Modern Honey Network

Shah Manthan Jigneshkumar
Computer Engineering, Vishwakarma Institute of Information Technology
Email: mjshah101@gmail.com, manthan.shah@viit.ac.in

Abstract- The concept of honeypot has been around for a while now. But honeypots are yet to be adopted by network security community as a standard majorly due to its complexity. MHN is an open-source project developed by Threatstream which is aimed at overcoming the risk and complexity of honeypots and integration with tools to analyze and productively use the data or intelligence gathered by such honeypots. MHN is easy to deploy, configure and maintain and it provides integration with third-party apps and ability to add our own honeypots, which could go a long way to making honeypots an industry standard.

Index Terms- MHN, Modern Honey Network, Honeypot, Open-source, Threatstream, Cyber Security

1. INTRODUCTION

The concept of honeypot was first introduced by Clifford Stoll in his book 'The Cuckoo's Egg: Tracking a spy through the maze of computer espionage' in 1989. In 1990 Bill Cheswick of AT&T Bell Laboratories wrote a paper named 'An Evening with Berferd: In which a cracker is lured, endured, and studied'. The concept of honeypot has been realized, applied and grown in the late 20th century as evident by the release of the products like 'Deception Toolkit' in 1997, 'CyberCop Sting' Sin 1998, 'Snort', 'BackOfficer Friendly' and 'NetFacade', in 1998. The formation of the 'HoneyNet Project' led by Lance Spitzner in 1999 truly started the developments and formalization of concepts of honeypots.

Today in 2016, it can be concluded that the changes that significantly influenced the advances in honeypot research are adoption of new types of network applications, advances in networking hardware, geographical diversity in internet user population and growth in underground attackers' communities. Today we have already developed advanced honeypots like 'BitSaucer', 'Honeyware', 'Argos', 'HoneyBow', 'PhoneyBots' and advanced concepts like 'Shadow Honeypots' or 'Super Honeypots'.

But even after 25 years of private and crowd-sourced research and developments, honeypots have failed to receive industry wide adoption. The lack of adoption by the industry and intelligence community can be easily attributed to the complexity of deployment, configuration and maintenance; and underdeveloped or still-developing technologies related to Big Data, Negative Database, Data Mining, Social Engineering and Data Analytics.

1.1 Objectives / Problem Statement

(1) To create or implement a software solution which is easy to deploy and maintain.
(2) To provide tools along with the software solution to analyze and leverage the data gathered.
(3) To overcome the demerits of honeypots and make honeypots widely acceptable by the industry and security/intelligence community.

2. HONEYPOT

A honeypot is an offensive security measure. Lance Spitzner, the founder of the Honeynet Project defines honeypot as "an information system resource whose value lies in unauthorized or illicit use of that resource". According to Lance, a honeypot is "a resource whose value is in being probed, attacked or compromised". Honeypot is a trap set to detect, deflect or in some other manner counteract attempts at unauthorized use of information systems. As such honeypot is a security device that is designed to lure malicious activity towards itself. Capturing such malicious activity allows for studying it to understand the operations and motivation of attackers and the black hat community, and subsequently lead to betterment of security of cyber information resources.

2.1 Classification of honeypots

Honeypots can be classified into various categories based upon different object of classification scheme. Such classification based upon various criteria is given below.

(1) Interaction Level
Interaction level describes if the exposed functionality of a honeypot is limited.

- Low interaction
  Low-interaction honeypots are easier to deploy and maintain. Low-interaction honeypots work by emulating certain selected services and operating systems and have limited interaction, thus decreasing the risk of information exposure.

- High Interaction

  High-interaction honeypots are very complex, with real operating systems and applications. In this type of honeypots, attacker's interaction is not limited and no restrictions are imposed on the attack behavior or attack vector which allows honeypot administrators to capture and log extensive details about the full extent of the attacker's methods, ability and motivation.

(2) Data Capture
Data capture describes the data that a honeypot is able to capture from an attack point of view.

