A Security Pattern for Honeypots

EDUARDO B. FERNANDEZ, Florida Atlantic University
ELIAS BOU-HARB, Florida Atlantic University
VIRGINIA M. ROMERO, Florida Atlantic University

We present a security pattern for honeypots. The intention of this pattern is to set up a decoy system to attract attacks generated by intruders in order to understand, study and deflect these attacks. The study of the behavior, mechanisms and strategies used will prepare us for future attacks. This pattern classifies these attacks and records them in a catalog for further recognition and classification of the attacks

Categories and Subject Descriptors: D.2.11 [**Software Engineering**]: Software Architectures - Patterns

Additional Key Words and Phrases: Honeypot, attack detection, security patterns, intrusion detection

ACM Reference Format:

Fernandez E.B., Bou-Harb E. and Romero V.M. 2017. A Security Pattern for Honeypots. HILLSIDE Proc. Of Asian Conf. on Pattern Lang. of Prog. 6th (Mar 2017), 5 pages.

1. INTRODUCTION

We present a security pattern for honeypots. The intention of this pattern is to set up a decoy system to attract attacks generated by intruders in order to understand, study and deflect these attacks. The study of the behavior, mechanisms and strategies used will prepare us for future attacks. The study of the behavior, mechanisms and strategies used will prepare us for future attacks. This pattern classifies these attacks and records them in a catalog for further recognition and classification of the attacks.

This pattern adds to the collection of security patterns described in [Fernandez 2013]; more specifically it can be considered an addition to its Chapter 10, it is a variety of Intrusion Detection System. Our audience are software developers and security designers as well as researchers and students of security.

.

2. A SECURITY PATTERN FOR HONEYPOTS

Intent

Attract attacks generated by intruders to understand and capture their intentions, mechanisms, strategies and potential for success in order to prepare better to defend against these attacks. Attackers are lured by mimicking internal, typically sensitive services.

Context

Systems and corporate networks connected to the Internet.

.

Problem

On one hand, corporate networks and their corresponding services are constantly being attacked. When attackers bypass the external defense mechanisms, they can easily target any internal service. There is a critical need to defend against such attempts to protect the real operational services and to alert against an infiltrated attacker [Even 2017].

Author's address: E.B.Fernandez, 777 Glades Road, Boca Raton, FL, 33431. Dept. of Computer and Elect. Eng. And Computer Science; email: fernande@fau.edu; E.Bou-Harb, 777 Glades Road, Boca Raton, FL, 33431. Dept. of Computer and Elect. Eng. And Computer Science; email: ebouharb@fau.edu; V.M.Romero, 777 Glades Road, Boca Raton, FL, 33431. Dept. of Computer and Elect. Eng. And Computer Science; email: vromero@fau.edu

On the other hand, it is often hard to derive the modus operandi (attack signatures) for novel and complex attacks. This is especially factual when operating in less explored or constrained environments such as power grids or manufacturing plants. We need a way to detect those attacks in real time.

Forces

- Extensibility and Flexibility—ability to promptly add or modify certain features and/or protocols to reflect new mimicked services, which help lure more attackers or attract attackers infiltrating different realms.
- Usability—the software should be easy to install, configure, manage and its reporting should be comprehensive and actionable.
- Legitimacy as network assets/resource—the software should closely mimic the services desired to be protected; from an attacker perspective, the honeypot should appear as a typical legitimate operational service.
- Mingling Factor—the detection device should communicate with other assets on the network to contribute to its deception characteristics.
- Exploitation resilience—the detection device should lure attackers but prevent its components from complete system exploitation.
- Alerting and Logging—any connection attempt must produce an alert and should be logged.
- Reporting: a honeypot must provide clear and actionable reports of the detected attacks in an appropriate, non-proprietary format.

Solution

Place an easy and appealing target, a honeypot, to attract attacks and lure intruders, so we can assess (i.e., infer, characterize, mitigate, and attribute) them and prepare ways to halt them.

