A Deception Framework for Survivability Against Next Generation Cyber Attacks

Ruchika Mehresh[1], **and Shambhu Upadhyaya**[2]
[1]Department of Computer Science and Engineering, University at Buffalo, Buffalo, NY 14260, USA
[2]Department of Computer Science and Engineering, University at Buffalo, Buffalo, NY 14260, USA

Abstract— *Over the years, malicious entities in cyber-space have grown smarter and resourceful. For defenders to stay abreast of the increasingly sophisticated attacks, the need is to understand these attacks. In this paper, we study the current trends in security attacks and present a threat model that encapsulates their sophistication.*

Survivability is difficult to achieve because of its contradictory requirements. It requires that a critical system survives all attacks (including zero-day attacks), while still conserving the timeliness property of its mission. We recognize deception as an important tool to resolve this conflict. The proposed deception-based framework predicts an attacker's intent in order to design a stronger and more effective recovery; hence strengthening system survivability. Each design choice is supported by evidence and a detailed review of existing literature. Finally, we discuss the challenges in implementing such a framework and the directions that can be taken to overcome them.

Keywords: Deception, mission critical systems, recovery, security, survivability

1. Introduction

This is the era of cyber-warfare and it is no longer limited to military domain. Knapp and Boulton [12] have reviewed information warfare literature from 1990 to mid-2005 and made a strong case for how cyber warfare has extended to other domains outside military. Baskerville [3] has discussed how the asymmetric warfare theory applies to information warfare and how it has expanded to the electronic business domain. According to the asymmetric warfare theory, attackers have the advantage of time and stealth over defenders. Thus, in order to counter this imbalance, defense needs to be "agile and adaptive."

Owing to this increasing hostility, critical systems in cyber space need to be protected. This need for protection extends beyond the routine fault tolerance and security into the domain of survivability. Ellison et al. [8] describe survivability as "the capability of a system to fulfill its mission in a timely manner in the presence of attacks, failures and accidents." Survivability focuses on continuity of a mission (set of essential services) without relying on the guarantee that precautionary measures will always succeed. It concentrates on the impact of an event rather than its cause. There are four basic layers of protection in a survivable system: Prevention or resistance against faults/attacks; Detection of faults/attacks; Full recovery of the essential services (mission) after the fault/attack and; Adaptation or evolution to reduce the possibility or effectiveness of future faults/attacks.

While the first two layers, prevention and detection already provide strong defense, recovery is the fallback option should these layers fail to protect the system. However, recovery being the last phase, needs protection (or a fallback). Mehresh et al. [20] discuss the possible attacks on the recovery phase of a critical system. Because adaptation/evolution mechanisms are generally activated during or after recovery, they are rarely effective if recovery fails. Therefore, recovery phase needs further protection to assure mission survivability.

One of the major challenges of designing a survivable system is to ensure that all the precedented or unprecedented threats are dealt with, while conserving the timeliness property of the mission. Since dealing with unprecedented attacks (zero-day attacks) requires monitoring the entire traffic, it becomes difficult to ensure timeliness property. Hence, these two requirements of surviving all kinds of threats and conserving the timeliness property are contradictory in nature. We propose deception as a tool to handle this conflict and even out the asymmetry in cyber warfare. Defensive deception is an act of intentional misrepresentation of facts to make an attacker take actions in defender's favor [7]. In this work, we make the following contributions:

- Study current trends in sophisticated attacks against mission critical cyber systems and present a next generation threat analysis/model (Section 3).
- Derive formal set of requirements for a survivable system to defend against such attacks (Section 4).
- Transform these requirements into a survivability framework where each design choice is supported with evidence and detailed reasoning (Section 5).

We review the related work in Section 2. In Section 3, we present an assessment of next generation threats. Formal requirements for the survivability framework are laid out in Section 4 and Section 5 presents the framework in detail. Section 6 concludes the paper by discussing the challenges

involved in implementing this framework.

2. Related Work

The static nature of today's networks presents a sitting and vulnerable target. Patch development time for most exploits is much higher than the exploit development time. Repik [29] documents a summary of internal discussions held by Air Force Cyber Command staff in 2008. His work makes a strong argument in favor of using deception as a tool of defense. He discusses why planned actions taken to mislead hackers have merit as a strategy and should be pursued further.

