Enabling Defensive Deception in Distributed System Environments

Nathaniel Soule, Partha Pal,
Shane Clark, Brian Krisler
BBN Technologies
Cambridge, USA
{nsoule, ppal, sclark, bkrisler}@bbn.com

Anthony Macera
United States Air Force Research Laboratory
Rome, USA
anthony.macera.1@us.af.mil

***Abstract*—While attackers have used deception to hide their identities, cause surprise, or mislead victims, defensive use of deception has been limited to honeypots and moving target defenses (MTDs). This has left unexplored a powerful defensive strategy namely, active manipulation of the adversary's decision loop. In contrast to the *passive approach* of honeypots and MTDs, this *active approach* deliberately interacts with the adversary to cause him to think he is succeeding and expend effort in an alternate reality. The work described in this paper took initial steps to realize active defensive deception in the context of distributed systems and built a prototype that creates an alternate reality in which to trap, learn about, and manipulate adversarial actors without affecting normal and legitimate operations. This prototype, called KAGE, employs Software Defined Networking (SDN), and virtualization to create a malleable substrate in which deception can occur. Deception is necessarily context dependent. In the case of KAGE, deception is tied to the mission purpose served by the distributed system being defended, specifically the services running, and the configuration, scale, and complexity of the environment. Consequently, there is no single deception strategy that will fit all system and mission contexts. KAGE therefore presents a framework through which a wide array of deceptions can be composed from component building blocks. This work-in-progress paper introduces the concept of active defensive cyber deception, discusses the early stage KAGE prototype, and introduces some of the challenges intrinsic to enabling defensive deception in distributed environments.**

Keywords—deception; distributed systems; security;

I. Introduction

In the cyber domain deception has primarily been applied for offensive purposes. In that context adversaries attempt to deceive by masking their identities or their code signatures, by launching false attacks as cover for the real attack hiding in the noise, or by deceiving users into clicking malicious links, open malicious documents, etc. In the defensive domain deception has found more limited uses. The state of the art in defensive cyber deception[6] is focused on honeypots [1] that serve as observation, detection, or distraction points, and on Moving Target Defenses (MTDs) that attempt to dynamically modify the defended system to invalidate the attacker's understanding of, or position in the system [2].

These limited uses only scratch the surface of what defensive cyber deception can achieve. The goal of the KAGE effort is to move the defensive deception front forward, expanding into a new area of active defensive deception, where the system seeks to interfere with and disrupt the adversary's decision control loop instead of the traditional *shutdown* response. A detected intrusion therefore does not result in blocking clients from the network or shutting down the host. Instead the adversary's interactions are seamlessly moved into an alternate reality where the defender can safely keep the adversary fruitlessly engaged for long periods of time, drawing attention and fire away from the real targets. Active deception can in addition be used to guide the adversary to defender-selected data or elements of the system, and thus to defender-selected beliefs about the world. This enables manipulation of their control loop which can be employed to prolong a distraction, impart false truths, or to cause the adversary to show their hand (use exploits, reveal their intentions and goals, etc.). These learnings can then be employed to automatically harden the real systems, outside of the alternate reality, before the adversary has had a chance to truly interact with them. These capabilities allow the defender to use the attacker's own energy against him – the more effort an adversary expends in the alternate reality the stronger the defenses can grow.

The remainder of this paper is structured as follows: Section II provides an overview of KAGE's active deception approach, Section III details the KAGE architecture, Section IV presents feasibility focused initial demonstration execution results, and conclusions are drawn in Section V.

II. Overview

KAGE enables *active deception* where the adversary's decision loop is deliberately manipulated by proactive and reactive adaptation of the environment (network and host) as KAGE learns about the capabilities and intents of the client with which it is engaging. KAGE targets all phases of an attack (e.g., reconnaissance, toehold establishment, privilege escalation), and leverages the adversary's progress through the distinct stages to inject altered information about the system and its environment to misguide the adversary.

Distribution Statement "A" (Approved for Public Release, Distribution Unlimited). This material is based upon work supported by the Air Force Research Laboratory under Contract No. FA8750-15-C-0079.

