POSTER: Evaluating Reflective Deception as a Malware Mitigation Strategy

Thomas Shaw
The University of Tulsa
thomas-shaw@utulsa.edu

James Arrowood
Haystack Security, LLC
jim.arrowood@haystacksecurity.com

Michael Kvasnicka
The University of Tulsa
ttk164@utulsa.edu

Shay Taylor
The University of Tulsa
sat763@utulsa.edu

Kyle Cook
The University of Tulsa
kac330@utulsa.edu

John Hale*
The University of Tulsa
john-hale@utulsa.edu

CCS CONCEPTS

• **Security and privacy → Malware and its mitigation**;

KEYWORDS

Deception Technology; Reflective Deception; Malware

ACM Reference Format:
Thomas Shaw, James Arrowood, Michael Kvasnicka, Shay Taylor, Kyle Cook, and John Hale. 2017. POSTER: Evaluating Reflective Deception as a Malware Mitigation Strategy. In *Proceedings of CCS '17, Dallas, TX, USA, October 30-November 3, 2017,* 3 pages.
https://doi.org/10.1145/3133956.3138833

1 INTRODUCTION

Deception is a defensive technique that influences attacker behavior through misinformation or concealment of information [5, 8, 11, 13]. It may be used to forestall an attack, lead an attacker away from a target, and learn about an attacker's motives and capabilities.

This poster describes a defensive deception technique – Reflective Deception – along with an experimentation platform to evaluate its efficacy and performance in mitigating malware. Techniques such as this hold promise for extending the role of deception as a highly integrated cyber defense strategy. Moreover, the platform presented here constitutes a blueprint for the safe and systematic assessment of deception-based malware mitigation strategies.

2 REFLECTIVE DECEPTION

Reflective Deception blends traits from two established deception techniques – honeypots and interdiction. Honeypots expose vulnerable systems to attackers, enticing them to reveal their intent and tactics [3, 10, 12]. A honeypot requires an "operational cover story" sensitive to the context of surrounding system elements. The services it exposes should blend in with neighboring systems, while offering an attractive target to attackers. One benefit of honeypots (and defensive deception techniques, in general) is that they increase the workload of an attacker.

Deception has also played a role in the protection of digital content online. File spoofing technologies emerged shortly after the turn of the century, flooding P2P networks with decoy media [4, 7]. Interdiction based on file spoofing relies on: (1) fabricating decoys with meta data and file characteristics identical to authentic source media, and (2) the ability to saturate a target network with a robust population of decoys. The principal objectives in such a strategy are to conceal the authentic media, while increasing the burden and work required by the attacker to locate it.

Several efforts promote the use of deception to disrupt various phases of what has become known as the intrusion or cyber "kill chain" [1, 3, 9]. Deception has assisted with manipulating an attacker's C2 infrastructure via DNS redirection and by manipulating the actions on objectives phase through honeypots. Other work extends deceptive capabilities in this phase by deceiving the attacker as they interact with the operating system [6].

However, many of these applications require detecting an intruder before the deception can be deployed. Reflective Deception is a variant of deception with no such constraint [2]. A core tenet of Reflective Deception is to manufacture and amplify decoy responses to malware before it is even detected, thereby overwhelming an attacker with spurious feedback. For example, when combating malware distributed through attachments and links, a Reflective Deception strategy would be to open and execute the malware in a secure sandbox a multitude of times. This creates the illusion of dozens, hundreds, or potentially thousands of malware infected systems that call back to the attacker to produce volumes of command and control channels as illustrated in Figure 1. During this time, if a user genuinely falls prey to the malware, their callback traffic is buried within the decoy callbacks distributed over time, thus protecting the infected user as seen in Figure 2.

3 EXPERIMENTATION PLATFORM

The experimentation platform we propose comprises: (i) a physical infrastructure, (ii) a virtual infrastructure, (iii) a victim domain, and (iv) an attacker domain (Figure 3). This platform provides a secure, agile and highly automated solution for exploring the properties and performance of deception and related malware mitigation strategies. It can be deployed and operated in a manner completely disconnected from the Internet or with extremely limited and tightly controlled access to it.

