Improving Security using Deception

Mohammed H. Almeshekah, Eugene H. Spafford and Mikhail J. Atallah

November 11, 2013

1 Abstract

As the convergence between our physical and digital worlds continues at a rapid pace, much of our information is becoming available online. In this paper we develop a novel taxonomy of methods and techniques that can be used to protect digital information. We discuss how information has been protected and show how we can structure our methods to achieve better results. We explore complex relationships among protection techniques ranging from denial and isolation, to degradation and obfuscation, through negative information and deception, ending with adversary attribution and counter-operations. We present analysis of these relationships and discuss how can they be applied at different scales within organizations. We also identify some of the areas that are worth further investigation. We map these protection techniques against the cyber kill-chain model and discuss some findings.

Moreover, we identify the use of deceptive information as a useful protection method that can significantly enhance the security of systems. We posit how the well-known Kerckhoffs's principle has been misinterpreted to drive the security community away from deception-based mechanisms. We examine advantages these techniques can have when protecting our information in addition to traditional methods of hiding and hardening. We show that by intelligently introducing deceptive information in information systems, we not only lead attackers astray, but also give organizations the ability to detect leakage; create doubt and uncertainty in any leaked data; add risk at the adversaries' side to using the leaked information; and significantly enhance our abilities to attribute adversaries. We discuss how to overcome some of the challenges that hinder the adoption of deception-based techniques and present some recent work, our own contribution, and some promising directions for future research.

2 Introduction

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

In this paper we present a novel taxonomy of protection mechanisms and controls that can be deployed to protect our information. The main motivation of this approach is to show the range of security controls organization can deploy, what the objectives are of each group and how these mechanisms can interact with each other to achieve better overall security. We also discuss some of the interesting relationships among these groups and how this can be exploited to link isolated controls. Furthermore, we argue that deceit and negative information can play a crucial rule in enhancing our defensive and counter-operational capabilities. We take the traditional schemes of honeypots and honeynets to a more interesting level and discuss some practical examples of how can they be applied. We examine the main challenges deception-based mechanisms have faced and discuss how they can be overcome, presenting some novel use cases.

To fit all the pieces together, for a holistic and practical approach to security, we map our taxonomy to the cyber kill-chain model introduced by Lockheed Martin in [2]. We develop and expand some of the stages of the cyber kill-chain model and show that we can have more effective security controls at each stage examining how individual controls can interact.

