and Internet Security: Repelling the Wily Hacker*, 1st ed., Addison–Wesley Professional, New York, 1994; C. Stoll, *The Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage*, Pocket Books, New York, 2005.

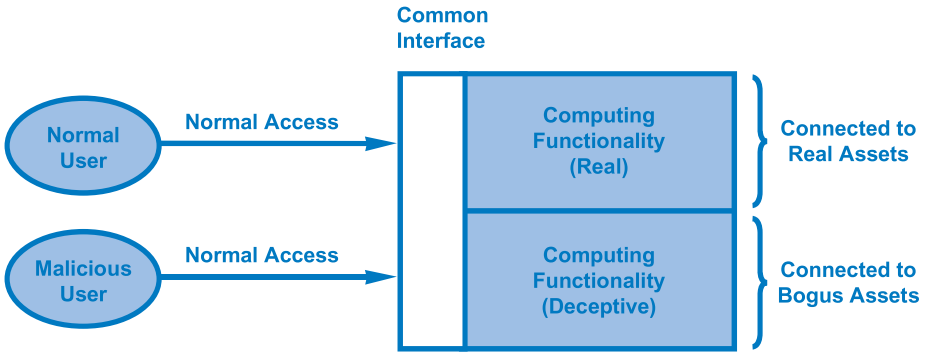


Figure 2.1 Use of deception in computing.

unknowingly access deceptive functionality connected to bogus assets. In a well-designed deceptive system, the distinction between real and trap functionality should not be apparent to the intruder (see Figure 2.1).

The purpose of deception, ultimately, is to enhance security, so in the context of national infrastructure it can be used for large-scale protection of assets. The reason why deception works is that it helps accomplish any or all of the following four security objectives:

- *Attention*—The attention of an adversary can be diverted from real assets toward bogus ones.
- *Energy*—The valuable time and energy of an adversary can be wasted on bogus targets.
- *Uncertainty*—Uncertainty can be created around the veracity of a discovered vulnerability.
- *Analysis*—A basis can be provided for real-time security analysis of adversary behavior.

The fact that deception diverts the attention of adversaries, while also wasting their time and energy, should be familiar to anyone who has ever used a honey pot on a network. As long as the trap is set properly and the honey pot is sufficiently realistic, adversaries might direct their time, attention, and energy toward something that is useless from an attack perspective. They might even plant time bombs in trap functionality that they believe will be of subsequent use in targeting real assets. Obviously, in a honey pot, this is not the case. This type of deception is a powerful deterrent, because it defuses a cyber attack in a way that could fool an adversary for an extended period of time.

The possibility that deception might create uncertainty around the veracity of a discovered vulnerability has been poorly explored to date. The idea here is that when an intruder inevitably stumbles onto an exploitable hole it would be nice if that intruder were led to believe that the hole might be a trap. Thus, under the right circumstances, the intruder might actually choose to avoid exploitation of a vulnerability for fear that it

has been intentionally planted. While this might seem difficult to implement in many settings, the concept is powerful because it allows security managers to defuse existing vulnerabilities *without even knowing about them*. This is a significant enough concept that it deserves repeating: The use of deception in computing allows system security managers to reduce the risk of vulnerabilities *that they might not even know are present*.

Deception is a powerful security tool, as it protects even unknown vulnerabilities.

The fact that real-time analysis can be performed on a honey pot is reasonably well known in the computing community today. Perhaps this is because it is a widely accepted best practice that security administrators should try to observe the behavior of intruders that have been detected. Most intrusion detection systems, for example, include threat management back-end systems that are designed to support such an objective. In the best case, the forensic analysis gathered during deception is sufficiently detailed to allow for identification of the adversary and possibly even prosecution. In the most typical case, however, accurate traceability to the original human source of a problem is rarely accomplished.

Luckily, the success of deceptive traps is assisted by the fact that intruders will almost always view designers and operators of national assets as being sloppy in their actions, deficient in their training, and incompetent in their knowledge. This extremely negative opinion of the individuals running national infrastructure is a core belief in virtually every hacking community in the world (and is arguably justified in some environments). Ironically, this low expectation is an important element that helps make stealth deception much more feasible, because honey pots do not always have to mimic a perfectly managed environment. Instead, adversaries can generally be led to find a system environment that is poorly administered, and they will not bat an eyelash. This helps the deception designer.

Honey pots should not necessarily mimic perfect environments.

The less well-understood case of openly advertised deception relies on the adversary believing that designers and operators of national assets are competent enough to plant a believable trap into a national asset. This view represents a hurdle, because the hacking community will need to see convincing evidence before they will ever believe that anyone associated with a large organization would be competent enough to manage a complex program of deceptive computing. This is too bad, because open use of deception carries great advantages, as we will explain in more detail below. In any event, the psychology of understanding and managing adversary views is not straightforward. This soft issue must become part of the national infrastructure protection equation but will obviously require a new set of skills among security experts.

Effective cyber deception involves understanding your adversary.

The most common implementation of deception involves the insertion of fake attack entry points, such as open service ports,

that adversaries might expect to see in a normal system. The hope is that an adversary would discover (perhaps with a scanner) and then connect to these open service ports, which would in turn then lead to a honey pot. As suggested above, creating realism in a honey pot is not an easy task, but several design options do exist. One approach involves routing inbound open port connections to physically separate bogus systems that are isolated from real assets. This allows for a "forklift"-type copying of real functionality (perhaps with sensitive data sanitized) to an isolated, safe location where no real damage can be done.

Recall that, if the deception is advertised openly, the possibility arises that an adversary will not bother to attempt an attack. Admittedly, this scenario is a stretch, but the possibility does arise and is worth mentioning. Nevertheless, we will assume for the balance of this discussion that the adversary finds the deceptive entry point, presumes that it is real, and decides to move forward with an attack. If the subsequent deception is properly managed, then the adversary should be led down a controlled process path with four distinct attack stages: *scanning*, *discovery*, *exploitation*, and *exposing* (see Figure 2.2).

During the initial scanning stage, an adversary is searching through whatever means is available for exploitable entry points. The presumption in this stage is that the service interface includes trap functionality, such as bogus links on proxied websites that lead to a honey pot for collecting information. It is worth noting, however, that this "searching" process does not always imply the use of a network by an adversary. Instead, the adversary might be searching for exploitable entry points in contracts, processes, locked cabinets, safes, or even relationships with national infrastructure personnel. In practice, one might even expect a combination of computing and noncomput-

Bear in mind that a cyber honey pot might require coordination with a tangible exploitable point outside the cyber world.