Deception itself in warfare is not new [33], [6]. However, deception has several associated legal and moral issues with its usage in today's society. Cohen [6], the author of deception toolkit [5] discusses moral issues associated with the use of deception throughout his work. Lakhani [13] discusses the possible legal issues involved in the use of deception-based honeypots.

Deception aims to influence an adversary's observables by concealing or tampering with the information. Murphy [21] discusses the techniques of deception like, fingerprint scrubbing, obfuscation, etc. Her work is based on the principle of holding back important information from the attacker to render the attack weak. There is vast literature and taxonomies [6], [30], [29] on the use of deception to secure computer systems and information in general. Our framework essentially builds on these principles to handle sophisticated attacks.

3. Next generation threat assessment

In this section, we discuss the latest trends in sophisticated attacks on critical systems. This analysis helps us derive the attack patterns required to design a secure solution for the next-generation of critical systems.

Today's market forces and easy access to high-end technology have changed the cyber attack landscape considerably. As reported by Washington Post [22], malicious sleeper code is known to be left behind in the U.S. critical infrastructure by state-sponsored attackers. This sleeper code can be activated anytime to alter or destroy information. Similar stealth methodologies are also employed during multi-stage delivery of malware discussed in [28] and the botnet's stealthy command and control execution model in [11]. We already see a rising trend of stealthy smart malware all around [10]. Stuxnet, for instance, sniffs for a specific configuration and remains inactive if it does not find it. "Stuxnet is the new face of 21st-century warfare: invisible, anonymous, and devastating" [9]. Another instance of smart malware is 'Operation Aurora' that received wide publicity in 2009-10. The most highlighted feature of Aurora is its complexity, sophistication and stealth [16]. It includes numerous steps to gain and maintain access to privileged systems until the attacker's goals are met. The installation and working of this malware is completely hidden from the user.

A recently published report by McAfee surveyed 200 IT executives from critical infrastructure enterprises in 14 countries [2]. The report documents cyber-security experts expressing concerns about the surveillance of U.S. critical infrastructure by other nation-states.

Considering these trends, we set our focus on addressing attacks from a resourceful, adaptive and stealthy adversary. Note that aggressive attackers are easier to spot and hence the routine security measures generally take care of them. However, multi-shot, stealthy attackers rely on techniques that are difficult to detect and thus need innovative defense [28].

An attacker can cause maximum damage to a mission critical system during its crucial stage (like, the final stage). Multi-shot attackers are stealthy. They sniff around the system, install backdoors, place sleeper code, fragmented malware, etc., while evading detection. Thus, these stealthy attackers have a long time to infect the system. If discovered at a late stage, sometimes the only way left to recover is a system-wide sanitation which may disrupt the mission. Such drastic measures can cause a huge financial loss due to the heavy investments made during the course of a mission. At other times, the defender may not even get an opportunity to react. Thus, the need is to make these stealthy attackers manifest an easily detectable pattern at as early a stage as possible. Additionally, some stealth is required on defense system's part if it aims for a no-loss recovery from the attack. Most smart attackers and malware come with a contingency plan (to destroy or steal information on discovery). Thus, spooking an attacker without being prepared for the consequent contingency plan can be catastrophic. Thus, detection needs to be stealthy too, until the defender comes up with a plan to deal with a spooked attacker.

Based on the discussion above, we design a perceived smart attack flow. It is an extension of the basic one presented by Repik [29]. The attack flow is described in Algo. 1. Let ϕ be the set of exploitable vulnerabilities for a system with state s(t), where t is time. For each vulnerability ν in ϕ, the amount of resources required to exploit it is represented by r[ν]. Total resources available to an attacker is $\hat{r}$. Risk associated with exploiting each vulnerability ν is $\rho[\nu]$. Maximum risk that the attacker can afford is $\hat{\rho}$.

A sophisticated attack usually starts with intelligence gathering and initial planning. Based on the available resources, an attacker decides either to exploit a currently known vulnerability or keep searching for more. Attack occurs in multiple stages involving installing backdoors, rootkits, etc., until a crucial stage is reached. An attack during crucial stage has the maximum pay-off for the attacker. If discovered, most attackers have a contingency plan that may involve deleting or destroying information.