A honeypot, when configured with services operating in such realms, could be used to capture significant attack strategies, intentions and mechanisms to aid in deriving effective attack signatures that can be used for detection, mitigation, and attribution, as well as to indicate possible defenses.

Structure

Figure 1 shows the class diagram of this pattern. The Honeypot includes several Service Proxies that represent real Services provided by the local Server. When the Attacker attempts to access any of these proxies the Attack Detector captures its Signature which is placed in the Attack Signature Catalog. The Attack Detector activates an Alert for the system administrator and keeps a Log of each event received during the attack. Reports are generated at periodic intervals from the Log.

Dynamics

Use cases for the Honeypot include setting up the catalog of signatures and adding proxies. The dynamics for those cases are fairly simple and not shown. More interesting is the use case for detecting an attack, shown in Figure 2. When the Attacker tries to access a service through its Proxy, the Attack detector captures its signature, registers it in the Log, and creates an Alert. This signature can be used later to recognize another instance of the attack and to find countermeasures for the attack.

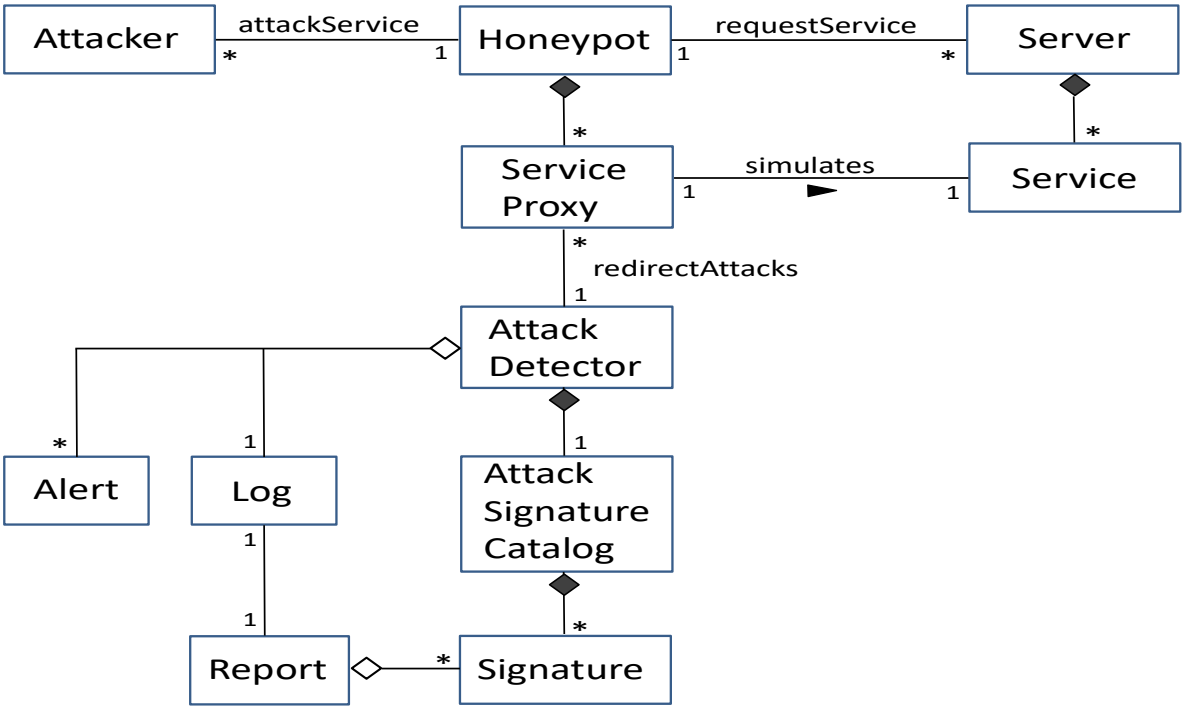

Figure 1. Class diagram of the Honeypot Pattern

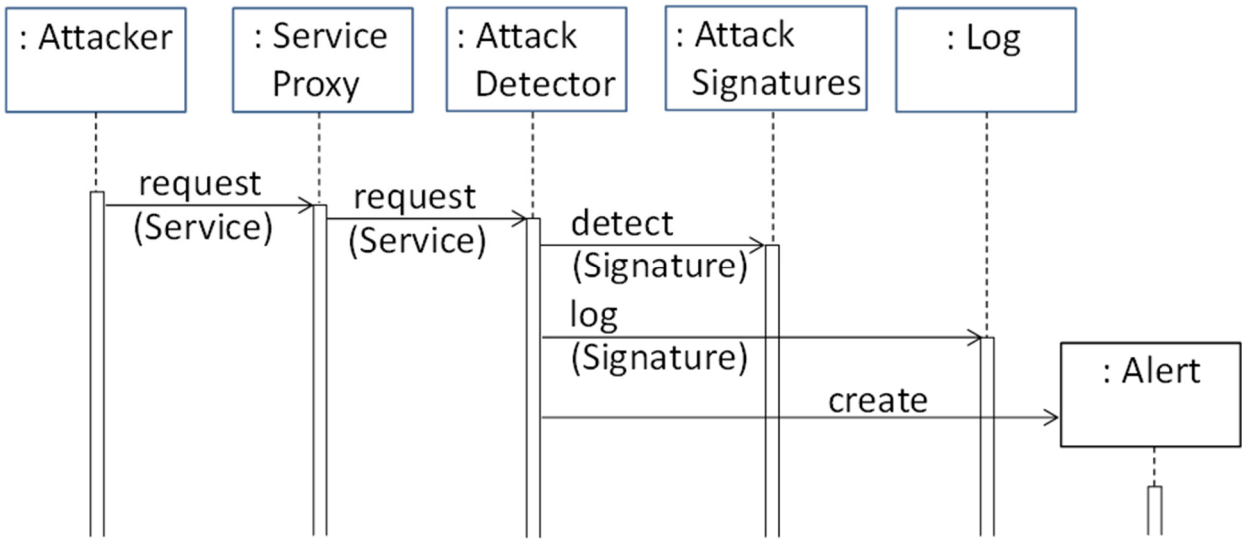

Figure 2. Sequence diagram for use case Detect Attack

Implementation

You can configure and deploy honeypots internally, as an additional layer of protection. Investigate how and where past attacks have succeeded. Generate threat models from past and novel attacks and try to characterize future possible attacks. Determine where you have gaps in your current detection methodologies and deploy tailored honeypots to meditate those gaps [Even 2017, Grimes 2016].