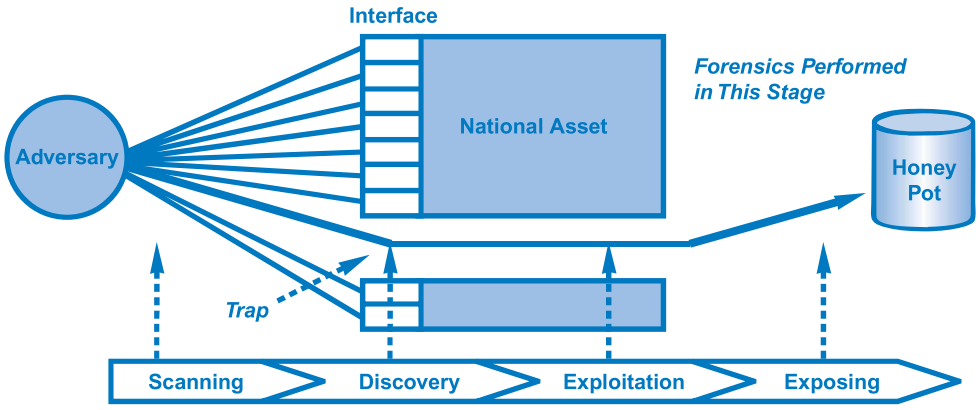


Figure 2.2 Stages of deception for national infrastructure protection.

ing searches for information about exploitable entry points. The deception must be designed accordingly.

During the discovery phase, an adversary finds an exploitable entry point, which might be real or fake. If the vulnerability is real, then one hopes that good back-end security is in place to avoid an infrastructure disaster. Nevertheless, the decision on the part of the intruder to exploit a discovered vulnerability, real or fake, is an important trigger point. Good infrastructure security systems would need to connect this exploitation point to a threat management system that would either open a security trouble ticket or would alert a security administrator that an intruder has either started an attack or fallen for the deceptive bait. Obviously, such alerts should not signal an intruder that a trap is present.

During the exploitation stage, the adversary makes use of the discovered vulnerability for whatever purposes they might have. If the vulnerability is real, then the usual infrastructure break-in scenario results. If the vulnerability is a trap, however, then its effectiveness will be directly related to the realism of the honey pot. For both stealth and non-stealth deception, this is the initial stage during which data becomes available for forensic analysis. A design consideration is that the actual asset must never become compromised as a result of the trap. This requirement will likely result in deceptive functionality running on computing "islands" that are functionally separated from the real assets.

Actual assets must remain separate and protected so they are not compromised by a honey pot trap.

During the exposing stage in deception, adversary behavior becomes available for observation. Honey pots should include sufficient monitoring to expose adversary technique, intent, and identity. This is generally the stage during which management decisions are made about whether response actions are warranted. It is also a stage where real-time human actions are often required to help make the deceptive functionality look real. As we stated above, a great advantage that arises here is the low expectation the adversary will have regarding system administrative competency on the part of the infrastructure team. This allows the security team to use the excuse of poor setup to cover functional gaps that might exist in the deception.

Monitoring honey pots takes security to the next level: potential for responsive action.

Any one of the four stages of deception can raise significant legal and social issues, so any program of national infrastructure protection must have participation from the national legal community to determine what is considered acceptable. The difference between a passive trap and an active lure, for example, is subtle and must be clarified before a live deployment is made into infrastructure. From a social perspective, one might hope that the acceptance that exists for using deception to catch online stalkers would be extended to the cyber security community for catching adversaries targeting national infrastructure.

Scanning Stage

In this first stage, the presumption is that an adversary is scanning whatever is available to find exploitation points to attack national infrastructure. This scanning can include online searches for web-based information, network scans to determine port availability, and even offline searches of documents for relevant information. Deception can be used to divert these scanning attempts by creating false entry points with planted vulnerabilities. To deal with the offline case, the deception can extend to noncomputing situations such as intentionally leaving a normally locked cabinet or safe door open with bogus documents inserted to deceive a malicious insider.

The deceptive design goal during scanning is to make available an interface with three distinct components: *authorized services*, *real vulnerabilities*, and *bogus vulnerabilities*. In a perfect world, there would be no vulnerabilities, only authorized services. Unfortunately, given the extreme complexity associated with national infrastructure services, this is an unrealistic expectation, so real vulnerabilities will always be present in some way, shape, or form. When deception is used, these real vulnerabilities are complemented by fake ones and should be indistinguishable. Thus, an adversary will see three components when presented with a national asset interface with deception (see Figure 2.3).

Bogus vulnerabilities will generally be inserted based on the usual sorts of problems found in software. This is one of the few cases where the deficiencies of the software engineering discipline can actually be put to good use for security. One might imagine situations where new vulnerabilities are discovered and then immediately implemented as traps in systems that require protection. Nevertheless, planted holes do not always have to be based on such exploitable software bugs or system misconfigurations. In some cases, they might correspond to properly administered functionality, but that might not be considered acceptable for local use.

Honey Pots can be Built into Websites

A good example of a trap based on properly administered functionality might be a promiscuous tab on a website that openly solicits leaks of information; this is found sometimes on some of the more controversial blog sites. If legal and policy acceptance is given, then these links might be connected in a local proxied Intranet to a honey pot collection site. Insiders to an organization might then consider leaking information directly using this link to the seemingly valid Internet site, only to be duped into providing the leak to the local security team. Again, this should only be considered for deployment if all legal and policy requirements are met, but the example does help illustrate the possibilities.

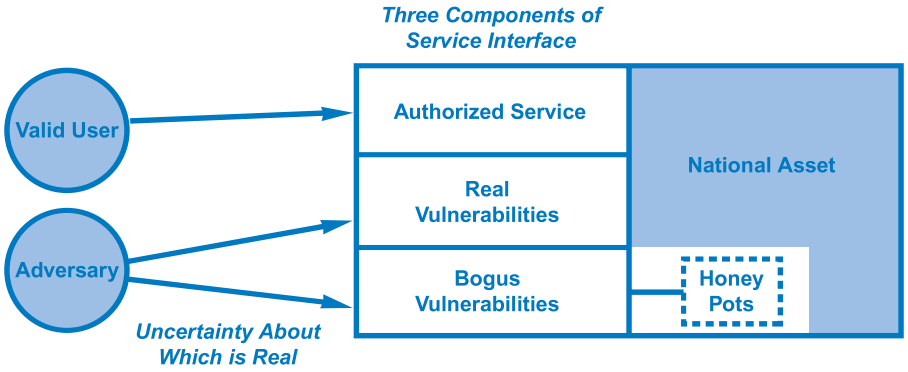


Figure 2.3 National asset service interface with deception.