- Events
- Attacks
- Intrusions
- None

(3) Containment Strategy
Containment Strategy relates to the measures a honeypot takes to defend against malicious activity spreading from it.

- Block
- Defuse
- Slow Down
- None

(4) Distribution Appearance
Distribution appearance relates to how the honeypot system is perceived from outside. It also relates to simulation or virtual honeypots.

- Distributed
- Stand-alone

(5) Communication Interface
It describes the type of interface that is used by the user to interact with honeypot.

- Network Interface
- Non-network hardware interface
- Software API

(6) Role in multi-tier architecture
This category describes the role played by honeypot in multi-tier architecture.

- Server-side

  This type of honeypot acts as a server. It does not provide or simulate and production services. It passively waits for requests from clients
- Client-side

  This type of honeypot acts as a client. It provides or simulates various production services and actively sends requests to servers.

2.2 Merits and Demerits

Merits of a honeypot system are listed below.

(1) Smaller data sets
(2) Reduced false positives
(3) Catching false negatives
(4) Protection against encrypted attacks
(5) Protection from unknown or zero-day attacks.
(6) High flexibility
(7) Minimum resource requirement

Demerits of a honeypot system are,

(1) Risk / Exposure
(2) Limited field of interaction and view
(3) Discovery and fingerprinting

When placed with other security measures like Intrusion Detection Systems (IDS), Intrusion Prevention Systems (IPS), and Firewalls; honeypots can become a highly effective tool against attacks performed by black hat community. Most traditional security measures like IDS, IPS or Firewalls detect and prevent attacks using predefined signatures embedded into them. Such tools cannot protect against Zero-day attacks whose signatures do not exist in their database. Honeypots are designed to protect against all known and unknown attacks. As every activity in a honeypot is by definition unauthorized, it can detect stealthiest attacks and any new or unseen anomalies or attacks stand out. This helps honeypots with reducing the false positives and catching the false negatives.

3. MHN

MHN stands for Modern Honey Network. MHN is an open-source project developed by a prominent cyber security company named Threatstream. The aim behind the project was to make an easily adoptable honeypot software solution motivated by the limited adoption of honeypots as an advisable security solution. MHN is completely free open source software which supports honeypot deployments of both internal and external nature at a large and distributed scale. MHN uses the HPFeeds standard and low-interaction honeypots to keep effectiveness and security at enterprise grade levels. MHN provides full REST API out of the box and from the secure deployment to the aggregation of thousands of events. MHN provides enterprise grade management of various open source honeypot software which are dominantly used in the industry currently. MHN has been designed as a free alternative to the current solutions on the market that can help increase the number of production honeypots. It relies on open source tools such as HPFeeds for collecting sensor data, Mnemosyne for indexing the data into MongoDB, and HoneyMap for real-time visualization of the information.

3.1 MHN Architecture

MHN is designed on three-tier architecture. Various sensors and honeypots are placed at the bottom. The tier above the sensors includes HPFeeds, HPFeeds-Logger, HoneyMap and Mnemosyne. The next tier has MongoDB. The top tier includes the Web App and Rest APIs. The architecture is shown in the figure below.

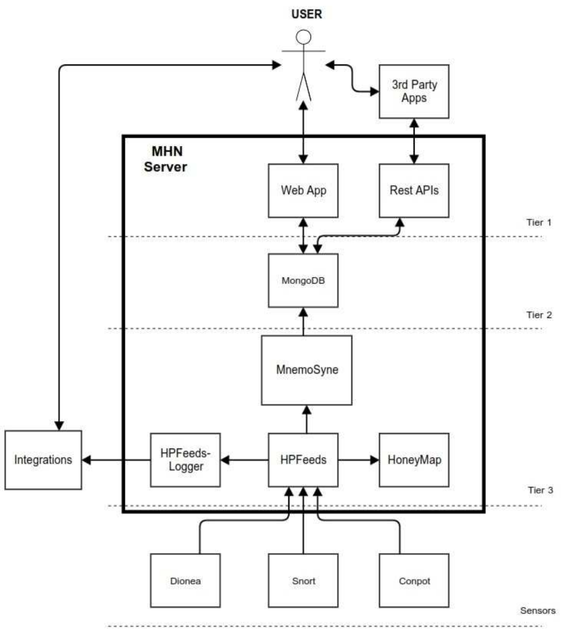


Fig. 1. Architecture of MHN Server

Going from bottom to top, features of each tier are listed below.