3 How we protect our Information

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

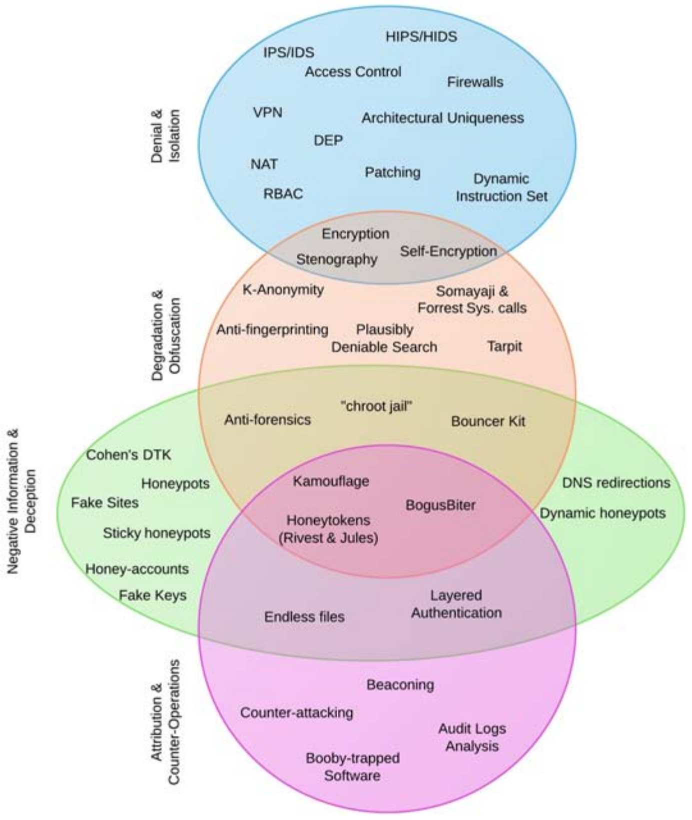

Figure 1: Taxonomy of Information Protection Mechanisms

The rationale behind having these intersecting categories is that a single layer of security is not adequate to protect organizations so multi-level security controls are needed [3]. In this model, the first goal is to deny unauthorized access and isolate our information systems from untrusted agents. However, if adversaries succeed in penetrating these security controls, we should have degradation and obfuscation mechanisms in place that slow the lateral movement of attackers in penetrating our internal systems. At the same time, this makes the extraction of information from penetrated systems more challenging. Even if we slow the attackers down and obfuscate our information, advanced adversaries may explore our systems undetected. This motivates the need for a third level of security controls that involves using means of deceit and negative information. These techniques are designed to lead attackers astray and augment our systems with decoys to detect stealthy adversaries. Furthermore, this deceitful information will waste the time of the attackers and/or add risk during their infiltration. We will provide more discussion and give some example on how to achieve these goals in the third subsection below. The final group of mechanisms in our taxonomy is designed to attribute the attackers and give us the ability to have counter-operations. Booby-trapped software is one example of counter-operations that can be employed.

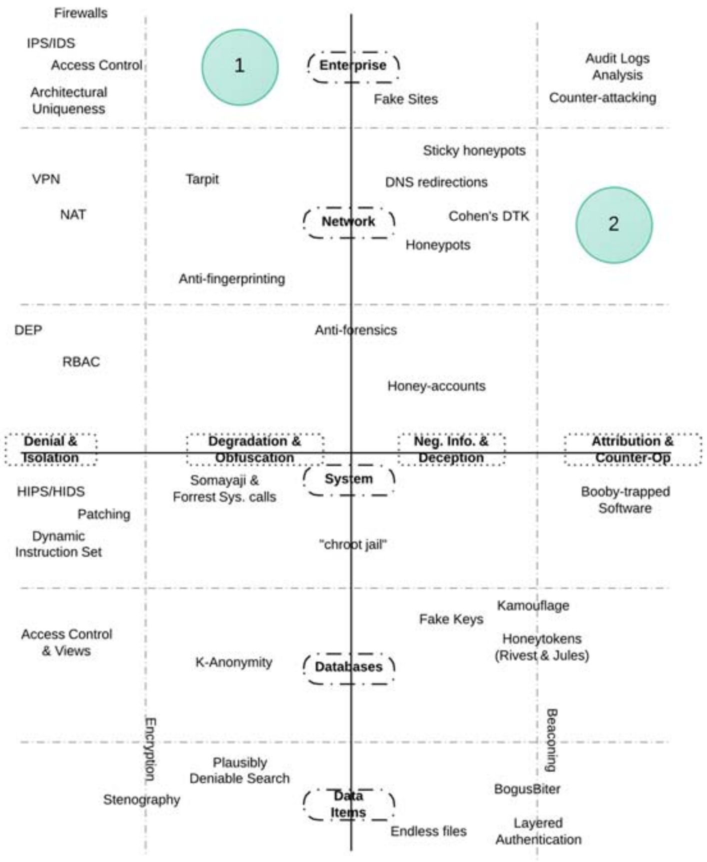

Figure 2: Plotting the Taxonomy Over Multiple Scales

In Figure 2, we plot these four categories to five levels of scales within organizations:

- Data items; this includes files and objects.
- Databases; which are collections of data items creating bigger, coherent objects.
- Systems; this refers to individual systems within our organization. For example, end-points and servers fall into this category.
- Networks; which are a network of systems connected together with communications equipment such as switches and routers.
- Enterprise level; which refers to the highest level of abstraction in the digital realm. This abstraction also includes issues that deal with users and human actors.