A prominent goal of deception is to observe the adversary in action. This is done via real-time collection of data about intruder activity, along with reasoned analysis about intent. For example, if the intruder seems to be guessing passwords over and over again to gain access to a honey pot system, the administrator might decide in real time to simply grant access. A great challenge is that the automation possibilities of such response are not currently well understood and are barely included in security research programs. This is too bad, because such cases could really challenge and ultimately improve the skills of a good security administrator. One could even imagine national groups sponsoring contests between live intruders and live administrators who are battling against each other in real time in a contrived honey pot.

Allowing an intruder access increases your risk level but also allows the security administrator to monitor the intruder's moves.

Deliberately Open Ports

Intruders routinely search the Internet for servers that allow connections to exploitable inbound services. These services are exploitable generally because they contain some weakness such as a buffer overflow condition that can be tripped to gain privileged access. Once privileged access is obtained, the intruder can perform administrative tasks such as changing system files, installing malware, and stealing sensitive information. All good system administrators understand the importance of *hardening* servers by disabling all exploitable and unnecessary services. The problem is that hardening is a complex process that is made more difficult in environments where the operating system is proprietary and less transparent. Amazingly, most software and server vendors still deliver their products in configurations that include most services being default enabled.

The deliberate insertion of open service ports on an Internet-facing server is the most straightforward of all deceptive computing practices. The deliberately open ports are connected to

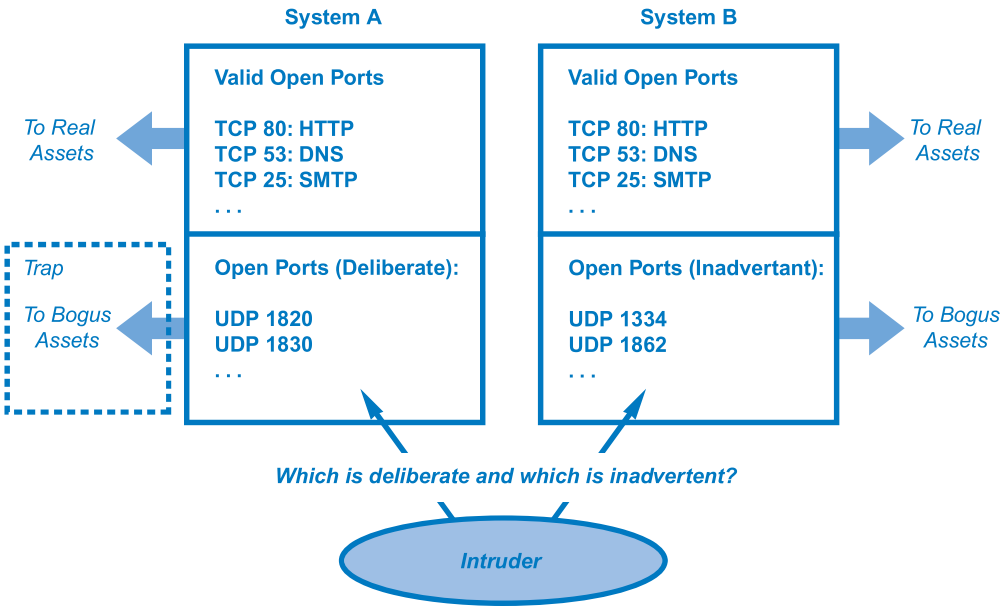


Figure 2.4 Use of deceptive open ports to bogus assets.

back-end honey pot functionality, which is connected to monitoring systems for the purpose of observation and analysis. The result is that servers would thus present adversaries of national infrastructure with three different views of open service ports: (1) valid open ports one might expect, such as HTTP, DNS, and SMTP; (2) open ports that are inadvertently left open and might correspond to exploitable software; and (3) open ports that are deliberately inserted and connected to bogus assets in a honey pot. As long as it is generally understood that deception could *potentially* be deployed, there could be some uncertainty on the part of the adversary about which open ports are deliberate and which are inadvertent (see Figure 2.4).

Another challenge is for security managers to knowingly keep open ports after running scanners that highlight these vulnerabilities.

Security managers who use port scanners as part of a normal program of enterprise network protection often cringe at this use of deception. What happens is that their scanners will find these open ports, which will result in the generation of reports that highlight the presumed vulnerabilities to managers, users, and auditors. Certainly, the output can be manually cropped to avoid such exposure, but this might not scale well to a large enterprise. Unfortunately, solutions are not easily identified that solve this incompatibility between the authorized use of port scanners and the deliberate use of open ports as traps. It represents yet another area for research and development in deceptive computing.

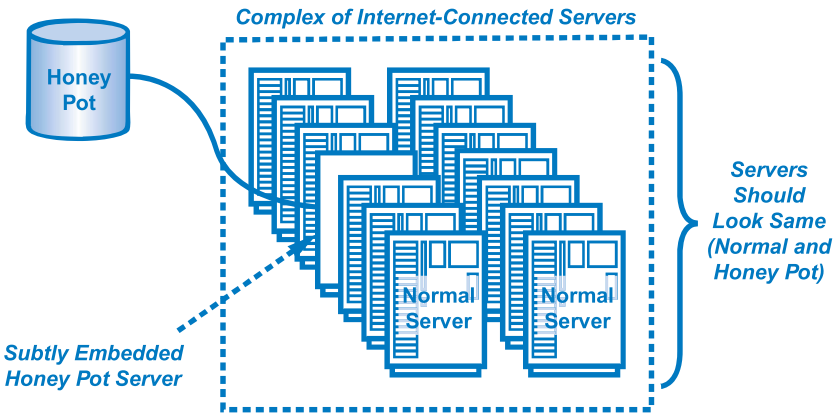


Figure 2.5 Embedding a honey pot server into a normal server complex.

An additional consideration with the deliberate use of open ports is that care must be taken on the back end to ensure that real assets cannot be exploited. Not surprisingly, practical techniques for doing this are not well known. For example, if the back-end deceptive software connected to deliberately open ports shares resources with valid assets, then the potential exists for negative side effects. The only reasonable approach today would involve deliberately open ports on bogus servers that are honey pots with no valid resources. These servers should be subtly embedded into server complexes so they look normal, but they should be hardwired to separate honey pot assets. This reduces the likelihood of negative side effects on normal servers (see Figure 2.5).

In practice, the real challenge to the deceptive use of open ports is creating port-connected functionality that is sufficiently valid to fool an expert adversary but also properly separated from valid services so no adversary could make use of the honey pot to advance an attack. Because computer science does not currently offer much foundational assistance in this regard, national infrastructure protection initiatives must include immediate programs of research and development to push this technique forward.