(1) Sensors
This layer of architectures involves the sensors which are the data generation points. The purpose of this tier is to make use of multiple sensors deployed in different geographical locations with different configurations and to form a honeynet or a honeypot network. Various sensors that can be deployed easily via MHN include Snort, Suricata, Dionaea, Conpot, Kippo, Amun, Glastopf, Wordpot, ShockPot, p0f and Elastichoney. Along with computer network sensors, MHN also provides the facility to deploy a sensor for Raspberry Pi WSN. On top the already configured sensors listed above, MHN provides the facility to add new sensors very easily.

(2) Tier-3
This layer has HPFeeds at its core. HPFeeds is an open-source tool used to collect data from various sensors deployed at the bottom tier. HPFeeds-logger is used to collate the data collected by HPFeeds and feed the data to third-party app integrations like Splunk, ArcSight and ELK. HoneyMap is used for real-time visualization of data gathered by HPFeeds. Thus this layer is dedicated to data collection and collation. Mnemosyne is used to index the data into MongoDB.

(3) Tier-2
This tier has MongoDB at its core to store, manage and manipulate data. Mnemosyne is used to index the data into MongoDB. Such this layer consists of data management system

(4) Tier-1
This is the top layer of MHN architecture. This layer involves MHN Web App and Rest APIs which the user directly interacts with. The Rest APIs include CEF and STIX support for direct SIEM integration through our Commercial platform developed by Threatstream called 'Optic'.

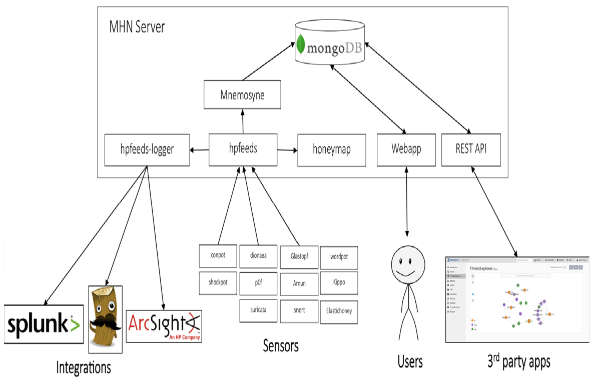


Fig. 2. Architectural Layout of MHN

3.2 Sensors

As mentioned in the MHN architecture, sensors are deployed at the base of MHN server. These sensors are various honeypots, IDS, passive scanners which generate the data about the attack, the attacker and the attack vector. These sensors log every activity in the honeypots and scan them for known malware or virus or worm signatures and detect anomalies. The data generated by these scanners is the raw products that will be collected, collated, indexed, stored and process to provide useful results. Various sensors are available to deploy, pre-configured as a part of MHN Server and Web app. These sensors are listed below.

(1) Dionea
Dionea is an open-source low-interaction honeypot that is designed to capture attack payloads and malware. Dionaea is meant to be a nepenthes successor. Dionea is embedding python as a scripting language. It uses libemu to detect shell codes. Dionea also supports IPv6 and TLS. The ultimate goal of

Dionea is gaining a copy of the malware. Dionaea's intention is to trap malware exploiting vulnerabilities exposed by services offered or simulated to a network.