Securing a system is an economic activity and organizations have to strike the right balance between cost and benefits. Our taxonomy provides a holistic overview of security controls, with an understanding of the goals of each group and how can they interact with each other. This empowers decision makers on what and which security controls they should deploy. In the four subsections that follow we discuss each one of the four categories, illustrating their goals and providing some practical examples.

3.1 Denial and isolation

The first, and most common, mechanism to protect information systems is to deny all access, execution and manipulation of our systems and data unless explicitly allowed. This gives us the ability to create a boundary around our systems isolating them from the outside. This group covers a wide variety of security controls that can be sub-grouped into three major categories; controls installed around the perimeter, such as firewalls; within our internal systems, such as Access Control; and at the end-points, such as anti-virus and intrusion prevention. We give several examples of mechanisms in the upper-most oval in Figure 1.

Security controls in this category are designed to achieve two main goals:

- Prevent unauthorized access to information stored in our systems.
- Hide the existence and/or the nature of our systems and/or the data stored in them.

Such mechanisms can be applied at all scales within our information systems as presented in Figure 2. At the enterprise level, we employ security controls such as firewalls and access control systems. More advanced mechanisms such as having unique system architecture and advanced intrusion prevention systems can be used. At the network level technologies such as network address translation (NAT) and virtual private networks are used to isolate and hide parts of our systems denying unauthorized access to them. Denial mechanisms can also be applied at the systems level. Tools such as data execution prevention (DEP) [4] and patching security vulnerabilities are commonly used. More sophisticated mechanisms such as dynamic instruction sets can be used to obfuscate the instruction set a computer can execute and, therefore, prevent any unauthorized programs from running [5]. At the database and data item granularity level mechanisms such as encryption and steganography can be used.

3.2 Degradation and obfuscation

When adversaries overcome the first line of defense, we have three general classes of objectives, detect them, slow them down, and disguise and/or hide data. Many security mechanisms are used to address these issues. Security controls in this group can be grouped into four categories:

- Slow down the attackers.
- Prevent and significantly reduce the probability that an adversary can recover sensitive data.

- Obfuscate the value/nature of our systems and/or the data stored in them.
- Create noise around the valuable information to reduce its utility.

At the data item level, mechanisms such as k-anonymity [6] and plausibly deniable search [7] have been used to degrade the information obtained – directly and indirectly – from our data. At the systems' level, slowing down the response of system calls when detecting anomalies has proposed to degrade the adversaries' infiltration speed [8]. At the network level, tarpits are used to throttle the spread of malware and spam within the organization [9]. We note that there is a shortage of these techniques to employ at the top level in our target hierarchy (the enterprise level).

3.3 Negative information and deception

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

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

Deception based techniques are increasingly gaining interest within the information security community [10] [11]. Security controls in this category are designed to achieve three main goals:

- Lead the attackers astray.
- Add decoys to our system to detect data leakage and intrusions.
- Add risk and doubt to the data obtained by the adversary.

When attackers infiltrate the system and successfully overcome traditional detection and degradation mechanisms we would like to have the ability to not only obfuscate our data, but also lead the attackers astray by deceiving them and drawing their attention to other data that are false or intestinally misleading.

Furthermore, exhausting the attacker and causing frustration is also a successful defensive outcome. This can be achieved by planting fake keys and/or using schemes such as endless files [12]. These files look small on the organization servers but when downloaded to be exfiltrated will exhaust the adversaries' bandwidth and raise some alarms. Moreover, with carefully designed deceiving information we can even cause damage at the adversaries' servers. A traditional, successful, deception technique can be learned from the well-known story of Farewell Dossier during the cold war where the CIA provided modified items to a Soviet spy ring. When the Soviets used these designs thinking they are legitimate, it resulted in a major disaster affecting a trans-Siberian pipeline.