Approved for Public Release; Distribution Unlimited: 88ABW-2015-6193 12/30/2015

978-1-5090-2002-7/16/$31.00 ©2016 IEEE

To accomplish this goal, KAGE employs a dynamic and customizable set of maneuvers aimed at steering the adversary away from critical components, prolonging the engagement, and allowing the defender to learn about them. These maneuvers, unlike honeypots or MTDs, do not just passively observe the adversary or unilaterally adapt the system. Instead, they react in real time in a context-sensitive manner to adversary actions in order to present a false reality that is consistent with the adversary's expectations. To be flexible and general, *deception* cannot be "hard-coded" as that would result in brittle responses that can only handle a particular set of adversary actions. An arsenal of maneuvers deployable in novel compositions and configurations at the will of the defense is thus needed to react and adapt to new situations. We constructed KAGE as a framework in which deceptions can be developed, deployed, and orchestrated enabling this level of adaptation, maneuverability, and dynamism.

We explain KAGE's defensive deception in the context of a network enclave *E* that hosts a number of services that are accessed by clients situated outside *E*. This is a typical distributed system environment commonly found in a corporate or military intranet or in cloud data centers. The objective is to add deception to the existing defense of the services. At the highest level, KAGE hides the real instance of a defended service S, and advertises another instance S' to the potential users. SDN capabilities are used to manipulate service requests addressed to S' such that the requests are actually processed by S, but appear to the requestor to be handled by S'. At the same time, any traffic addressed to S' *that is not a request to the defended service* is delivered only to S'. Reconnaissance will often fall into this category. Existing security mechanisms may detect or prevent these non-service level attacks/interactions. But even if they do not, the impact of the successful attack will affect only S' and not S. Detection of any such reconnaissance or attack will trigger an adaptive maneuver that will redirect the offending client to another instance of the server S'' which will act as the alternate reality playpen where KAGE will continue to engage the adversary and give them a false sense of progress.

The KAGE prototype uses virtualized endpoints to allow for easy creation, monitoring, and flexible manipulation of S, S' and S''. The real endpoint S (which remains invisible to the outside world, but actually provides the service) is referred to as the **REP**, while S' and S'' are virtual endpoint copies referred to as **VEP**s. S' is the VEPU, or VEP for traffic of **U**nknown intent that closely examines all traffic, and the S'' is the VEPM, or VEP for traffic of known **M**alicious intent that corresponds to the core of the alternate reality and is where targeted deceptions play out. These hosts are depicted in Figure 1, along with a KAGE controller that receives sensor inputs, reasons, selects deceptions, and executes those deceptions. An SDN-enabled switch at the entry point(s) to the network enclave provides the network malleability needed to execute certain types of deceptions.