(2) Snort

Snort is an open source network intrusion prevention system. It was developed by Martin Roesch in 1998. Snort is now being developed by 'Sourcefire' which has been owned by CISCO since 2013. It is capable of performing real-time traffic analysis and packet logging on IP networks. It can perform protocol analysis and content searching/matching. It can be used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes and OS fingerprinting attempts. Snort has been included in the 'Open Source Hall of Fame' by 'InfoWorld' as one of the "greatest open source software of all time".

(3) Conpot

Conpot has been developed as a part of the 'HoneyNet Project'. Conpot is an Industrial Control System honeypot with the goal to collect intelligence about the motives and methods of adversaries targeting industrial control systems. It is a low-interactive server-side honeypot. It is easy to deploy, modify and extend.

(4) Kippo

Kippo is a medium-interaction SSH honeypot. It was written in Python. Kippo was designed to log brute force attacks. Kippo logs the entire shell interaction performed by the attacker. It was written in Python.

(5) Amun

Amun is a low-interaction honeypot. Amun aims at capturing malware in an automated fashion through the use of emulated vulnerabilities.

(6) Glastopf

Glastopf is a Web Application honeypot which was initially developed by Lukas Rist in 2009. Glastopf is a client-side honeypot written in Python. It emulates thousands of vulnerabilities using vulnerability type emulation to gather data from attacks targeting different web applications.

(7) Wordpot

Wordpot is a Wordpress honeypot which detects probes for plugins, themes, timthumb and other common files used to fingerprint a Wordpress installation.

(8) ShockPot

ShockPot is a Web app honeypot created by Threatstream. It is designed to guard against the attacks done by black hat community exploiting vulnerabilities in the Bash remote code namely "ShellShock". This honeypot detects the attacks exploiting ShellShock vulnerability on HTTP port 80 and logs the exploit code and scripts. Such data generated by ShockPot can be easily fed to VirusTotal or Threatstream Optic to extract URLs and files.

(9) ElasticHoney

ElasticSearch is an open-source search server developed by Shay Banon. ElasicHoney is an ElasticSearch Honeypot. It is designed to catch attackers trying to exploit RCE vulnerabilities in ElasticSearch.

(10) P0f

P0f is a passive packet analyzer and a versatile passive OS fingerprinting tool. It was developed in C programming language by Michal Zalewski. P0f is a tool that utilizes an array of sophisticated, purely passive traffic fingerprinting mechanisms to identify the actors behind any incidental TCP/IP communications without interfering with the communication in any way.

(11) Suricata

Suricata is an open-source IDS/IPS developed by Open Information Security Foundation (OISF). Suricata is a rule-based ID/PS engine that utilizes externally developed rule sets to monitor network traffic. It provides alerts to the system administrator when suspicious events occur. Suricata was designed to be compatible with most of the existing network security components. Suricata features unified output functionality. Suricata also has pluggable library options, which are used to accept calls from other applications.

3.3 HPFeeds

HPFeeds was developed under the HoneyNet Project. The HPFeeds data sharing model has since evolved into HPFriends and HoneyMap. HPFriends can be used to choose and share data feeds with friends. HoneyMap is a real-time world map which shows the honeypots deployed and attacks being carried out on them.
HPFeeds was designed as systems that can implement a lightweight authenticated publish/subscribe protocol for exchanging live data feeds. HPFeeds uses a simple wire-format so that feeds can be subscribed using any language. This makes HPFeeds very versatile. HPFeeds supports integration with WireShark. The channel users can choose the serialization format to subscribe to structured data feeds. Thus HPFeeds can gather data from multiple honeypot software, format them and feed them to other software like HPFeeds-logger, HPFriends, HoneyMap, Mnemosyne, and WireShark which can analyze and process the data to produce human readable, useful results.

HPFeeds plays the very important role as a data gathering point for MHN Server. HPFeeds lies at the base of MHN Server. HPFeeds can have multiple inputs from different honeypot software which work as a data generation point. HPFeeds has internal connections to HPFeeds-logger, HoneyMap and Mnemosyne.