Discovery Stage

The discovery stage corresponds to the adversary finding and accepting the security bait embedded in the trap. The two corresponding security goals during this stage are to make an intruder believe that real vulnerabilities could be bogus and that bogus vulnerabilities could be real. The first of these goals is accomplished by making the deception program well-established and

openly known. Specific techniques for doing this include the following:

- *Sponsored research*—The use of deception in national infrastructure could become generally presumed through the open sponsorship and funding of unclassified research and development work in this area.
- *Published case studies*—The open publication of case studies where deception has been used effectively in national asset protection increases the likelihood that an adversary might consider a found vulnerability to be deliberate.
- *Open solicitations*—Requests for Information (RFIs) and Requests for Proposals (RFPs) should be openly issued by national asset protectors. This implies that funding must be directed toward security projects that would actually use deceptive methods.

Openly advertised use of deception may cause adversaries to question whether a discovered vulnerability is valid or bogus.

Interestingly, the potential that an adversary will hesitate before exploiting a real vulnerability increases only when the use of deception appears to be a real possibility. It would seem a hollow goal, for example, to simply announce that deception is being used without honest efforts to really deploy such deceptions in national infrastructure. This is akin to placing a home protection sign in the landscaping without ever installing a real security system. For openly advertised deception to work, the national infrastructure team must be fully committed to actually doing the engineering, deployment, and operation.

The second goal of making bogus vulnerabilities look real will be familiar to computer security experts who have considered the use of honey pots. The technique of duplication is often used in honey pot design, where a bogus system is a perfect copy of a real one but without the back-end connectivity to the real asset being protected. This is generally done by duplicating the front-end interface to a real system and placing the duplicate next to a back-end honey pot. Duplication greatly increases realism and is actually quite easy to implement in practice (see Figure 2.6).

Turn discovered vulnerabilities into advantages by mimicking them in honey pot traps.

As suggested above, the advantage of duplication in honey pot design is that it maximizes authenticity. If one finds, for example, a real vulnerability in some front-end server, then an image of that vulnerable server could be used in future deceptive configurations. Programs of national infrastructure protection should thus find ways to effectively connect vulnerability discovery processes to honey pot design. Thus, when a truly interesting vulnerability is found, it can become the front end to a future deceptive trap.

Deceptive Documents

The creation and special placement of deceptive documents is an example method for tricking adversaries during discovery.

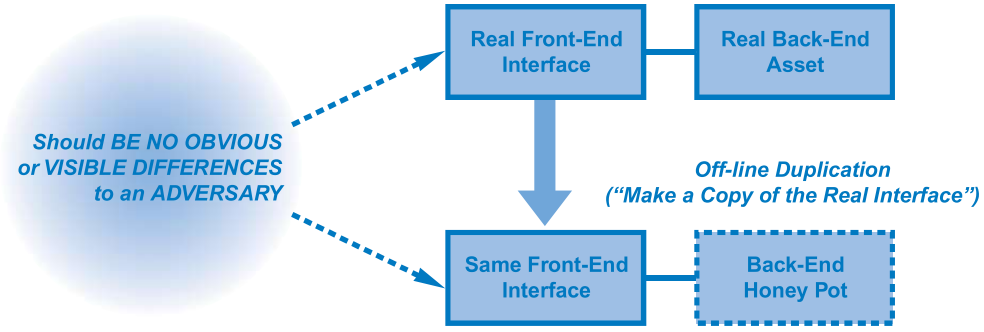


Figure 2.6 Duplication in honey pot design.

This technique, which can be done electronically or manually, is especially useful for detecting the presence of a malicious insider and will only work under two conditions:

- *Content*—The bogus document must include information that is convincingly realistic. Duplication of a valid document with changes to the most sensitive components is a straightforward means for doing this.
- *Protection*—The placement of the bogus document should include sufficient protections to make the document appear truly realistic. If the protection approach is thin, then this will raise immediate suspicion. Sabotage can be detected by protecting the bogus document in an environment that cannot be accessed by anyone other than trusted insiders.

An illustrative approach for national infrastructure protection would follow these steps: First, a document is created with information that references a specially created bogus asset, such as a phone number, physical location, or server. The information should never be real, but it should be very realistic. Next, the document is stored in a highly protected location, such as a locked safe (computer or physical). The presumption is that under normal circumstances the document should sit idly in the locked safe, as it should have no real purpose to anyone. Finally, the specially created bogus asset is monitored carefully for any attempted compromise. If someone finds and grabs the document, then one can conclude that some insider is not to be trusted.

Steps to Planting a Bogus Document

To effectively plant a bogus document, consider following these steps:

1. Create a file with instructions for obtaining what would appear to be extremely sensitive information. The file could include a phone number, an Internet address for a server, and perhaps a room location in some hotel.

2. Encrypt the file and store it on a server (or print and lock it in a safe) that one would presume to be protected from inside or outside access.
3. Put monitoring of the server or safe in place, with no expectation of a time limit. In fact, the monitoring might go on indefinitely, because one would expect to see no correlative behavior on these monitored assets (see Figure 2.7).

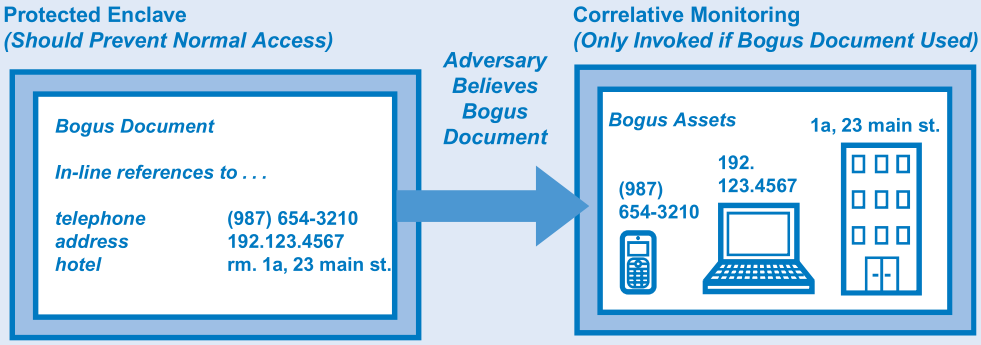


Figure 2.7 Planting a bogus document in a protected enclave.

It should be obvious that the example scheme shown in Figure 2.7 works as well for an electronic document protected by encryption and access control as for a manual paper document locked in a protected safe. In both cases, one would expect that no one would ever correlate these bogus references. If it turns out that the monitoring shows access to these bogus assets in some related way, then one would have to assume that the protected enclave has been compromised. (Monitoring a hotel might require complex logistics, such as the use of hidden cameras.) In any event, these assets would provide a platform for subsequent analysis of exploitation activity by the adversary.