*To Whom Correspondence Should Be Addressed

CCS '17, October 30-November 3, 2017, Dallas, TX, USA

ACM ISBN 978-1-4503-4946-8/17/10.
https://doi.org/10.1145/3133956.3138833

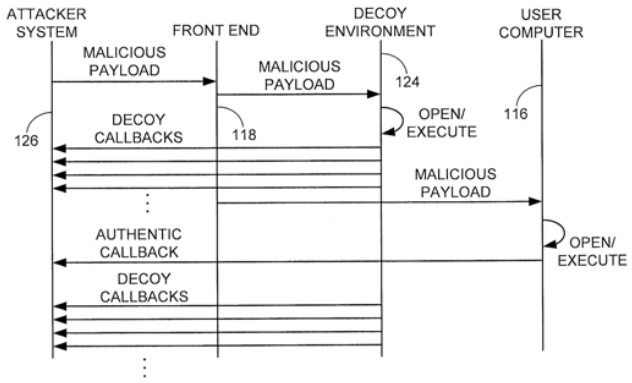

Figure 1: Reflective Deception Callbacks.

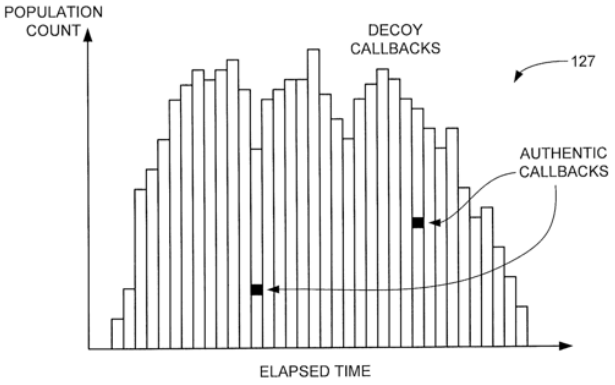

Figure 2: Temporal Distribution of Decoy Callbacks.

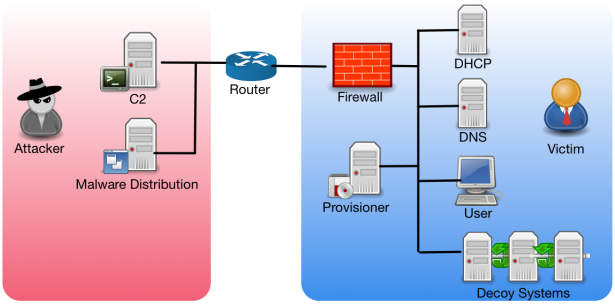

Figure 3: Reflective Deception Experimentation Platform.

3.1 Physical Infrastructure

The physical infrastructure constitutes hardware that hosts the virtual infrastructure. A minimal configuration accommodates one attacker machine, two infrastructure machines, and three victim machines. The physical infrastructure should be capable of running a hypervisor to support multiple internal networks and machines with multiple network interface cards (NICs). (Most virtualization clusters should be able to handle these minimum requirements.) A reference solution has been fielded within a VMWare Cluster at the University of Tulsa. This cluster consists of 18 physical machines totaling 128 cores (300 GHz of aggregate processing power), 464 GBs of RAM, and 8.7 TBs of storage. VMware's VCenter seamlessly balances virtual machines across the physical systems.

3.2 Virtual Infrastructure

The virtual infrastructure consists of support systems not participating as attacker or victim. These include a router, firewall, DNS server, DHCP server, and the provisioning and deployment servers.

3.2.1 Router and Firewall. The router is the backbone of the virtual infrastructure, linking victim and attacker domains. Placing the attacker and victims on separate networks facilitates monitoring all traffic passing between the two. Installing a firewall creates additional options for logging and increased realism. The firewall can be used to watch for connections and block certain traffic to more faithfully represent a real-world environment.