When we inject false information we cause some confusion for the adversaries even if they have already obtained some sensitive information; the injection of negative information can degrade and/or devalue the correct information obtained by adversaries. Heckman and his team, from Lockheed Martin, conducted an experiment between a red and a blue team using some deception techniques, where they found some interesting results [11]. Even after the red team successfully attacked and infiltrate the blue system and obtained sensitive information, the blue team injected some false information in their system that led the red team to devalue the information they had obtained, believing that the new values were correct.

Another relationship can be observed between the last group of protection techniques, namely attribution, and deception techniques. Deception-based mechanisms are an effective way to lure attackers to expose themselves and their objectives when we detect them accessing things and conducting activities unusual for normal. Other tools, such as anomaly-based IDS, have similar goals, goal but the advantage deception-based tools have is that there is a clear line between normal user activities and abnormal ones. This is because legitimate users are clearly not supposed to access this information. This difference significantly enhances the effectiveness of deception-based security controls and reduces the number of false-positives, as well as the size of the system's log file.

As can be seen in Figure 2, deception mechanisms can be applied at all levels of scale within our systems. We defer the discussion of these to section 5.2 where we present some new ideas on how to deploy multiple cooperative deception mechanisms at all scales instead of having a single component that works in isolation.

3.4 Attribution and counter-operations

Sun Tzu, the great Chinese military strategist, once said "if you know your enemies and know yourself, you will not be imperiled in a hundred battles; if you do not know your enemies but do know yourself, you will win one and lose one". This brilliantly summarizes the current security state of many organizations around the world. We clearly need to know the attackers, attribute them and understand their objectives. Security controls in this last category are designed to achieve three main goals:

- Attributing the adversaries.
- Cause damage to any attackers.
- Increase overall risk in attacking our systems.

One of the traditional ways of learning about adversaries is analyzing the logs generated by our systems. However, one of the main challenges that has been hindering the adaptation of mechanisms intended for attributing adversaries is mixing those mechanisms with counter-attacking and “hacking back” mechanisms. We argue that attribution can be achieved using a wide variety of mechanisms without having to address the ethical and political issues surrounding counter-attacking.

We argue that intelligently planting deceiving information within our information systems can help us both in attributing some adversaries and detecting leakage. Two recent schemes have been proposed to protect information at the data item scale and the database scale. BogusBiter scheme has been proposed such that servers can detect when a user has fallen for phishing by submitting fake, i.e. deceiving credentials, when the user agent suspects a web page as phishing [13]. Rivest and Jules also proposed augmenting the password database in Unix with negative information such that the cracked password files can be detected [14]. Steganographic watermarking data can also serve as a means of detecting leakage and possibly providing attribution of sources.

4 Fitting all the pieces together

Employing techniques from all four of our groups provide a more effective approach than only using one or two. Also, we note that security is an economic activity and distributing budget at multiple layers may provide a better return on investment than more focused spending.

4.1 Cyber Kill-Chain Model

The cyber kill-chain introduced by Lockheed Martin researchers advocates an intelligence-driven security model [2]. The main premise behind this model is that for attackers to be successful they need to go through all these steps in the chain in sequence. Breaking the chain at any step will break the attack and the earlier that we break it the better we prevent the attackers from attacking our systems.

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

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

	Denial & Isolation	Degradation & Obfuscation	Deception & Negative Information	Attribution & Counter Operations
Reconnaissance	Firewalls, Architectural Uniqueness, NAT	Anti-fingerprinting	Artificial ports, Fake Sites	Audit Logs Analysis
Weaponization & Delivery	In-line Filters, IPS, Tarpit, IDS		Create artificial bouncing back, Sticky Honeypots	
Exploitation & Installation	Dynamic Instruction Set, Somayaji & Forrest sys. calls, HIPS, Patching, DEP, “chroot jail”, HIDS		Create artificial exploitation response	
Command & Control (operation)			Honeypot	
Lateral Movement & Persistence	VPN, Access Control, RBAC	Encryption, Self-Encryption	HoneyAccounts, HoneyFiles	
Staging & Exfiltration		Stenography	Honeytokens, Endless files, Fake Keys	Beaconing, Counter-Attacking, Booby Trapped Software

Table 1: Mapping Security Mechanisms to the Kill-Chain Model

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

To show how all the protection categories discussed above can fit together in protecting organizations we map them against the cyber kill-chain model summarized in Table 4.1.