Exploitation Stage

The third stage of the deception lifecycle for an adversary involves exploitation of a discovered vulnerability. This is a key step in the decision process for an adversary because it is usually the first stage in which policy rules or even laws are actually violated. That is, when an intruder begins to create a cyber attack, the initial steps are preparatory and generally do not violate any specific policy rules or laws. Sometimes security experts refer to this early activity as *low radar actions*, and when they are detected they are referred to as *indications and warnings*. Determining whether to respond to indications and warnings is a challenge, because response requires time and energy. If the track record of the security team involves many response actions to indications

and warnings that are largely false positives, then the organization is often tempted to reduce the response trigger point. This is a bad idea for national infrastructure, because the chances increase that a real event will occur that is not responded to promptly.

Responding to a large number of false positives is necessary to adequately protect national infrastructure.

As such, the protection of national infrastructure should involve a mind shift away from trying to reduce false positive responses to indications and warnings. Instead, the goal should be to deal with all instances in which indication and warning actions would appear to be building up to the threshold at which exploitation begins. This is especially important, because this threshold marks the first stage during which real assets, if targeted, might actually be damaged (see Figure 2.8).

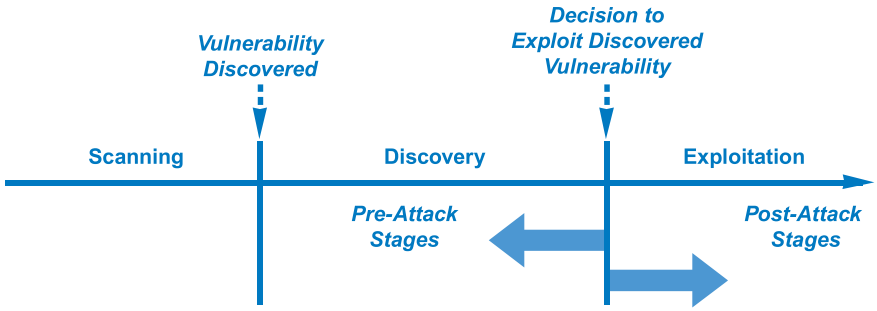
The key requirement at this decision point is that any exploitation of a bogus asset must not cause disclosure, integrity, theft, or availability problems with any real asset. Such noninterference between bogus and real assets is easiest to accomplish when these assets are kept as separate as possible. Physical separation of assets is straightforward; a real software application with real data, for example, could be separated from a bogus application with fake data by simply hosting each on different servers, perhaps even on different networks. This is how most honey pots operate, and the risk of interference is generally low.

Achieving noninterference in an environment where resources are shared between real and fake assets is more challenging. To accomplish this goal, the deception designer must be creative. For example, if some business process is to be shared by both real and fake functionality, then care must be taken by the deception operators to ensure that real systems are not degraded in any way. Very little research has been done in this area, especially for availability threats. Allowing a malicious adversary to execute programs on a live, valid system, for example, would provide opportunities for malicious resource exhaustion. Nevertheless, the general approach has considerable promise and deserves more attention.

When bogus and real assets reside on the same server, vulnerability risk increases dramatically.

A related issue involves the possibility that intrusion detection and incident response systems might be fooled during exploitation

Figure 2.8 Pre- and post-attack stages at the exploitation stage.



into believing that trap functionality is real. White hat teams in companies have dealt with this problem for years, and they must coordinate with security teams to ensure that their activity does not cause a false alarm. This can be accomplished in several ways:

- *Process coordination*—This involves the honey pot team coordinating their activities in advance with the infrastructure security teams. The problem is that it tends to highlight the deception and might destroy some hidden transparency, especially if the deception is designed to detect malicious insiders.
- *Trap isolation*—This involves making certain that security systems, such as intrusion detection probes, are not in the deceptive path. Ongoing architectural analysis is required in this case to make sure that the condition holds throughout the system lifecycle.
- *Back-end insiders*—If the security team includes some trusted insiders involved in response activities, then they might be able to ensure that a response to deceptive action does not waste time and resources. This works best if the insider is a decision-maker.
- *Process allowance*—In this case, the detection and response activities are allowed to proceed. This is the recommended case if the deception is considered sensitive and must be totally hidden from insiders.

The exploitation stage is the most dangerous part of the deceptive lifecycle because it involves real malicious activity from an adversary. Infrastructure teams must learn to understand and respect the potential for real damage if this stage is not managed carefully.

Procurement Tricks

One way to understand adversary behavior is to compare it in different environments. An example involves a deception designer creating two separate supplier solicitations for a given product or service. One solicitation would be for a benign, noncritical, nonsensitive application; the other would be for an obviously sensitive, nationally critical application. In both cases, *exactly the same product or service* would be requested, but when they are delivered to each application detailed examination would be initiated to identify differences. Any differences found in the delivered product by the intentionally planted "critical buyer" and "benign buyer" would be viewed as potential exploits (see Figure 2.9).

The deception described above only works if sufficient forensic capability exists to compare the two delivered products. For any product or service, this could include comparison of relative software size, system performance, product documentation, service functionality, or technical support. One could even imagine a second level of deception using social engineering where an impassioned plea would be made to the supplier for some undocumented type of

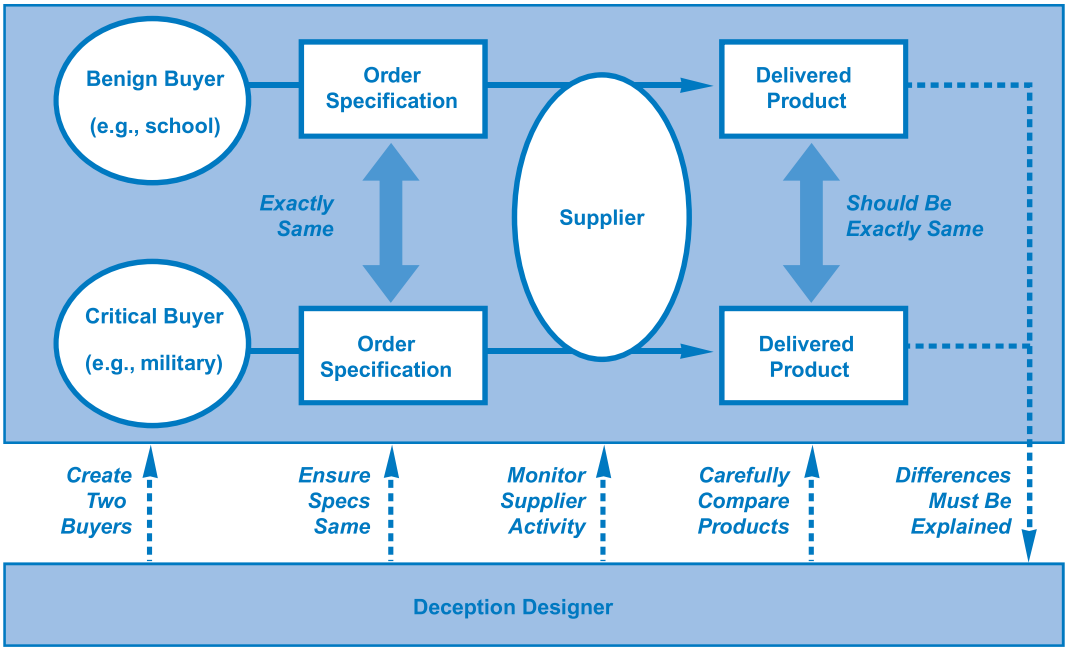


Figure 2.9 Using deception against malicious suppliers.

emergency support, usually remote administration. If either of the delivered products is set up for such remote administration, then the national asset manager would know that something is wrong.