The router in our solution is a VM running FreeBSD with two NICs to bridge two separate VMware virtual switches – one for the attacker domain and one for the victim domain. This router also doubles as the firewall. It only has access to the internal networks, with no outside connection to ensure the malware cannot escape the controlled environment. Both virtual networks are NAT'd, and require manual port-forwarding. This increases the complexity of configuring an attack, but ensures that the experimenter has full knowledge of the requirements of the attack and the environment.

3.2.2 Provisioning and Deployment Server. The provisioning and deployment machine is the most complex part of the infrastructure due to scalability requirements. There are several provisioning options, including scripting the duplication of VM templates and re-imaging existing VMs through a provisioning server. All victim machines must tie into this deployment server, which distributes the malware. It must accept an executable to be pulled and executed by all VMs. After an experiment, these machines must be destroyed or re-imaged to prevent contamination across experiments.

A local host executes a PowerShell/PowerCLI script connected to vCenter. It takes arguments to configure the experiment and set names for cloned decoys. A Windows VM acts as a template to create a snapshot and use VMWare's Linked Clones to produce a volume of decoys. The script then downloads the malware from the Malware Distribution Server, and runs PSExec to execute it across all decoys. The script waits a fixed period of time, then removes the VMs and the machines from the domain, erasing their DNS records.

3.2.3 DHCP AND DNS Servers. The DHCP server mainly applies to the victim domain, but is optional on the attacker side. It promotes scalable automated automated deployment. While static addresses can be assigned to victims, it is safer to assign them dynamically.

The role of DNS depends on the configuration of the provisioning and deployment server. Some require a fully qualified domain name, and thus internal DNS. The DNS server can also host an internal email server to test automated email and attachment scanning. The DNS server supports an intranet for the victim domain, and should not be publicly accessible. For simplicity, attacker domain machines in our solution have manually set static IP addresses.

3.3 Victim Domain

The victim domain comprises systems that are directly attacked or are used in the Reflective Deception response. This domain fielded by the provisioning and deployment server with a few static machines acting as legitimate targets of an attack. All other victim machines (decoy clones) are deployed as needed.

Incorporating diversity into the decoy population improves the efficacy of the Reflective Deception response. *Victim Masquerading* techniques such as synthetic web browsing, mouse movement and forged credentials can increase the amount of time it takes for an attacker to discern whether a system is a viable target. Automated credential generation can also be used to confuse credential harvesters designed to scrape credentials passing through the machine, such as those for bank accounts or e-commerce sites. Our primary technique is an automated credential generation tool connected to an internal web address. This site can be quickly themed based on the kind of credential, for example appearing as a banking website or e-commerce site. With the internal DNS server, these sites can be resolved to real, external domain names, further confusing an automated credential harvester (e.g. amazon.com).

3.4 Attacker Domain

The attacker domain only requires a few machines, depending on the malware under test. These include systems to launch attacks, monitor incoming and outgoing traffic, and house command and control operations. A separate host is responsible for the distribution of malware as it acts as a middle-man between domains to prevent the malware from leaking. The attacker domain may incorporate a malicious web server that encompasses these and other functions. In short, the systems in this domain must be configured to support a realistic manifestation of the malware's ecosystem.

3.4.1 Packet Capture Server. To determine whether Reflective Deception counter measure is effective, a packet capture server logs traffic from victim and attacker domains. It sits on the attacker domain, but is configured using port-mirroring to see traffic from the victim domain as well. Port mirroring on a switch copies and forwards data from source to destination ports. The source can be one or more VLANs/ports as long as the source port is not the same as the destination port. The server's interface is set to promiscuous mode to capture traffic intended for other machines. Since the switch sends packets directly to the packet capture's port it has no need for an IP address. Captured traffic is stored in PCap files and processed through a packet analyzer for analysis.

3.4.2 Malware Distribution Server. To prevent malware leaks, an air-gapped malware distribution server (MDS) runs an SSH server and a web server. A PowerCLI script switches the MDS between domains – neither domain is ever connected from anything other than the router/firewall. For matters of convenience, the malware is uploaded to the SSH server by secure copy, and downloaded from the HTTP server. Data can only be uploaded from the attacker domain and only downloaded from the victim domain. This inhibits leaks through the MDS and admits superior access control.