The deployment of the cyber kill-chain was seen as fruitful for Lockheed when they were able to detect an intruder who successfully logged into their system using the SecurID attack [15]. We adopt this model with slight modification to better reflect our additions.

5 The Use of Deception

Over history, deception has evolved to find its natural place in our societies and eventually our technical systems. In addition, deception and decoy-based mechanisms have been used in security for more than two decades in techniques such as honeypots and honeytokens. However, incorporating security measures formally based on deception is still not done. One of the reasons is that deception always involves two basic steps, as identified by Bell and Whaley, – hiding the real and showing the false [2]. This, at first glance, contradicts the widely believed misinterpretation of Kerckhoff's principle: "no security through obscurity." A more correct English translation of Kerckhoff's principle is "The system must not require secrecy and can be stolen by the enemy without causing trouble" [16].

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

In any system design there are three levels of transparency, openness and deception incorporation:

- **Full Honesty.** In such systems, the processes will always respond to any input with full "honesty". In other words, the system's response is always "trusted" and fully represents the internal state of the machine. For example, when the user asks for a particular network port to listen to the system response with real port or denies the request.

- **Naive Deception.** In such systems, the processes attempt to deceive the interacting user sending by crafting an artificial response. However, if the user, in this case the attacker, knows the deceptive behavior, e.g. by analyzing the previous deceptive response used by the system, the deception act becomes useless and will only alert the user that the system is trying to deceive her. For example, the system can designate a specific port that is used to try to deceive the attacker when they ask for a port while the do not carry the appropriate permissions.

- **Full Deception.** In this case, the systems "deceptive behavior" is indistinguishable from the normal behavior even if the user have previously interacted with the system. For example, the system responds to unauthorized port listening request just like a normal allowed request. However, an extra action is take to monitor the port, alert the system administrators and/or sandbox the listening process to limit the damage if the process downloads malicious content.

Deception-based techniques provide a significant advantage over traditional security controls. Currently, most of our defensive measures are playing **"whack-a-mole"** game with attackers. Whenever an attack surface we hit it hard with our prevention mechanisms and all what the attacker needs to do is to retreat and try different avenues. This is partially driven by the unquestioned assumption that **"hacking-back"** is unethical. With such behavior the attackers continuously learns about our abilities and behavior with their continuous probing and we obtain nothing, other than a panic in the security team, about their attempts and objectives. In fact, in many cases we cannot correlate multiple attacks that originate from the same entity. However, intelligently using deception techniques we can connect the dots and start to identify patterns that help us in enhancing our systems and defend against the next attack. Attackers attitude is that the completely trust the computer response and consider them "honest" response. Exploiting such trust can give us an edge and make us a step ahead of these attackers. Finally, with the increasing threat of advanced persistent threats we can correlate events and try to attribute attackers and learn about their motives.

5.1 Quick Overview of Deception in Computing

The use of deception in information systems dates back to the late 1980s. The earliest documented work in this area we have found is of Clifford in his book "The Cuckoo's Egg" [17] and Spafford in his own lab [12]. Some of the early use of Tripwire also involves deception. Since then a number of concepts on using deception for security has emerged.

One of the early pieces in using deception techniques to attribute and study attackers was the work of Cheswick in his infamous paper "An evening with Berferd" [18]. He described how he watched and interacted with an attacker in real time by manually simulating system responses. Later in 1997, Fred Cohen published the first deception toolkit (DTK)[1]. After that, many solutions have followed in the commercial and open source communities.

In the early 2000s, Spitzner published the first book on one of the widely used deception tools (honeypots) in which he provides a detailed study of these tools [19]. Since then honeypots have been used in multiple security applications such as detecting and stopping spam[2] and analyzing malware [20]. In addition, honeypots have been used to secure databases [21]. Honeypots are starting to find their way into mobile environment [22] and some interesting patterns and measurements are reported [23].