Algorithm 1 Attack pattern for sophisticated attacks

```
1: while TRUE do
2:   while φ = NULL AND ∀ν,ρ[ν]≥ ρ̂ do
3:     Gather intelligence
4:     Develop exploits
5:     Perform network reconnaissance
6:     Update vulnerability set φ
7:   end while
8:   if ∃ν, (r[ν]≤ r̂ AND ρ[ν]≤ ρ̂) then
9:     Install backdoors; Update r̂
10:    while s(t) ≠ ATTACK_DISCOVERED do
11:      if s(t) ≠ CRUCIAL_STAGE then
12:        WAIT
13:      else if ∃ν, (r[ν]≤ r̂ AND ρ[ν]≤ ρ̂) then
14:        Attack and exploit ν; Update r̂; Assess damage
15:        if s(t)=COMPROMISED then
16:          Operation successful and Exit
17:        end if
18:      else
19:        Terminate operation
20:      end if
21:    end while
22:    if Contingency plan exists then
23:      Execute contingency plan
24:    else
25:      Terminate operation
26:    end if
27:  else
28:    Terminate operation
29:  end if
30: end while
```

4. Formal requirements

In light of the threat assessment presented in the previous section, we now list down requirements for a state-of-the-art deception-based security framework for mission survivability.

1) **Prevention**: It is generally the first step towards developing any effective solution in dealing with security threats. Prevention not only attempts to stop the attacks from succeeding, but also dissuades attackers with limited resources.
2) **Detecting the smart adversary**: We identify two main challenges for developing a security solution. First, it should force or manipulate a stealthy attacker into leaving a discernible and traceable pattern. Second, detection of such a pattern should be hidden lest the attacker should get spooked and execute a contingency plan for which the defender is not likely to be prepared. In addition to that, the solution should provide basic prevention, detection and recovery while conserving the timeliness property of the mission.
For a given system state $s_1(t)$, there is a set ϕ_1 of suspicious actions (for instance, a possible exploit attempt). A user that chooses an action from this set is malicious with a probability p. However, he could be benign with a probability 1-p. Let system states $s_1(t)$, $s_2(t)$,....,$s_n(t)$ (where, n is the total number of system configurations) have ϕ_1, ϕ_2,....,ϕ_n as their respective sets of suspicious actions. For some system states, this set of actions can be more clearly categorized as malicious with higher probabilities p_i where, $1 \leq i \leq n$. Choosing such states more frequently helps the defender to come up with a clear user profile in a shorter time. In honeypots, a defender can choose states with higher p_i's, which means that if an attacker keeps choosing the actions from the set ϕ, his probability of being malicious ($p_1.p_2.p_3....p_n$) will cross the threshold in a shorter time. Thus, choosing and controlling these states is crucial in determining if an attacker is malicious with a higher probability in a shorter time.
3) **Effective recovery with adaptation**: If the attacker has penetrated the system via existing vulnerabilities, recovering the system to the same old state does not remove these vulnerabilities. Therefore, the need is to ensure that during each recovery, vulnerabilities that are being exploited are patched. In this paper, we assume proactive recoveries that are periodically scheduled. It is much easier to predict the timing impact of proactive recoveries and hence conserve the timeliness property of a survivable system. Reactive recoveries, if evoked excessively, can harm system's performance and mission's survivability.
4) **Zero-day attacks**: Any good survivability solution must deal with zero-day attacks. Several anomaly-based detection systems have been proposed in order to detect such attacks [4]. However, Liu et al. [15] describe the big challenge, "how to make correct proactive (especially predictive) real-time defense decisions during an earlier stage of the attack in such a way that much less harm will be caused without consuming a lot of resources?" Schemes that attempt to recognize a zero-day stealth attack usually take two approaches: predictive and reactive. Under the predictive approach, all the suspected policy violations are taken as a sign of intrusion. This results in higher rate of false alarms and hence service degradation. Under the reactive approach, defender takes an action only when he is somewhat sure of a foul play. Generally, it is difficult to know when to react. If the system waits unless a complete attack profile emerges, it may be too late to react. A good trade-off is offered by honeypots (a form of deception). The defender redirects all the suspicious traffic through honeypots which is responsible for blacklisting/whitelisting the traffic flows [25], [24].