3.4.3 Command and Control Center. The command and control (C2) center hosts remote administration tools (RATs) used to take control of victims. Under a standard firewall ruleset ('block all in' and 'allow all out'), a victim can connect to an attacker, but not vice versa. Thus, malware commonly 'phones home' to the attacker, creating a connection. RATs have the capability of seeing all machines connected to them simultaneously with the ability to manipulate multiple machines at once, giving rise to a botnet.

4 ON-GOING WORK

On-going work targets multiple objectives, including: (i) development of metrics for Reflective Deception, (ii) evaluation and refinement of our experimentation platform, and (iii) evolution of Reflective Deception tactics. A test using Dark Comet as a representative malware specimen has been conducted, with post-analysis efforts underway. The test ran multiple scenarios using decoy volumes ranging from 1 - 100, generating in aggregate 836 MB of PCap files. Plans are in place to upgrade the infrastructure to support larger decoy volumes as analysis on the first data set begins.

The definition of performance and efficiency metrics remains a thorny subject. Ultimately, a constellation of measures will serve best to identify the right balance in employing Reflective Deception. In addition, the development and operation of the experimentation platform has yielded insights about its architecture. Nuances with automation and VM management confounded naive design thinking, while fundamental rules regarding minimal hardware footprint and capabilities emerged. Strategies for coping with physical and virtual limitations have inspired alternative solutions and potential trade-offs in the platform's implementation. Finally, as the experimenter's familiarity with the attacker's viewpoint grows, concepts for enhanced tactics in Reflective Deception are being cultivated.

REFERENCES

[1] M. H. Almeshekah, E. H. Spafford, and M. J. Atallah. Improving security using deception. Technical Report CERIAS Tech Report, 13, Center for Education and Research Information Assurance and Security, Purdue University, 2013.

[2] J. Arrowood. Cyber attack disruption through multiple detonations of received payloads. Technical Report U.S. Patent No. 8,943,594, USPTO, January 27, 2015.

[3] B. Cheswick. An evening with berferd in which a cracker is lured, endured, and studied. In *Proc. Winter USENIX Conference*, 1992.

[4] N. Christin, A. Weigend, and J. Chuang. Content availability, pollution and poisoning in file sharing peer-to-peer networks. In *Proceedings of the 6th ACM conference on Electronic commerce*, pages 68–77, 2005.

[5] F. Cohen. A note on the role of deception in information protection. *Computers & Security*, 17(6):483–506, 1998.

[6] F. Cohen, D. Rogers, and V. Neagoe. Method and apparatus providing deception and/or altered execution of logic in an information system. Technical Report U.S. Patent No. 7,296,274, USPTO, November 13, 2007.

[7] J. Hale and G. Manes. Method to inhibit the identification and retrieval of proprietary media via automated search engines utilized in association with computer compatible communications network. Technical Report U.S. Patent No. 6,732,180, USPTO, May 4, 2004.

[8] K. Heckman. et al. Denial and deception in cyber defense. *Computer*, 48(4):36–44, 2015.

[9] Hutchins, M. E. Cloppert, and R. Amin. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. In *Proceedings of the 6th International Conference on Information Warfare and Security*, pages 113–125, March 17-18, 2011.

[10] C. Kreibich and J. Crowcroft. Honeycomb: creating intrusion detection signatures using honeypots. *ACM SIGCOMM Computer Communication Review*, 34(1):51–56, 2004.

[11] L. Pingree. Emerging technology analysis: Deception techniques and technologies create security technology business opportunities. Technical report, Gartner Inc., July 16, 2015.

[12] A. Yasinsac and Y. Manzano. Honeytraps, a network forensic tool. In *Sixth Multi-Conference on Systemics, Cybernetics and Informatics*, 2002.

[13] J. Yuill, D. Denning, and F. F. Using deception to hide things from hackers: Processes, principles, and techniques. Technical report, North Carolina State University, 2006.