In the light of these facts, it becomes imperative to design intelligent/adaptive honeypots which can proactively adapt to the complex and diverse traffic generated from attackers along with efficient monitoring, logging and isolation

capabilities. To this end, one implementation option is to leverage the Honeymix architecture [Han et al. 2016]. Honeymix addressed the above issues through the use of Software Defined Networks (SDN) and Network Function Virtualization (NFV). In particular, the implementation takes advantage of SDN to (1) centrally analyze the relations of various data obtained from the virtualized networks to learn attackers' patterns and/or behaviors, (2) dynamically balance incoming traffic based on its characteristics (i.e., application-aware packet processing in various realms) and (3) make a decision accordingly to avoid the detection of the presence of a honeynet and transition its interaction level to adaptively response to the attackers. Another implementation option could be the SIPHON architecture [Guarnizo et al. 2017] as a scalable high-interaction honeypot platform that is specifically designed for CPS/IoT devices.

Build attack signatures that aim at reducing both, false positives and false negatives. This is especially important when deployed in non-traditional environments such as cyber-physical systems (i.e., power stations, manufacturing plans, cargo terminals, etc.).

Honey Pots are typically deployed inside a firewall or inside the DMZ. They should appear to be standard services to avoid the attacker's suspicions. The idea is that the attacker returns so we can collect more information about his probing [Even 2017].

Known uses

- KFSensor is pre-configured to monitor all TCP and UDP ports, along with ICMP [KFSensor]. It is also configured with the emulation of common services. KFSensor is considered to be one of the best in [Grimes 2010].