Authors in [14], [13] introduced methodologies for employing honeynet in a production-based environment.

5) **Conserving timeliness property**: Timeliness property describes the capability of a mission survivable system to stick to its originally planned schedule. This being said, a schedule can account for periodic recoveries and some unexpected delays due to miscellaneous factors. In order to conserve this property, it is essential that all the indeterministically time-consuming operations be moved out of the mission's critical path. Thus, it is essential to redirect the suspicious traffic to a separate entity for further examination.
6) **Non-verifiable deception**: A good deception should be non-verifiable [23]. Deception is difficult to create but easier to verify. For instance, an attacker attempts to delete a file. Even if a deceptive interface gives a positive confirmation, the attacker can always verify if the file exits.

   For a state s(t), an action χ is expected to have an effect ω. Generally, deception (like in honeypots) involves confirming that χ has been performed but the effect ω is never reflected in the system. If the attacker has a feedback loop to verify ω, a deception can be easily identified. Therefore, either the feedback loop needs to be controlled so as to give the impression that ω exists, or the feedback loop should be blocked for all regular users. An open and honest feedback loop will help attacker in figuring out ways around deception by trial-and-error.

5. The Framework

5.1 Basics

Preventive deception is the first step in mission survivability. Traditionally, measures like firewall, encryption techniques, access control, etc. have been used as preventive measures. These measures have proved to be very successful in deterring weak adversaries. However, strong and determined adversaries are always known to find their way around these. McGill [17] suggests that the appearance of a system being an easy or a hard target determines the probability of attacks on it. Based on similar literature, we categorize deception-based prevention methodologies under following four headings:

- *Hiding*: Hiding is the most basic form of deception. One could use schemes like fingerprint scrubbing, protocol scrubbing, etc. to hide information from an attacker [34], [31]. Similarly, these schemes could also be used to feed false information to the attacker. Yuill et al. [35] have developed a model for understanding, comparing, and developing methods of deceptive hiding.
- *Distraction*: McGill [17] demonstrates that given two targets of equal value, an attacker is more likely to attack the target with lesser protection. However, Sandler and Harvey analytically prove that this tendency continues only till a threshold. If more vulnerabilities are introduced to a system, an attacker's preference for attacking that system does not increase beyond a certain threshold.

  System observables that attackers rely on can be manipulated to feed misinformation or hide information from attackers. Thus, strategies can be devised to affect an attacker's perception about the system. Studies like [18] model threat scenarios based on target's susceptibility and attacker's tendencies. Such models can be used to assess the attractiveness of a target to an attacker if its apparent susceptibility is manipulated via its observables.

Axiom 1: Adding more vulnerabilities to one of the two equal-value systems increases the likeliness (till a threshold) of an attack on the one with more vulnerabilities.

- *Dissuasion*: Dissuasion describes the steps taken by a defender to influence an attacker's behavior in his favor. It involves manipulating system observables to make it look like it has stronger security than it actually does. This may discourage attackers from attacking it. As shown in Algo. 1, if the estimated resources for exploiting the system go over $\hat{r}$ or the estimated risk goes over $\hat{\rho}$, the attacker will be dissuaded from attacking the system. Dissuasion is generally implemented as deterrence or devaluation. Deterrence involves a false display of greater strength. Devaluation, on the other hand, involves manipulating observables to lessen the perceived value that comes out of compromising a system. McGill [17] develops a probabilistic framework around the use of defensive dissuasion as a defensive measure.

Deception techniques are complementary to conventional prevention techniques rather than a replacement.

Axiom 2: False display of strength dissuades an attacker from attacking the system.

Axiom 3: Increasing or decreasing the perceived value of a system affects the attacker's preference of attacking the system favorably or adversely, respectively.

