Deception Based Cyber Attacks Detection Framework for Multiple Public Facing Internet Servers

Muhammad ISMAIL* and Jiang WANG

College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China

*Corresponding Author

Keywords: Deception, Honeypots, Mini traps, Attacks Detection, Apache.

Abstract. With the passage of time, cyber threats and attacks are becoming more and more sophisticated and are continually evolving at an alarming rate. To confront these attacks, the existing security solutions are failing repeatedly because the attackers are coming up with new and advance attack vectors. Proactive detection of these attacks is becoming a tough challenge. Incorporation of deception technology and tools have the potential to early detect and guard against modern cyber-attacks and threats. Deception technology and tools enable the administrators and security specialists to entice the adversaries’ target selection and therefore, lead them to execute actions that might disclose their aim and presence. A structured framework and proof-of-concept implementation of a Deception Server (DS) is proposed. DS centrally maintains and provides continuous deception across multiple internet facing application server’s environment and detects the activities of adversaries in their reconnaissance stage of an attack. Additionally, DS is also capable to mislead or astray the attackers from a successful target infiltration as it can generate deceit response on-the-fly.

Introduction

One of the critical phases of successful and mentioned attacks is reconnaissance. To provide services to their clients, almost all organizations deploy and use a variety of Internet-facing servers and services. By system arrangement, these servers and services honestly respond to all seeking requests whereas some requests that could not be identified are dropped or blocked. But still, from the attacker’s perspective, the particular dropped session could help them to extract some valuable information. Reports suggest that these services and server facing outside world are increasingly becoming targets and compromised as an entry point to launch larger attacks. The list of the OWASP’s top 10 security risks suggests that at-least six security risk are due to the design of computer system which always responds faithfully to incoming requests [1]. A new Verizon-Data-Breach-Investigation-Report (DBIR) [2]. In said research figures like that in 2016, 66% of attack breaches are unknown or unnoticed for a very long time. This trend is very alarming and on the rise from 56% in 2012. The majority of these incidents i.e. 35% were identified to be a web application. Furthermore, 84% of recorded attacks are so lethal that it took very little time to penetrate and infect computing systems [2]. Said research also revealed that very small portion i.e. only 5% of these threats and attacks were spotted using conventional IDS systems [2].

Deception based methods render profound leverage over orthodox security measures. Presently, majority security techniques are antiphonal controls to hackers because as soon as an attack detects, it is dealt by the defense mechanism in place. Sooner or later, relentless hackers find a weakness by knowing how the security tools interact with their request which ultimately leads them to a successful penetration. In contrast, deception-based mechanisms normally concentrate on hacker’s perceptions i.e. by manipulating attackers and at the same time enabling their opponents (defense team) to take necessary measures while orthodox security measures focus on detecting or preventing. To summarize it can be inferred that usually conventional security controls are
A framework of Deceptive Server (DS) is also proposed which provides deception in a centralize manner across internet facing application server’s environment. DS has the capability to (a) identify that the user is harmless or malicious, (b) reduce the risk of information leakage to an attacker, (c) moreover, it can also be helpful in reducing the probability of false positives.

A proof the concept implementation of DS has also been presented. Performance assessment of said model is also presented.

Related Work
For attack detection, the use of honeypots and honeynets been proposed. Placement of these tools on both external and internal end of a system is analyzed [3]. Similarly, honeypots are also used to detect botnets attacks [4] and similar attacks on wireless communication networks. In another proposed model honeyfiles were employed in order to detect the unauthorized resource utilization and access [5]. In other related research deception mechanism of honeywords has been used which demonstrated to detect rogue or compromised user password credentials [6]. In another literature [5] researcher demonstrated that why conventional security measure and defenses are not enough to counter unknown attacks and APTs. Authors also demonstrated that with a security model which comprise of various honeypots i.e. Vlahala, Dionaea and Honeyd how can we pro-actively detect APTs in a network. In another related research [7], the author proposes that how to disrupt and halt an APTs attack using deception based technique including honeypots. Research in [7] was extended in the literature [8] and the win-win game model was proposed that elaborates that how can be protected from sophisticated attacks in the future. In another literature [9], researcher discusses various advantages of deception based technologies in cyber defense against unknown and sophisticated attacks. The author summarized his work with the conclusion that with the use of stated technologies organizations, firstly, can forestall an attack, secondly, frustrates an attacker to make an error and thirdly, the rate of false-positive would be low.