The procurement lifecycle is one of the most underestimated components in national infrastructure protection from an attack perspective. Generally, security teams focus on selecting, testing, installing, and operating functionality, with seemingly mundane procurement tasks left to the supply chain team. This is a huge mistake, and adversaries understand this point well. Thus, national infrastructure protection initiatives must extend to the procurement process, and the clever use of deception is a powerful tool in this regard.

National infrastructure protection must extend from procurement to operating functionality in order to be truly effective.

Exposing Stage

The final stage in the deception lifecycle involves the adversary exposing behavior to the deception operator. Presumably, in this stage, the adversary is now hacking away at the trap functionality, convinced that all systems and assets are real. All sorts of possibilities arise in terms of how this hacking will proceed. It could be a flurry of intense activity in a short period of time or it could be a drawn-out process of low and slow actions, so the deception team must have patience. Also, during this stage, the adversary might expose the use of well-known hacking techniques and tools or, alternatively, could demonstrate use of techniques not previously seen by the security team (see Figure 2.10).

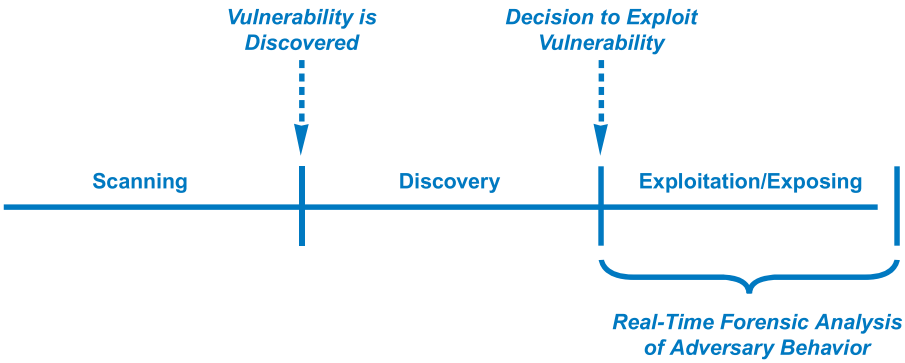


Figure 2.10 Adversary exposing stage during deception.

Observing intruder activity can be an informative but risky process during the exposure stage.

The challenge in this stage is that the deception must allow a window for observation of intruder activity, but must also be hidden. It must provide a convenient interface for the deception team to collect data but also must provide a way for commands to be issued or changes to be made in real time. Unfortunately, few commercial products exist that are designed to support these features. Specific functional requirements for the monitoring environment during the exposing stage of deception include the following:

- *Sufficient detail*—The monitoring environment must provide sufficient detail so the deception operator can determine what is going on. For example, overly cryptic audit logs in terse format with gaps in certain areas would not be the best approach. The usual administrative interface to an operating system (generally through a command interface) is often the most desirable approach. One should not expect fancy, colorful displays for the monitoring task because most security analysts prefer the convenience of a terse command line interface.
- *Hidden probes*—Monitoring in this stage of the deception works only if it is completely hidden. If an adversary figures out that someone is watching, then behavior modification would occur immediately. Simple tasks must therefore be implemented such as suppressed listing of any processes launched by the deception team (unless desired). The art of creating realistic functionality to hide probes requires support and nurturing in the security community.
- *Real-time observation*—The deception operator should have access to information about exposed behavior as it happens. The degree of real time for such monitoring (e.g., instantaneous, within seconds, within minutes) would depend on the local circumstances. In most cases, this observation is simply done by watching system logs, but more advanced tools are required to record and store information about intruder behavior.

As we suggested above, in all cases of deception monitoring the key design goal should be to ensure a believable environment. No suspicious or unexplainable processes should be present that could tip off an intruder that logging is ongoing. Fake audit logs are also a good way to create believability; if a honey pot is developed using an operating system with normal audit logging, then this should be enabled. A good adversary will likely turn it off. The idea is that hidden monitoring would have to be put in place underneath the normal logging—and this would be functionality that the adversary could not turn off.

Interfaces Between Humans and Computers

The gathering of forensic evidence during the analysis of intruder behavior in a honey pot often relies on detailed understanding of how systems, protocols, and services interact. Specifically, this type of communication can be performed in four different ways: *human-to-human*, *human-to-computer*, *computer-to-human*, and *computer-to-computer*. If we take the first term (human or computer) to mean the intruder and we take the second term to mean the honey pot manager, then we can make some logical distinctions.

First, it should be obvious that, in an automated attack such as a botnet, the real-time behavior of the attack system will not change based on some subjective observation of honey pot functionality. Certainly, the interpretation of the results of the botnet could easily affect the thinking of the botnet operator, but the real-time functionality is not going to be affected. As such, the most powerful cases in real-time forensic analysis of honey pot behavior will be the cases where human-to-human and human-to-computer interactions are being attempted by an intruder. Let's examine each in turn.

Real-time forensic analysis is not possible for every scenario, such as a botnet attack.

The most common human-to-human interaction in national infrastructure involves help desk or customer care support functions, and the corresponding attack approach involves social engineering of such activity. The current state of the art in dealing with this vulnerability is to train operators and customer care personnel to detect attempts at social engineering and to report them to the security team. Deception, however, introduces a more interesting option. If the likelihood is high that social engineering is being attempted, then an advanced approach to protection might involve deceiving the adversary into believing that they have succeeded. This can be accomplished quite easily by simply training operators to divert social engineering attempts to specially established help desks that are phony. The operators at these phony desks would reverse social engineer such attackers to get them to expose their identity or motivation (see Figure 2.11).