Honeypot is a tool of deception. It generally comes across as a system capable of a low-resource compromise with high perceived gains. Honeypot not only distracts an attacker from attacking the main system, but also logs attacker's activity heavily. Studying these logs can help the defender to gauge an attacker's capability and come up with a good strategy to ward off any future attacks. Spitzner describes honeypot as "a security deception resource whose value lies in being probed, attacked, or compromised" [32]. Honeypots are generally classified into two categories: Physical and Virtual. Physical honeypots are when real computer systems are used to create each honeypot. Virtual honeypots use

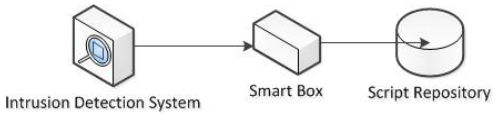

Fig. 1: Smart-Box

software to the workings of a real honeypot and the connecting network. They are cheaper to create and maintain and hence are used in the production environments more often. Virtual honeypots are further divided into high interactive and low interactive honeypots. Qasswawi et al. [27] provide a good overview of the deception techniques used in virtual honeypots.

High interactive honeypots provide an emulation for a real operating system. Thus, the attacker can interact with the operating system and completely compromise the system. Some examples are User Mode Linux (UML), VMware, Argos, etc. Low-interaction honeypots simulate limited network services and vulnerabilities. They can not be completely exploited. Examples are LaBrea, Honeyd, Nepenthes, etc. [26], [27].

Cohen's Deception Toolkit (DTK) laid the groundwork for low-interaction honeypots [5]. It led to the development of advanced products like Honeyd. Honeyd [32] simulates services at TCP/IP level in order to deceive tools like Nmap and Xprobe. Though it does not emulate the entire operating system, its observables are modified to give the impression that it does.

Honeypot farm is a cluster comprised of honeypots of the same or different kinds. Hybrid honeypot farms consist of a mixture of low and high interactive honeypots.

Smart-box is a module that we proposed to help figure out the best deception for a specific suspicious traffic flow. Conceptually, a smart-box works as shown in Fig. 1. It takes input from the IDS about the suspected traffic flow. The logic in smart-box then decides Attackers Intent, Objectives and Strategies (AIOS) based on this information [15]. Then it maps the AIOS to deception scripts. These scripts are stored in the script repository.

5.2 Design

Building up from the concepts discussed in the previous subsection, we extend the model presented in [13] to design our deception-based survivability framework.

As shown in Fig. 2, the mission survivable (production) system runs behind several layers of protection including firewalls, deception, etc. The first layer of proxy servers uses Axioms 1, 2 and 3 to mislead attackers into choosing systems that will re-route their traffic to honeypot farm via the smart-box. Rest of the unsuspected traffic goes through the main server, the firewall and the intrusion detection system. Intrusion detection system is another layer of defense which re-routes any suspicious traffic to the honeypot farm for further analysis.

Suspicious traffic is generally sieved out based on two criteria: either the intrusion detection system identifies an attack pattern in the traffic flow or the traffic originates to/from dark address space. Dark address space is the set of Internet's routable address reserved for future network expansion. These two criteria worked just fine until cloud computing came along. Now attackers can launch their attacks from behind the cloud using valid IP addresses and evade detection. Therefore, in addition to employing the above-mentioned two methods, we introduce a layer of distraction proxy servers. This layer contains a main server which is widely publicized to legitimate clients. This main server is extremely secure and its observed security is further enhanced (deception/deterrence). Thus, amateur attackers are dissuaded from attacking it. Other proxy servers expose a specific set of non-essential, vulnerable services. For instance, one server can keep the *ssh* port open to accept the traffic, while the other can mislead the attacker into thinking that it is running a vulnerable version of Windows operating system. These servers not only distract the malicious traffic away from the main server but, also inform the smart-box about attackers' intentions (based on their preference of proxy servers and vulnerabilities that they try to exploit).

In this design, we use smart-box to optimize resource allocation in hybrid honeypot farms. These honeypots should be assigned to the traffic flows based on the assessment about each flow's AIOS. This is because low-interaction honeypots can be easily verified if attacker suspects deception and tries to go in deeper. Use of high-interactive honeypots for deception is more fool-proof but consumes more computing and memory resources. Thus, smart-box helps in smart allocation of these honeypot resources by assessing the nature of an attack and re-routing the traffic to appropriate honeypots (similar to loading the deception scripts).