Additionally, in another research [10], the author proposes a model that detects unknown attacks and APTs using honeypots. The authors used honeypots agents. These agents were able to detect attacker activities by comparing and distinguishing between the malicious and genuine user, once the attacker is inside and already penetrated a target network. Oftentimes attackers gather said information for company web pages or other means and research suggests that there are four times more chances that targeted attacks i.e. spear phishing, to be successful [11]. In addition, the concept of “shadow honeypots” [12] have also been used. In their proposed shadow honeypots architecture, honeypots are incorporated in a real system and it diverts suspect-able traffic to a shadow system for thorough investigation. Sensors capable of Anomaly-Detection were positioned in front of the real production system. These sensors then take the decision to send the traffic to either to a real system or shadow honeypot machine. In another research experiment conducted by Heckman, a tool named “blackjack” was developed. Said tool dynamically makes copies of a production server internal state after hiding critical data and planting a deceit. Adversaries were then directed to that fake instance [13].

There are some shortcomings in early proposals. Mainly they are passive systems and come into action whenever an attacker pings resources that are not supposed to touch. Additionally, the loose pairing between counterfeit and real systems and also a higher difficulty factor to develop plausible deceits undermines the effectiveness of deception.

Proposed Framework for Attack Detection
The fundamental idea behind deceptive server (DS) is to employ deception in a protected or secure system, relatively at a higher logical view. Dissimilar to the conventional deception mechanism, the proposed DS applies deception to the system resource in two ways. Firstly, it responds over every
time with a new deceptive response that is generated concurrently. Secondly, it modifies system resource on-the-fly before sending it to the attackers or adversaries. DS aids the administrators or defenders by creating deceptive responses in two categories. First, the attack defenders will know that if the current request for the resource is a malicious or not. In the first category, deception will be added in such a way that there will be no need to entirely change the system resources every time. In the first category, deception will be employed in order to create like mini-traps and only that requester will fall for it that has the malicious intent. The request is set to be malicious if it is available in server blacklist, through a firewall or an attacker access the DS traps. Whenever, from first category process, defenders determine that the nature of requester is malicious and which could be an adversary, in the next i.e. second category, a deceit will be injected with the aim to confuse and leads the adversary off the right path which will give enough time for defenders to proactively act and take appropriate countermeasures. DS can be deployed in computer network environment as a server and its premier function is to provide decision-making in order to supplement servers facing outside world with deception in a Centralize manner. DS can be divided into two broad components i.e. 1) DS Kernel and 2) DS hook.

The kernel part of DS provides decision-making on how to treat incoming requests. The hook part of DS converts the decisions made in kernel component into the application specific format and action realized by servers present in the computer system environment. The overall design is shown in (Figure 1). Whereas the possible outcomes of DS kernel and its decision flow chart are given below in (Figure 2). In order to identify an attacker or make a fake image of the real system to confuse adversaries, there are two basic deceit types which are utilized and planted by DS into responses.

**DS Traps.** DS Traps inject plausible and tempting false variable which may have no such real use on a particular server. Whenever these fake variables are accessed or emended by an attacker or automated tools, defenders of the system will be proactively notified about the possibility of an attack which may still be at an early stage. Example of such traps which can be configured is given in the (Figure 3) and (Figure 4). A web server has files “.htaccess and robot.txt” mainly used for administrating purposes i.e. configuring the conduct or rules applied to responses. A fake password protected web-page can be added to mentioned files, acting as a deceit for adversaries to fall in it.

---

**Figure 1.** DS Design Overview.  
**Figure 2.** DS Kernel Decision Outcome Flow Chart.  
**Figure 3.** An example of DS Trap in `.htaccess`.  
**Figure 4.** An example of DS Trap in Hidden Data.
In both examples, a user with only malicious intent will try to access i.e. “admin-access.html” and change the variables i.e. “debug=true”. DS can flag such users and returned them with an active deceptive response.

**DS Active Deceptive Response.** When DS detects a malicious action with connected application servers, it is also capable to inject deception in order to provide a fake view of the entire system and confuse or lead the adversaries astray. One example is that a fake server’s state i.e. presents a fake error message of “server down for maintenance” can be used in order to provide a fake view of servers’ performance. An expedient attacker will leave the current server and go to try his luck on another server on his/her list. Also, DS can feed adversaries with false public information that can be used to launch a wide scale attack at a later stage.