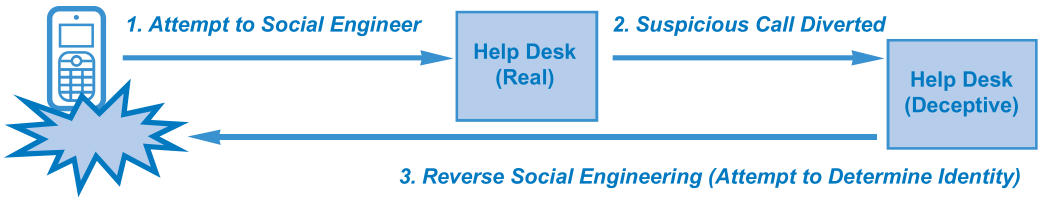


Figure 2.11 Deceptively exploiting the human-to-human interface.

An expert adversary may become aware of the security team observing the attempted intrusion.

The most common human-to-computer interaction occurs when an intruder is trying to gain unauthorized access through a series of live, interactive commands. The idea is that intruders should be led to believe that their activity is invoking services on the target system, as in the usual type of operating system hacking. A good example might involve an intruder repeatedly trying to execute some command or operation in a trap system. If the security team notices this intent and can act quickly enough, the desired command or operation could be deliberately led to execute. This is a tricky engagement, because an expert adversary might notice that the target configuration is changing, which obviously is not normal.

National Deception Program

One might hope that some sort of national deception program could be created based on a collection of traps strategically planted across national infrastructure components, tied together by some sort of deception analysis backbone. Such an approach is unlikely, because deception remains a poorly understood security approach, and infrastructure managers would be very hesitant to allow traps to be implanted in production systems. These traps, if they malfunction or do not work as advertised, could trick authorized users or impede normal operations.

Any realistic assessment of current security and information technology practice suggests that large-scale adoption of deception for national infrastructure protection would not be widely accepted today. As a result, programs of national deception would be better designed based on the following assumptions:

- *Selective infrastructure use*—One must assume that certain infrastructure components are likely to include deceptive traps but that others will not. At the time of this writing, many infrastructure teams are still grappling with basic computer security concepts; the idea that they would agree to install traps is not realistic. As such, any program of national deception must assume that not all components would utilize honey pots in the same manner.
- *Sharing of results and insights*—Programs of national deception can and should include a mechanism for the sharing of results and insights gained through operational use of traps and

honey pots. Certainly, insight obtained through forensic analysis of adversary behavior can be shared in a structured manner.

- *Reuse of tools and methods*—National deception programs could serve as means for making honey pot and trap software available for deployment. In some cases, deception tools and methods that work in one infrastructure area can be reused in another.

The most common criticism of deception in large-scale national security is that automated tools such as botnets are not affected by trap functionality. While it is true that botnets attack infrastructure in a blindly automated manner regardless of whether the target is real or fake, the possibility remains that trap functionality might have some positive impact. A good example might be national coordination of numerous bogus endpoints that might be ready and willing to accept botnet software. If these endpoints are designed properly, one could imagine them being deliberately designed to mess up the botnet communication, perhaps by targeting the controllers themselves. This approach is often referred to as a *tarpit*, and one might imagine this method being quite interesting for degrading the effectiveness of a botnet.

Finally, let's briefly look at how to improve the plans to shore up the defenses against cyber attacks. Amateur cyber attacks are expected to show more deception, and cyberwarfare attacks can be expected to show more too.

The Deception Planning Process Against Cyber Attacks

Cyber attacks are increasing in technical sophistication, as easier attacks are being blocked or foiled. Deception can be a useful force multiplier for mission plans in cyberspace just as in real battle spaces. Many new deceptions are expected, since very few of the possible ploys have been explored, and it will become very easy for deceptions to succeed: Defenses have come to a screeching halt, and defenders are becoming less aware of deceptions being practiced, so the pool of potential victims for many attacks is increasing.

Nevertheless, successful deception starts with a deception plan. A successful deception process is one in which the ends dictate the means. This is reinforced by the fact that deception plans are driven by the desired effect on the target. Deception planners are guided toward a successful deception planning process that requires command involvement and approval at each stage of the process. This process could be a 34-part step-by-step planning process for deception to increase the probability of success (see "An Agenda for Action in the Planning Process for Deception" below):

An Agenda for Action in the Planning Process for Deception

When completing the Planning Process for Deception checklist, the deception planner should adhere to the provisional list of actions to prepare for contingencies in the event that the deception fails. During the course of the deception, the planner also seeks feedback to ensure that the target is responding in the expected way. The order is not significant; however, these are the activities for which the research would want to provide a detailed description of procedures, review, and assessment for ease of use and admissibility. Current measures that must be adhered to, in order to plan for deception, include (check all tasks completed):

1. Identifying the strategic goal.
2. Deciding how the target should react.
3. Determining what the target should perceive.
4. Deciding what to hide and show.
5. Analyzing the pattern for hiding.
6. Analyzing the pattern for showing.
7. Designing the desired effect with the hidden method.
8. Selling the effect to those who are executing the deception.
9. Deciding the communications channels to transmit the deception.
10. Making sure that the target buys the effect and falls for the deception.
11. Pretending to be a naive victim to entrap deceptive cyber attackers.
12. Camouflaging key targets or make them look unimportant or disguise software as different software.
13. Doing something an unexpected way.
14. Inducing the cyber attacker to download a Trojan horse.
15. Secretly monitoring attacker activities.
16. Transfering Trojan horses back to cyber attacker.
17. Trying to frighten the cyber attacker with false messages from authorities (like “we know where you are, and a drone is coming to take you out,” etc. …).
18. Transfering the cyber attack to a safer machine like a honeypot.
19. Swamping the cyber attacker with messages or requests.
20. Associating false times with files.
21. Falsifying file-creation times.
22. Falsifying file-modification times.
23. Deliberately delaying processing commands.
24. Lying that you cannot do something or do something unrequested.
25. Lying that a suspicious command succeeded.
26. Lying about reasons for asking for an additional password.
27. Planting disinformation, redefining executables, and giving false system data.
28. “Emulating” hardware of a machine in software for increased safety.
29. Sending data too large or requests too hard back to the cyber attacker.

30. Systematically misunderstanding cyber attacker commands, as by losing characters.
31. Being a decoy site for the real site.
32. Asking questions that include a few cyber attacker–locating ones.
33. Giving false excuses why you cannot execute cyber attacker commands.
34. Pretending to be an inept defender, or have easy-to-subvert software.

Summary

Deception occurs in cyberspace. An analysis of how deception is used in cyber attacks can help in understanding them, with the goal of developing effective defenses for future cyber attacks against the critical national infrastructure. The deception methods described in this chapter are not difficult to use. While there have not been confirmed instances of cyberwar using deception, cyberwarfare specialists are developing cyberweapons using these methods. However, a wide variety of deception methods can be used to ensure that particular cyber-attack deceptions against a particular target are totally ineffective:

- Buffer overflows can be done by sending insincere large inputs to programs.
- To achieve surprise, cyber attacks can involve rarely used software, ports, or network sites.
- Cyber attacks can have surprising targets such as little-used software features.
- Cyber attacks can occur at surprising times.
- Cyber attacks can occur from surprising sites.
- To maximize concealment, cyber attacks can be done very slowly, as by sending one command a day to a victim computer.
- Cyber attacks can modify file or audit records in time and details to make cyber attackers appear to have been doing something different at a different time.
- Cyber attacks can claim abilities that they do not possess for purposes of extortion, such as the ability to disable a computer system.