Logging tools and analyzer in the honeypot farm recognize an attack and create a complete attack profile. Based on this attack profile, the flow is whitelisted and forwarded to the production server or blacklisted. If blacklisted, either automated patches, if available, are executed in the next recovery cycle, or a system administrator is alerted. This is the step where the attack profile helps the defender to develop an effective patch for the next recovery cycle, while the unsuspected malicious actor stays busy playing with the honeypot. Thus, deception buys defender the time to design an effective recovery.

Since a system is "as secure as its weakest point", we need to make sure that this framework not only provides good security but is tamper-proof at all times. Since all the modules in this design like, proxy servers, the traffic redirection module, intrusion detection systems, etc. are connected to the same network, they are always susceptible to intrusions. Therefore, these modules need to be tamper-proof in order for the entire design to be tamper-proof.

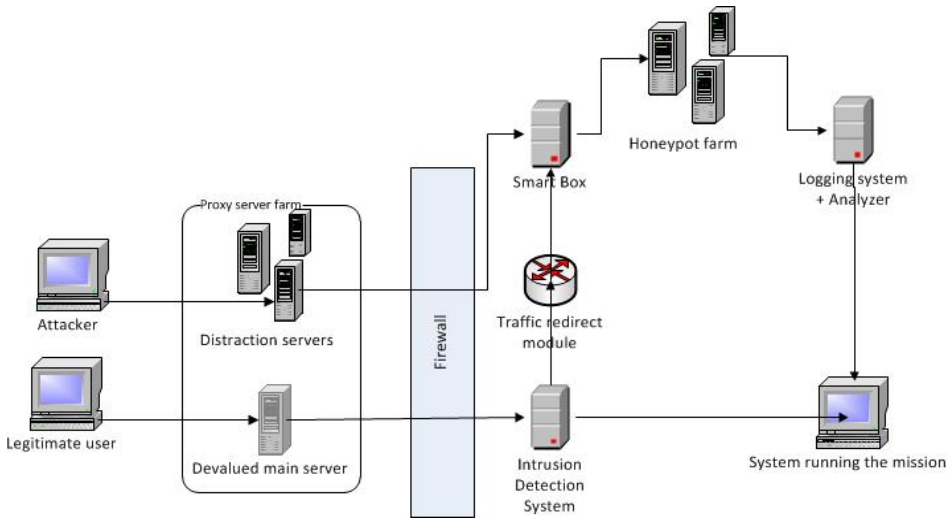

Fig. 2: Deception framework for mission survivability

We can use techniques like lightweight cyclic monitoring in order to make sure that IDS on all these modules (like proxy servers, IDS, etc.) stay tamper-proof [19]. Then using the scheme described by Mehresh et. al in [20], the integrity signature of each module is surreptitiously detected and sent to the production system for verification. The detection is secret so the attacker is not spooked. This arrangement introduces a cyclic integrity-check. All modules make sure that production system works tamper-free at all times, while the production server takes care of the integrity-check for all modules.

Anagnostakis et al. [1] proposed shadow honeypots as an effective solution to deploy honeypots in a production environment. Shadow honeypots use a combination of anomaly intrusion detection systems and shadow honeypots. A variety of anomaly detectors monitor traffic in the network and the suspected traffic is forwarded to a shadow honeypot. Shadow honeypot is an identical copy of the production server but instrumented to detect potential attacks. Misclassified traffic is verified by the shadow and transparently handled correctly. We see many challenges in this approach. First, predictive anomaly detectors (higher sensitivity) will have more false positives and will direct more misclassified traffic to the shadow honeypot, creating unnecessary delays and overhead. Reactive anomaly detectors (lower sensitivity) will take more time to create a complete profile and may miss a lot of malicious traffic before identifying a problem with the flow. Moreover, identifying zero-day attacks ask for a higher sensitivity intrusion detection. Additionally, each suspected traffic flow may need separate copies of shadow honeypot (else an attacker can verify deception by initiating two parallel malicious flows). This further increases the overhead.