The prefix "honey-*" has been used to refer to a wide range of techniques that incorporate the act of deceit in them. The fundamental idea behind the use of the word "honey" is for these techniques to work they need to entice attackers to interact with them, in other word, fall for the bait, i.e. "honey". The term honeytokens have been proposed by Spitzner [24] to refer to honeypots but at a smaller granularity. Yuill et al used the term *honeyfiles* to refer to files that have enticing names distributed in the system that act as a becoming mechanism when they are accesses [25]. HoneyGen was also used to refer to tools that are used to generate honeytokens [26].

Honeypots in the literature come in two different flavors; server honeypot and client honeypot. The server honeypot is a computer system that contains no valuable information and is designed to look vulnerable for the goal of enticing attaked to access them. Client honeypots are more active in the sense that they are vulnerable user agents that troll many servers to see if they get compromised by some servers who are owned by people attacking other clients [27]. Honeypots based techniques have been a valuable tool that has been used for the detection, prevention and response. Nevertheless, such techniques suffer from the following major limitations:

- As the prefix **honey-*** indicates, for such techniques to become useful, the adversary needs to interact them. Attackers and malware are becoming increasingly sophisticated and their ability to avoid honeypots are significantly enhancing.

- Assuming we manage to lure the attacker into our honeypot, we need to be able to *contentiously* deceive them that they are in the real system. Chen et al studies such issue and we can see that a number of malware not just detect that they are inside a honeypot, but also change their behavior [28]. Attackers can be in a position where they have the ability to conduct counter-intelligence activities by behaving in a manner that is different than how would they do in real environment, i.e. counter deceive us.

- In honeypots, attackers are supposed to interact with it such that we can learn more about their objectives and attribute them. However, with high-interaction honeypot there is a risk that attackers

[1]http://www.all.net/dtk/
[2]projecthoneypot.org

might even exploit this system and use it as a vector to the other parts of our internal systems. Of course, with correct separation and DMZs we can alleviate the damage, but many organizations consider the risk more than the gained value and avoid using such tools.

5.2 Deception in-Depth

One of the first formal definitions of honeypots, by Spitzner, defines honeypot as "an information system resource whose value lies in unauthorized or illicit use of that resource" [19]. Deception techniques based on the honeypot model suffer from three main disadvantages:

1. Usually such techniques are stand-alone system components by themselves.
2. They are only useful if the attacker interacts with them. This brings two main disadvantages; (i) adversaries who are after a specific target will not be motivated to interact with other enticing bait. (ii) Honeypots are largely useless against insiders who are aware of the existence of the honeypots.
3. They intrinsically introduce risk when deploying them. There is a continuous worry that honeypots can be used as a vector to attack the system.

The majority of currently-deployed deception-based security mechanisms work in isolation and are designed to achieve a specific goal. To the best of our knowledge, the first published work of using multiple interacting deception techniques at different levels in a system is the work of Wang in [29]. Their approach is what we refer to as Deception in-depth.

Many deception techniques, such as honeypots, work in isolation and independently of other parts of current information systems. This design decision has been partly driven by the security risks associated with honeypots. We argue that intelligently augmenting our systems with interacting deception-based techniques can significantly enhance our security and gives us the ability to achieve deception in depth. If we examine Table 1, we can see that we can apply deception at every stage of the cyber kill-chain, allowing us to break the chain and possible attribute attackers. At the reconnaissance stage we can lure adversaries by creating a site and have honey-activities that mimic a real-world organization. As an example, an organization can subscribe with a number of cloud service providers and have honey activities in place while monitoring any activities that signal external interest. Another example is to address the problem of spear-phishing by creating a number of fake persons and disseminating their information into the Internet while at the same monitoring their contact details to detect any probing activities; some commercial security firms currently do this.