Nonetheless, the diversity of deceptions should increase in the future as the continued development of automated tools will permit attackers to try many methods at once. But, diversity in defenses against deceptions should also increase. Deception will be increasingly common in asymmetric cyberwar, as it is in asymmetric conventional warfare, for tactics and strategies by the weaker participant.

Finally, let's move on to the real interactive part of this chapter: review questions/exercises, hands-on projects, case projects, and optional team case project. The answers and/or solutions by chapter can be found online at http://www.elsevierdirect.com/companion.jsp?ISBN=9780123918550.

CHAPTER REVIEW QUESTIONS/EXERCISES

True/False

1. True or False? The use of deception in computing involves deliberately misleading an adversary by creating a system component that looks fake but is in fact a trap.
2. True or False? Deception can be used to divert scanning attempts by creating false entry points with planted vulnerabilities.
3. True or False? A secondary goal of deception is to observe the adversary in action.
4. True or False? The deliberate insertion of closed service ports on an Internet-facing server is the most straightforward of all deceptive computing practices.
5. True or False? The discovery stage corresponds to the adversary finding and accepting the security bait embedded in the trap.

Multiple Choice

1. The reason why deception works is that it helps accomplish any or all of the following four security objectives:
   A. Attention, energy, uncertainty, and analysis
   B. Attention, vulnerability, uncertainty, and analysis
   C. Attention, energy, honey pot, and analysis
   D. Attention, energy, uncertainty, and traceability
   E. Implementation, energy, uncertainty, and analysis
2. If the deception is properly managed, then the adversary should be led down a controlled process path with four distinct attack stages, except which one:
   A. Scanning
   B. Functionality
   C. Exploitation
   D. Discovery
   E. Exposing
3. Honey pots should include sufficient monitoring to expose which of the following three:
   A. Adversary technique
   B. Depth
   C. Intent

D. Identity
E. Diversity

4. The deceptive design goal during scanning is to make available an interface with which three distinct components:
A. Authorized services
B. Real vulnerabilities
C. Unrealistic expectations
D. System misconfigurations
E. Bogus vulnerabilities

5. Servers will present adversaries of the national infrastructure with which three different views of open service ports:
A. There could be some uncertainty on the part of the adversary about which open ports are deliberate and which are inadvertent.
B. Valid open ports one might expect, such as HTTP, DNS, and SMTP.
C. If the back-end deceptive software connected to deliberately open ports shares resources with valid assets, then the potential exists for negative side effects.
D. Open ports that are inadvertently left open and might correspond to exploitable software.
E. Open ports that are deliberately inserted and connected to bogus assets in a honey pot.

Exercise

Problem

A diversified Fortune 500 corporation that provides products and services to domestic and foreign governments and commercial customers suspected that a deceptive intruder was in their network; however, they knew neither the extent of the compromise, nor what (if any) data had been breached. The persistent deceptive intruders used tools and techniques that left trace evidence on each computer system they compromised. These host-based indicators of compromise are present every time the intruders attack a network. The corporation (client) called a team of advanced persistent threat (APT) experts to validate their concerns, scope the intrusion, and provide a remediation strategy. APTs are used to identify, scope, and remediate the APT in the government and defense industrial base. The APT consists of skilled and sophisticated deceptive hackers who deploy a complex arsenal of deception malware against specific targets in the Defense Industrial Base (DIB), financial, manufacturing, and research industries. Please explain how the APT went about resolving the problem.

Hands-On Projects

Project

The Defense Information Systems Agency (DISA) within the U.S. Department of Defense (DoD) has parallel missions to evaluate new network defense technologies, policies, and tactics and to train DoD personnel to repel deception attacks upon critical cyber national infrastructures. DISA deployed a fully operational evolved cyber range to measure the resiliency (deception, performance, security, and stability) of its network and data center infrastructures, conduct advanced research and development, and train its cyber warriors. Nevertheless, DISA needed the full functionality of a cyber range to carry out its missions. Just as traditional soldiers need a firing range to hone their skills using the latest weaponry, cyber warriors need a similar environment in the virtual world to train for deception in cyber attacks. Yet, the agency could not wait several more years for the launch of the DoD's National Cyber Range, nor could it spend the millions of dollars required for traditional custom-built cyber ranges. A new, evolved model of cyber range was required, one that could be set up in hours with minimal infrastructure, then customized for cyberwarfare scenarios within minutes. So, how would DISA's cyber security team go about creating the cyber range model?

Case Projects

Problem

Let's look at a real-world scenario of how one of the world's largest banks was challenged to harden network and data center critical infrastructure security deception measures without degrading the high performance required in the financial services industry. The bank's network security team (NST) was charged with institutionalizing a network security certification process to measure the resiliency (performance, security, and stability) of every element of the network before and after deployment. The goal for the team was to right size the critical infrastructure for each line of business without introducing risk, ensuring that they did not over- or underinvest in the network infrastructure. The team used a standardized and repeatable program to certify that devices are able to:

- Protect sensitive customer data from external deception attacks and insider threats.
- Ensure cyber secure, rapid financial transactions.
- Reduce the risk of legal liabilities associated with noncompliance.

Explain how the bank's network security team should handle this situation.

Optional Team Case Project

Problem

Yahoo! is focused on delivering fast and reliable commerce, communications, and social networking services to millions of users around the world. With one of the world's largest network and cloud infrastructures, Yahoo! faces unique challenges as it fulfills its vision to be the center of people's online lives by delivering personally relevant, meaningful Internet experiences. Yahoo!'s traffic volume and application complexity has grown rapidly over the past decade, driving the company to build out a massive network and application infrastructure to support more and more load. The company continues to invest in high-capacity servers, load balancers, routers, and switches, plus massive firewalls. Previously, however, Yahoo!'s security team had no way to stress this enormous critical infrastructure and measure its resiliency to ensure performance, stability, and cyber security. Yahoo! needed a solution to validate the performance, functionality, and capacity of its systems under a wide mix of real-world traffic, including video, instant messaging, and web applications, as well as live cyber security attacks and load from millions of users. Please identify how Yahoo!'s security team stressed their enormous critical infrastructure and measured its resiliency to ensure performance, stability, and cyber security.

This page intentionally left blank