6. Discussion and Conclusion

The most important factor to consider while designing any aspect of this deception framework is to remember that nothing will remain a secret if it is widely deployed. Hence, an effective deception must assume that an attacker knows about its existence with some probability. That's why all deceptions should be non-verifiable. In this case, when an attacker sees several proxy servers with vulnerabilities, he sends traffic flows to all the servers. The flow that gets through the fastest is the main server (under the safe assumption that going through honeypot farm adds to the delay). That's why, any feedback loop for the attacker must also be controlled with deception. Discussing deception in feedback loop is beyond the scope of this paper.

Another major challenge is the design of the smart-box. A smart-box performs two major functions: assess the nature of the traffic flow and, map the AIOS to a honeypot. Designing an implementation of both these functions is a major challenge and will benefit excessively from the use of machine learning algorithms. Deceptions in honeypots can also be made customizable based on the parameters provided by the smart-box. Other challenges like designing proxy servers, re-routing, choosing the IDS, etc. depends on the system that the framework is used for and the designer.

In this paper, we focus on designing survivable mission critical systems. We began with analyzing the current cyber security attacks to derive the next generation threat assessment. Multi-shot, stealthy attacks came out as a major threat in this assessment. We then defined a set of requirements around this threat for a survivability framework. Based on evidence and existing literature, we identified deception as an important tool of defense and designed a deception framework for survivable systems. This framework deals with zero-day attacks while still conserving the timeliness

property of a mission. It uses concepts of deception to introduce a preventive layer of proxy servers that helps system to further narrow down the suspicious traffic. This traffic then gets rerouted to a smart-box that selects the honeypot this traffic is forwarded to. Honeypots provide an important functionality of uncovering the stealthy patterns in these traffic flows with a higher probability in a shorter time. This way, the framework helps in identifying and rooting out the stealth attacks at an early stage.

A major advantage of this framework is the strong recovery that it provides. It buys defender more time to analyze the suspected traffic flow without spooking the adversary. The analyzer and log modules help in designing a secure and more effective recovery patch. Hence, this framework ensures system survivability equipped with a strong recovery phase.

In future, we plan to address the challenges we discussed above and work towards implementing a prototype of this framework.

7. Acknowledgments

This research is supported in part by DoD Grant No. H98230-11-1-0463. Usual disclaimers apply.

References

[1] K. G. Anagnostakis, S. Sidiroglou, P. Akritidis, K. Xinidis, E. Markatos, and A. D. Keromytis. Detecting Targeted Attacks Using Shadow Honeypots. *Proceedings of the 14th conference on USENIX Security Symposium*, page 9, 2005.

[2] S. Bake, N. Filipiak, and K. Timli. In the Dark: Crucial Industries Confront Cyberattacks. *McAfee second annual critical infrastructure protection report*, 2011.

[3] R. Baskerville. Information Warfare Action Plans for e-Business. *3rd European Conference on Information Warfare and Security*, pages 15–20, 2004.

[4] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41:15:1–15:58, 2009.

[5] F. Cohen. Deception Toolkit, 2001.

[6] F. Cohen, D. Lambert, C. Preston, N. Berry, C. Stewart, and E. Thomas. A Framework for Deception. *IFIP-TC11, Computers and Security*, 2001.

[7] D. C. Daniel and K. L. Herbig. *Strategic Military Deception*. Pergamon Press, 1982.

[8] R. J. Ellison, D. A. Fisher, R. C. Linger, H. F. Lipson, T. A. Longstaff, and N. R. Mead. Survivability: protecting your critical systems. *IEEE Internet Computing*, 3:55–63, 1999.

[9] M. J. Gross. A Declaration of Cyber-War, April 2011.

[10] A. Kapoor and R. Mathur. Predicting the future of stealth attacks. *Virus Bulletin Conference*, 2011.

[11] E. J. Kartaltepe, J. A. Morales, S. Xu, and R. Sandhu. Social network-based botnet command-and-control: emerging threats and countermeasures. *Proceedings of the 8th international conference on Applied cryptography and network security (ACNS)*, pages 511–528, 2010.

[12] K. J. Knappa and W. R. Boulton. Cyber-Warfare Threatens Corporations: Expansion into Commercial Environments. *Information Systems Management*, 23:76–87, 2006.