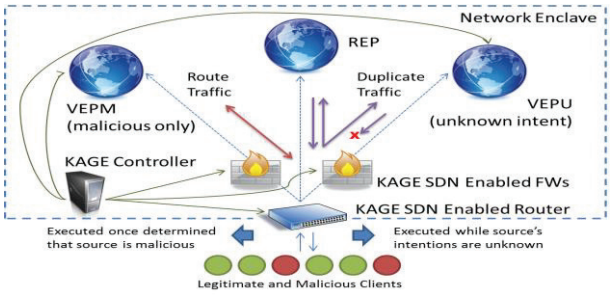

Figure 1. KAGE High Level Elements

Business traffic addressed to the VEPU (since this is the host that is visible to the outside) is duplicated by the SDN switch to both the real service (i.e., the REP that remains hidden from the outside world) and the VEPU. All other traffic addressed to the VEPU, such as network/port scans or traffic not related to business service requests, is delivered only to the VEPU. The VEPU is heavily instrumented and thus can more easily detect malice without putting a burden on the real systems processing business interactions. This traffic splitting isolates the real service, the REP, from all attack vectors except those inherent to the business service itself. Responses from the REP are masqueraded to appear as if they come from the advertised service copy (making the REP all but invisible), and responses from the VEPU (to business service requests) are dropped before leaving the network. VEPU responses resulting from non-business service interactions (e.g. ping, SSH) are returned as is, and are used as additional observables to monitor. Once the VEPU detects malicious actions, the KAGE controller transitions the actor in question into the alternate reality by redirecting all traffic to the VEPM (while modifying returned traffic to appear to come from the VEPU as normal). Once in the alternate reality, targeted deceptions occur based on the adversary's actions. For example, if KAGE detects that the adversary is attempting to execute a SQL injection attack, it can intercept the request, and send a false response that appears to be coming from the web server serving the attacked application, indicating that the SQL injection worked, but the table name used by the adversary was incorrect. This will cause the adversary to waste time on further refinements of their attack, and to reveal their goals (e.g. exfiltrating data, inflicting damage). KAGE relies on detection to trigger movement of an adversary to the alternate reality. It provides infrastructure that supports stealthy observation, but as a framework relies on integration of existing sensors to provide most detection capabilities.

III. ARCHITECTURE

KAGE assumes that anticipating every deception need in advance is infeasible and building narrowly scoped deceptions will lead to a brittle solution (where unanticipated adversary behavior can lead to deception failure). KAGE therefore provides a framework for effecting adaptive behavior using relatively low-level building blocks as depicted in Figure 2. This compositional approach offers a platform on which many deceptions can be easily constructed and connected.

The building blocks can come from existing network and system services and defenses, or be purpose built for KAGE. Each building block can provide sensing and/or actuation capabilities. In the context of deception, actuators are used to

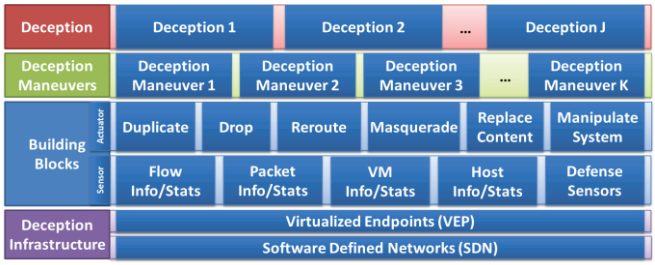

Figure 2. KAGE Layered Component Design

create a malleable world in which elements of reality can be hidden, distorted, fabricated, etc. Most of our current building blocks are built upon two infrastructural layers, virtualization and SDN, which are described in detail in Sections A and B. The sensing building blocks are used to build a world model that describes what KAGE knows about the adversary in order to tailor targeted deceptions at them. The KAGE controller builds this model, analyzes it in order to select deceptions, and composes building blocks into executable deceptions. The controller is described in detail in Section C.

A. Software Defined Networking

KAGE's network level sensing and actuation relies on a malleable substrate that can be easily programmed. The basic construct described in Figure 1 depends on the ability to duplicate, redirect, block, and masquerade network traffic. All of these actuations are possible with traditional networking, but require complex and vendor-specific implementations. On the other hand, SDN enables these capabilities in a robust manner using an open API available across many existing network devices (routers, switches, firewalls, etc.). SDN defines an architecture where network control is decoupled from the data plane. By separating the control and data planes, networks become dynamically programmable, allowing for on-the-fly altering of routes and traffic patterns. KAGE uses a popular SDN framework called OpenFlow as the network substrate. By using OpenFlow, KAGE can perform real-time network reprogramming, in reaction to observed behaviors in need of deceptive action. KAGE uses the Floodlight [5] SDN controller to manage OpenFlow. Floodlight provides a plugin-based architecture that simplifies the bridge between KAGE and the network-level building blocks. The KAGE Floodlight plugins provide Representational State Transfer (REST) interfaces that expose a set of parameter-based HTTP calls for implementing network deceptions. These higher-level abstractions (drop, duplicate, pose, surrogate, new network entity notification, etc.) map internally to multiple network-level flows that allow the KAGE controller to easily monitor and alter the flow of network traffic at the packet level.