At the exploitation and delivery stage, the example we discuss below in changing the way we patch vulnerabilities can show how deception can be achieved. For internal reconnaissance and lateral movement we can have bait files and tripwires [30] around our systems. Furthermore, having honeydocs that can beacon back when they are exfiltrated is another deception technique that can be applied.

One of the challenges security offices continuously face is the zero-day vulnerabilities and corresponding patch-exploit cycles. A black market exists for vulnerabilities that have not been patched which are commonly referred to as zero-days [31]. The intrinsic value of these is twofold; first there is a large probability that these vulnerabilities are still a weakness in the targeted systems; second, and more importantly, there is a negligible risk of using them to attack systems from the attacker perspective. We argue that if we augment deception techniques in our patching cycles we can address these two points and create doubts and risks in these markets rendering the value of zero-day vulnerabilities lower.

The main phases of a life cycle of zero-day vulnerabilities are; discovery of a vulnerability, constructing an attack; then after some time a patch is created and released and finally the patch is applied throughout organizations. We can augment deception techniques with the patch to perform two tasks; fix the vulnerability and, at the same time, when the software gets an input that matches the exploit it should respond as if the vulnerability still exists while also alerting the systems administrator; this approach was a feature in the AAFIDS system [32]. We can go a step further and mimic a legitimate response diverting the adversary into a sandbox and attributing them by learning their goals and behavior. This can be also combined with honeypots to lure attackers by further deceiving them that the vulnerability already exists and learning more about their motives and objectives.

With such deployed mechanism we can achieve three main goals:

- When adversaries obtain and exploit a zero-day vulnerability, this will come with its own risk. The adversaries do not know whether the response they will get from the targeted system is a result of an unpatched vulnerability or a decoy to lure them further into exposing themselves.

- The market for zero-days will be affected as the previously perceived value of obtaining such exploits is diminished because of the risk they come with. Even further, some artificial exploits can purposely be fed into these market to improve our ability to attribute the customers within these markets.

- Another, less obvious, value of such an approach is with attacks that are intended to aggregate data from attacked systems. If the newly created patches inject false data, we achieve the goal of poisoning the attackers' database.

6 Conclusion

We have outlined a new classification scheme for deception techniques in cyber security. We have shown how some of these techniques have been known and used for many years, but that the field is under-developed. We believe this is because of a misapprehension of the nature of deception in security.

We have explained how systems can be augmented to use deception and false information to protect them and their data, to degrade attacks, to expose attackers, to enhance attribution, and possibly to be used to damage or degrade attacker capabilities. There are many more examples than we have shown in this brief paper, and the field could undoubtedly benefit from further study.

More directly, good security is really based on trust, and we believe that it is desirable to ensure that attackers have less trust in a system than its legitimate users. We have outlined a framework in which this goal may be pursued.

References

[1] Verizon, "Threats on the horizon - the rise of the advanced persistent threat."

[2] E. M. Hutchins, M. J. Cloppert, and R. M. Amin, "Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains," *Leading Issues in Information Warfare & Security Research*, vol. 1, p. 80, 2011.

[3] M. Sourour, B. Adel, and A. Tarek, "Ensuring security in depth based on heterogeneous network security technologies," *International Journal of Information Security*, vol. 8, no. 4, pp. 233–246, 2009.

[4] R. Riley, X. Jiang, and D. Xu, "An architectural approach to preventing code injection attacks," *Dependable and Secure Computing, IEEE Transactions on*, vol. 7, no. 4, pp. 351–365, 2010.

[5] M. J. Wirthlin and B. L. Hutchings, "A dynamic instruction set computer," in *FPGAs for Custom Computing Machines, 1995. Proceedings. IEEE Symposium on*, pp. 99–107, IEEE, 1995.

[6] L. Sweeney, "k-anonymity: A model for protecting privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 05, pp. 557–570, 2002.

[7] M. Murugesan and C. Clifton, "Providing privacy through plausibly deniable search.," in *SDM*, pp. 768–779, 2009.