[13] A. D. Lakhani. Deception techniques using Honeypots. *MSc Thesis, ISG, Royal Holloway, University of London*, 2003.

[14] J. G. Levine, J. B. Grizzard, and H. L. Owen. Using honeynets to protect large enterprise networks. *IEEE Security and Privacy*, 2:73–75, 2004.

[15] P. Liu, W. Zang, and M. Yu. Incentive-based modeling and inference of attacker intent, objectives, and strategies. *ACM Transactions on Information and System Security (TISSEC)*, 8, 2005.

[16] McAfee Labs and McAfee Foundstone Professional Services. Protecting your critical assets, lessons learned from "Operation Aurora". *Technical report*, 2010.

[17] W. L. McGill. Defensive dissuasion in security risk management. In *IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 2009.

[18] W. L. McGill, B. M. Ayyub, and M. Kaminskiy. Risk Analysis for Critical Asset Protection. *Blackwell Publishing Inc*, 27:1265–1281, 2007.

[19] R. Mehresh, J. J. Rao, S. J. Upadhyaya, S. Natarajan, and K. Kwiat. Tamper-resistant Monitoring for Securing Multi-core Environments. *International Conference on Security and Management (SAM)*, 2011.

[20] R. Mehresh, S. J. Upadhyaya, and K. Kwiat. Secure Proactive Recovery - A Hardware Based Mission Assurance Scheme. *Journal of Network Forensics*, 3:32–48, 2011.

[21] B. S. Murphy. Deceiving Adversary Network Scanning Efforts Using Host-Based Deception. Technical report, Air Force Institute of Technology, Wright-Patterson Air Force Base, 2009.

[22] E. Nakashima and J. Pomfret. China proves to be an aggressive foe in cyberspace, November 2009.

[23] V. Neagoe and M. Bishop. Inconsistency in deception for defense. In *Proceedings of the 2006 workshop on New security paradigms*, 2007.

[24] R. R. Patel and C. S. Thaker. Zero-Day Attack Signatures Detection Using Honeypot. *International Conference on Computer Communication and Networks (CSI- COMNET)*, 2011.

[25] G. Portokalidis and H. Bos. SweetBait: Zero-Hour Worm Detection and Containment Using Low- and High-Interaction Honeypots. *Science Direct*, 51:1256–1274, 2007.

[26] N. Provos and T. Holz. *Virtual Honeypots: From Botnet Tracking to Intrusion Detection*. Addison-Wesley, 2008.

[27] M. T. Qassrawi and H. Zhang. Deception Methodology in Virtual Honeypots. *Second International Conference on Networks Security Wireless Communications and Trusted Computing (NSWCTC)*, 2:462–467, 24–25, 2010.

[28] M. Ramilli and M. Bishop. Multi-Stage Delivery of Malware. *5th International Conference on Malicious and Unwanted Software (MALWARE)*, 2010.

[29] K. A. Repik. Defeating adversary network intelligence efforts with active cyber defense techniques. Master's thesis, Graduate School of Engineering and Management, Air Force Institute of Technology, 2008.

[30] N. C. Rowe and H. S. Rothstein. Two Taxononmies of Deception for Attacks on Information Systems. *Journal of Information Warfare*, 3:27–39, 2004.

[31] M. Smart, G. R. Malan, and F. Jahanian. Defeating TCP/IP stack fingerprinting. *Proceedings of the 9th conference on USENIX Security Symposium*, 9:17–17, 2000.

[32] L. Spitzner. Honeynet Project, Know Your Enemy: Defining Virtual Honey-nets, 2008.

[33] S. Tzu. *The Art of War (Translated by James Clavell)*. Dell Publishing, New York, NY, 1983.

[34] D. Watson, M. Smart, G. R. Malan, and F. Jahanian. Protocol Scrubbing: Network Security Through Transparent Flow Modification. *IEEE/ACM Transactions on Networking*, 12:261–273, 2004.

[35] J. Yuill, D. Denning, and F. Feer. Using Deception to Hide Things from Hackers: Processes, Principles, and Techniques. *Journal of Information Warfare*, 5:26–40, 2006.