- Honeyd is an open source honeypot [Provos].

- OpenCanary [Canary] is another open source honeypot, which exceeds Honeyd by its ease of use and configuration, and actionable reporting features.

- Conpot [Conpot] is an open source honeypot specifically tailored towards industrial control systems.

Consequences

- Extensibility and Flexibility—a honeypot will effectively be able to mimic diverse and tailored services to increase the detection rate.
- Usability—a well-thought interface can achieve usability.
- Legitimacy as network assets/resource—the honeypot will increase its detection rate by optimizing its deception characteristics.
- Mingling Factor—the honeypot will lure more attackers, which also increases its detection and protection rate.
- Exploitation resilience—the honeypot will be able to handle more attackers, thus increasing its protection rate.
- Alerting and Logging—the honeypot will be able to effectively alert concerned parties about an ongoing attack for effective mitigation and use logged data for future planning.
- Reporting: the honeypot will be able to generate cyber threat intelligence that can effectively be used in long-term analysis and actionable thwarting of attacks.

Related patterns

- Intrusion Detection Systems (IDS) [Fernandez 2013]. IDSs detect and report attacks in real time by employing various inference engines, which could rely on signatures, behavioral analysis or a combination of those. A honeypot, in contrast, adopts the concept of detection by deception, which adds a complementary layer to the defense strategy.
- The Log is a special case of the Security Logger/Auditor [Fernandez 2013]. A Logger keeps track of user's actions in order to determine who did what and when. It also provides controlled access to its records for Audit purposes.

- The Service Proxy is a variety of the Proxy pattern of [Gamma et al. 1994]. A Proxy represents an object and controls access to it.

Acknowledgements

We thank our shepherd, Shinpei Hayashi, for his valuable comments that significantly improved our paper. The participants in Track A of the Asian PLoP conference of 2017 provided valuable comments.

REFERENCES

Canary. "Bring back the Honeypots". http://thinkst.com/stuff/bh2015/thinkst_BH_2015_notes.pdf

Conpot. " ICS honeypot". http://thinkst.com/stuff/bh2015/thinkst_BH_2015_notes.pdf

L.R.Even, "Honey Pot systems explained", https://www.sans.org/security-resources/idfaq/what-is-a-honeypot/1/9

E.B.Fernandez, "Security patterns in practice: Building secure architectures using software patterns", Wiley Series on Software Design Patterns, 2013.

E. Gamma, R. Helm,R. Johnson, and J. Vlissides, Design patterns –Elements of reusable object-oriented software, Addison-Wesley 1994.

R.A.Grimes, "Intrusion detection honeypots simplify network security", InfoWorld, Nov. 17, 2010, http://www.infoworld.com/article/2624430/intrusion-detection/intrusion-detection-honeypots-simplify-network-security.html (last accessed Jan. 03, 2017)

R.A.Grimes, "Spread honeypots over your defense plan", InfoWorld, March 29, 2016 http://www.infoworld.com/search?query=honeypots&contentType=article%2Cresource (last accessed Jan. 03, 2017)

Juan Guarnizo, Amit Tambe, Suman Sankar Bunia, Mart´ın Ochoa, Nils Tippenhauer, Asaf Shabtai, and Yuval Elovici. SIPHON: Towards scalable high-interaction physical honeypots. arXiv preprint arXiv:1701.02446, 2017.

Wonkyu Han, Ziming Zhao, Adam Doup´e, and Gail-Joon Ahn. HoneyMix: Toward sdn-based intelligent honeynet. In Proceedings of the 2016 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization, pages 1–6. ACM, 2016.

KFSensor. KFSensor, Advanced Windows Honeypot System, http://www.keyfocus.net/kfsensor/

Niels Provos, "Honeyd: A Virtual Honeypot Daemon" (Extended Abstract), http://metro.citi.umich.edu/u/provos/papers/honeyd-eabstract.pdf