B. Virtualization

KAGE relies on virtualization to manage the set of endpoints (REP, VEPU, etc.) necessary to present a complete deception environment, and Virtual Machine Introspection (VMI) to implement application/service monitoring and management within the VEP containers. The KAGE prototype can thus quickly start, stop, or duplicate endpoints as necessary to respond to failures or evolving needs and can implement VMI plugins that allow the KAGE controller to observe user interaction patterns. These plugins have a low probability of detection by the adversary, as all monitoring code runs in the hypervisor as opposed to within the monitored guest VM.

The KAGE prototype uses StackDB[3] as its VMI tool. StackDB is an open source "stackable debugger" that allows the host operating system to probe processes in guest VMs at multiple layers of abstraction. For example, a StackDB policy could alert the KAGE controller to any instance of the sys_execve system call initiated by a PHP process in the guest. The alert would include all of the arguments to the system call, such as the process being executed, and could also show the lines of source code from the running PHP script that led to the call. These details are key to assessing adversary capabilities and in crafting targeted deceptive responses.

C. Controller

Central to the KAGE framework is the KAGE controller. It acts as the intelligence center of KAGE and weaves all other components together. The controller accepts sensor inputs to build its model of the current state of the world, and analyzes this model (or subsets of it) in order to execute deception maneuvers that are constructed as compositions of individual actuators. The model itself is structured as a graph and is stored in an instance of the graph storage and query engine Neo4j. The graph can contain arbitrary structure and is built automatically from the sensor events received by KAGE. Here JSON events from sensors are converted to a graph representation (e.g., a JSON message describing a client, an attack, and the fact that the client initiated the attack would result in graph nodes for the client and the attack, and a named relationship between them). Neo4j provides a rich query language that allows for efficient queries over graph content and structure. This allows for KAGE state analyzers to quickly detect certain state patterns and initiate an appropriate response (e.g., detecting a client is performing web page enumeration reconnaissance, and modifying responses to portray a false web site structure). KAGE currently employs pattern matching rules to derive the next steps of the deception from the graph, but we plan to explore case based reasoning, and game theoretic approaches as future enhancements.

IV. Feasibility Demonstration

The purpose of this early stage KAGE prototype was to bear out the difficult problems, and demonstrate the feasibility of an active deception approach. We have thus yet to conduct an in-depth quantitative evaluation of the current prototype, e.g. to assess its response time under a realistic network load. As a preliminary feasibility test, we have implemented and successfully executed multiple variations of a demonstration scenario that integrates and exercises many of the major KAGE components to orchestrate a brief deception campaign. Figure 3 describes the demonstration workflow at a high level. The demonstration makes use of two KAGE plugins: a port scan detector acting as a sensor, and a SQL injection detector [4] adapted to work as both a KAGE sensor and actuator.

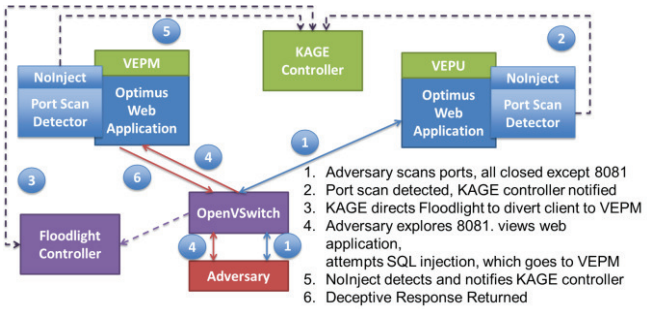

Figure 3. Feasibility Evaluation Scenario

The workflow starts with an unknown malicious client attempting to port scan the advertised endpoint. As it would for any connection from a previously unidentified client, the SDN switch duplicates the client's traffic to the VEPU VM. The VEPU VM runs the port scan detector that observes the scan coming from the adversary and alerts the KAGE controller that a scan is underway from the source address.

The KAGE controller parses the alert from the sensor plugin, searches for other alerts concerning the same client in its database, and applies its reasoning rules to determine a course of action. In this case, the controller decides to redirect all traffic from the adversary to the VEPM container (moving the actor into the "alternate reality"), because a port scan is an indicator of malicious intent. The controller executes its alternate reality transition plugin, which is composed of network level deception building blocks. These building blocks command the SDN switch to transparently reroute all traffic from the adversary to the VEPM container, and to cause the VEPM to masquerade as the VEPU to ensure the adversary remains unaware of the switch. The SDN switch implements this change immediately. We have observed this to happen before the port scan even completes, resulting the remaining part of the port scan to execute against the VEPM.