[8] S. A. Hofmeyr, S. Forrest, and A. Somayaji, "Intrusion detection using sequences of system calls," *Journal of computer security*, vol. 6, no. 3, pp. 151–180, 1998.

[9] T. Eggendorfer, “Combining the smtp tar pit simulator with white listing.,” in *Security and Management*, pp. 333–339, 2008.

[10] A. Greenberg, “A different approach to foiling hackers? let them in, then lie to them.,” April 2013.

[11] K. Heckman, “Active cyber network defense with denial and deception.” Video, March 2013.

[12] E. Spafford, “More than passive defense.”

[13] C. Yue and H. Wang, “Bogusbiter: A transparent protection against phishing attacks,” *ACM Transactions on Internet Technology (TOIT)*, vol. 10, no. 2, p. 6, 2010.

[14] A. Juels and R. Rivest, “Honeywords: Making password-cracking detectable,” *Unpublished draft*.

[15] K. J. Higgins, “How lockheed martin’s ’kill chain’ stopped securid attack.”

[16] F. Petitcolas, “la cryptographie militaire.”

[17] C. P. Stoll, “The cuckoo’s egg: Tracing a spy through the maze of computer espionage,” 1989.

[18] B. Cheswick, “An evening with berferd in which a cracker is lured, endured, and studied,” in *Proc. Winter USENIX Conference, San Francisco*, 1992.

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

[20] D. Dagon, X. Qin, G. Gu, W. Lee, J. Grizzard, J. Levine, and H. Owen, “Honeystat: Local worm detection using honeypots,” in *Recent Advances in Intrusion Detection*, pp. 39–58, Springer, 2004.

[21] C. Fiedler, “secure your database by building honeypot architecture using a sql database firewall.”

[22] C. Mulliner, S. Liebergeld, and M. Lange, “Poster: Honeydroid-creating a smartphone honeypot,” in *IEEE Symposium on Security and Privacy*, 2011.

[23] M. Wählisch, A. Vorbach, C. Keil, J. Schönfelder, T. C. Schmidt, and J. H. Schiller, “Design, implementation, and operation of a mobile honeypot,” *arXiv preprint arXiv:1301.7257*, 2013.

[24] L. Spitzner, “Honeytokens: The other honeypot,” 2003.

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

[26] M. Bercovitch, M. Renford, L. Hasson, A. Shabtai, L. Rokach, and Y. Elovici, "Honeygen: An automated honeytokens generator," in *Intelligence and Security Informatics (ISI), 2011 IEEE International Conference on*, pp. 131–136, IEEE, 2011.

[27] C. Seifert, I. Welch, P. Komisarczuk, *et al.*, "Honeyc-the low-interaction client honeypot," *Proceedings of the 2007 NZCSRCS, Waikato University, Hamilton, New Zealand*, 2007.

[28] X. Chen, J. Andersen, Z. M. Mao, M. Bailey, and J. Nazario, "Towards an understanding of anti-virtualization and anti-debugging behavior in modern malware," in *Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on*, pp. 177–186, IEEE, 2008.

[29] W. Wang, J. Bickford, I. Murynets, R. Subbaraman, A. G. Forte, and G. Singaraju, "Detecting targeted attacks by multilayer deception," *Journal of Cyber Security and Mobility*, vol. 2, pp. 175–199.

[30] G. H. Kim and E. H. Spafford, "The design and implementation of tripwire: A file system integrity checker," in *Proceedings of the 2nd ACM Conference on Computer and Communications Security*, pp. 18–29, ACM, 1994.

[31] A. Greenberg, "Shopping for zero-days: A price list for hackers' secret software exploits."

[32] J. S. Balasubramaniyan, J. O. Garcia-Fernandez, D. Isacoff, E. Spafford, and D. Zamboni, "An architecture for intrusion detection using autonomous agents," in *Computer Security Applications Conference, 1998. Proceedings. 14th Annual*, pp. 13–24, IEEE, 1998.