**Implementation and Performance Analysis of the Proposed Framework**

DS has two primary operation mode components i.e. decision making and an administration component. It is implemented as a command line tool in Python. Administration component provides the defenders an ability to configure and setup the deception server. In order to store all deception rules, user connection requests information and determine action, a SQL database is configured internally in DS. For code and database interaction *peewee* library (an ORM i.e. Object-Relational-Mapper) has been implemented.

DS inputs (receive as command line arguments) and outputs are application independent. Application server hook (Apache web server) prepares the request for DS and do the request translation into instruction that is application specific. Apache web server is configured to use mod-security element/module in order to enforce its WAF (*Web-Application-Firewall*) which hooks a script with DS for communication. An inside Apache server *Lua* script is created to intercept all incoming server requests after the request headers are parsed. Also said the script is used to prepare and send the request to DS. Additionally, it parses the response of DS and inserts it inside WAF. Apache web server provides good configuration flexibility due to which DS can be integrated using a specific directory on the server which can be its part or only when a particular type or specific resource is requested. Apache hook is implemented and internally configured as *.conf* file. It is loaded and runs consistently in the back ground when Apache starts. The overall integration of Apache with DS is shown in (Figure 5).

A number of experiments have been designed to analyze and check DS performance and its integration with Apache web server. We evaluated the performance by running the DS with and without Apache. To measure the difference, the *httpperf* and *autobench* tool are used in order to benchmark the performance of Apache in mentioned two cases.

![Figure 5. Apache Integration with DS.](image1)

![Figure 6. Comparison between Apache Mod-Sec and DS.](image2)

Average response time in milliseconds against a number of requests simultaneously sent was recorded in four different configuration categories w.r.t content filtering and *mod-security* module of Apache is shown in Figure 6). The red line and yellow line indicates Apache performance with and without enabling *mod-security* module with its default configuration and rules. The blue line
shows the Apache’s average response time when *mod-security* is enabled and ModSecurity-Core-Rule-Set of OWASP version 2.2.9 is enforced. The green line shows Apache performance when DS is deployed along with *mod-security*. In this case, mod-security is used to enforce DS rules. Running all DS logic furnishes us a response-time of less than a millisecond which is identical to other running modes as shown in Figure 6).

To further analyze that DS is providing its core role, we set an experiment that is by sending one hundred requests to internet facing servers. Out of hundred requests generated, forty five were from sources which were listed in the server’s blacklists. We find out that DS has successfully injected traps to all request that were present in the blacklists as shown in (Figure 7).

![Figure 7. Request Treatment and Injection Ratio of DS.](image)

**Significance of the Proposed Framework**

To block malicious user’s requests to access services, usually, servers are equipped and rely on blacklists. However, nowadays, these blacklists are becoming very easier to bypass by the attackers as they are opting for new and unconventional attack vectors. DS is designed in such a way that it uses these blacklists along with traps which enhance server’s security. Furthermore, conventional security measures usually handle the majority of user requests as harmless unless they are listed in firewall blacklist. However, DS enables security administrators to utilize deception whenever they are not sure that the user request is genuine or malicious. DS can differentiate between users and treat them differently. It can treat users in three separate categories i.e. harmless, suspicious and malicious. In general, DS is capable to utilize different traps, which enables security administrators to identify the user is harmless or malicious. Unless users do not interact with these traps, they move to the next category i.e. the suspicious category. In next step, DS is will send some active-deceptive- responses (ADR) and whenever attackers respond to that they can be distinguished and evaluated to the category of malicious users shown in (Figure 8).

![Figure 8. DS User Categorization.](image)

The key advantage of DS is that it is designed such that unlike in the case of honeypots, attackers do not directly interact with it. Also, it not runs an exact replica of production servers and services to lure adversaries to interact with it otherwise it can be compromised and used as entry point to launch a widespread attack against resources. In addition, unlike the conventional use of server honeypots, DS does not add administrator’s overhead to patch and fine-tune another set of fake
systems. DS does not keep and maintain a replica of targeted resources either. Moreover, DS employees deceptive traps which can be helpful in reducing the number of false positives as well. The said functionality of our proposed DS is very helpful to system administrator and defenders, as they will no longer require creating and deploying clone of the entire production system.

**Summary**

In this paper, we presented a structured framework in order to demonstrate that how deception can be used and applied to defenses of an application server environment in order to detect attacks and threats. A proof of concept implementation and performance analysis has also been exhibited. In last part of the paper, we also described the significance of the proposed structure and its distinctive features versus the conventional use of honeypots, honey nets for attacks/threats detection.

**References**