Upon completion of the port scan, the adversary observes that TCP port 8081 is open, and attempts to connect to the REP web service (with which the adversary still believes they are interacting) with a web browser. In reality, the adversary successfully connects to an instrumented copy of the service in the VEPM VM. Upon seeing a login page, the adversary assumes that he is interacting with the real web service and attempts to mount a SQL injection attack against that page.

The VEPM web service is instrumented with a SQL injection detector that identifies the SQL injection attack before it reaches the database and alerts the KAGE controller of the attack, including the content of the malicious query in the alert. The KAGE controller applies its reasoning rules which are themselves composed of analysis plugins. A SQL injection analysis plugin inspects the malicious query and extracts the table names from the query. The plugin then uses the table name information to construct a convincing deceptive response, e.g., "*table KAGE.users does not exist*". KAGE is also employing another plugin that allows the plain text response to be wrapped in HTML corresponding to the error message styling and wording of any of a set of common web servers, further misleading the adversary.

V. Conclusions

Defensive cyber deception shows great promise as a potent tool in the defenders toolbox. The KAGE effort has thus far produced an extensible and demonstrable prototype that shows the feasibility and potential effectiveness of 1) actively targeting, disrupting, and manipulating an adversary's decision control loop, 2) creating an alternate reality and seamlessly transitioning actors deeper into this alternate reality as malicious acts are detected, 3) employing a malleable SDN for deceptive purposes, 4) a building-block based approach to constructing cyber deceptions, and 5) constructing a pluggable framework for quick and easy addition or removal of arbitrary maneuvers. These results make great strides towards a robust defensive cyber deception capability, but are only the first steps on this path. The costs and benefits of deception in defense needs to be rigorously evaluated before practical use, and there are numerous R&D challenges that need further work. Below we outline a few.

Both legitimate and adversarial clients have timing expectations regarding interactions with networks, hosts, and services. Deviation from expectations can not only disrupt legitimate activities, but can provide indicators to attackers that they are being deceived. Separation of realities into real and alternate, while providing isolation and freedom of maneuvering, also introduce challenges in distributed consistency and seamless transitioning. The distributed nature of KAGE's building blocks adds complexity, in terms of dynamic distributed composition, and with respect to the timing challenges mentioned above. Future work will continue to prove the benefits of defensive deception while addressing these key challenges. Finally, defensive cyber deception could potentially erase the advantage adversaries currently hold over defenders. If the defender can be placed in a position of control - intentionally directing adversary energy to times and places of the defender's choosing, and using that energy against the attacker himself, an important step will have been taken towards correcting the current imbalance between defense and offense in cyber-security.

References

[1] Provos, Niels. "A Virtual Honeypot Framework." USENIX Security Symposium. Vol. 173. 2004.

[2] Okhravi, Hamed, M. A. Rabe, T. J. Mayberry, W. G. Leonard, T. R. Hobson, D. Bigelow, and W. W. Streilein. Survey of cyber moving target techniques. No. MIT/LL-TR-1166. MASSACHUSETTS INST OF TECH LEXINGTON LINCOLN LAB, 2

[3] Johnson, David, Mike Hibler, and Eric Eide. "Composable multi-level debugging with Stackdb." ACM SIGPLAN Notices. Vol. 49. No. 7. ACM, 2014.

[4] Swanhart, Justin. Detecting (and even preventing) SQL Injection. Percona Live, April 2013. URL: https://www.percona.com/live/mysql-conference-2013/sites/default/files/slides/noinject-130424183128-phpapp02.pdf

[5] Project Floodlight. Floodlight OpenFlow Controller. URL: http://www.projectfloodlight.org/floodlight/, accessed December 2015.

[6] K.E. Heckman et al, "Denial and Deception in Cyber Defense", Computer, vol 48, no 4, pp 36-44, Apr. 2015