3.4 Mnemosyne

Mnemosyne is used as a data normalization tool, which connects HPFeeds with MongoDB.

Mnemosyne has three main objectives:

(1) Provide immutable persistence for HPFeeds.
(2) Normalization of data to enable sensor agnostic analysis.
(3) Expose the normalized data through a RESTful API.

Mnemosyne uses following collections:

(1) Counts
This collection was added specifically for MHN. This collection was designed to speed up aggregation queries. It can be described as pre-computed summaries of counts over time-ranges.

(2) Daily_stats
This is a default collection of Mnemosyne which stores daily summaries statistics across various dimensions of data.

(3) Dork
This default collection stores searches, file paths and associated metadata collected from honeypot. This collection specifically relates to web application honeypots. It serves Dorks collected by Glastopf.

(4) File
This is the default collection used to store files and associated metadata collected from various honeypots.

(5) HPFeed
This default collection stores the raw message sent back from a honeypot in the original format (Usually JSON).

(6) Metadata
This collection has been specifically added by MHN. It is used to store metadata around IP addresses. This collection is used as a pre-computed collection of observations around an attacker IP. This collection is mainly used for P0f data.

(7) Session
This default collection is the most important. This collection summarizes honeypot interaction as a normalized 5-tuple record.

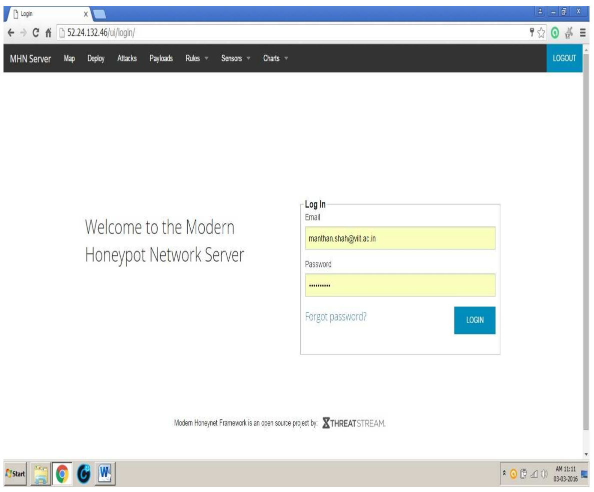
(8) URLs
This default collection is used to store URLs and associated metadata collected from various honeypots. These URLs are potentially serving malicious content.

(9) HPFeeds Database
This collection is specifically used by HPFeeds broker.

(10) Auth_key
This collection stores data about HPFeeds users. It gets populated as new users are added to HPFeeds.

3.5 MHN Web Application

MHN Server provides an attractive web application designed by ThreatStream.
This web app can be used to log in to the MHN Server and do manipulations graphically.


Login
52.24.132.46/ui/login/
MHN Server Map Deploy Attacks Payloads Rules Sensors Charts LOGOUT

Welcome to the Modern
Honeypot Network Server

Log In
Email
manthan.shah@viit.ac.in
Password
••••••••••
Forgot password? LOGIN

Modern Honeynet Framework is an open source project by: THREATSTREAM.

Start AM 11:11 03-03-2016

Fig. 3. Screenshot of MHN Web App Log-in

Having logged into the MHN Server, the homepage is shown. This homepage provides a summary. The attack count of past 24 hours is shown along with a list of top attacker IPs, attacked ports, honeypots, sensors and attack signatures.

The web app provides many facilities:

(1) Map
This tab provides a direct integration to HoneyMap in MHN Server. A real-time map of the world is shown. Different honeypots deployed and attacks are also displayed in real-time fashion.

(2) Deploy

This tab provides the user with the ability to deploy pre-configured honeypots very easily using GUI.

(3) Attacks

This tab provides an attacks report. The attacks are listed using date, sensor, country, source IP, destination port, protocol and honeypot used. These records can be filtered using various 'parameters provided in the search filter.

(4) Payloads

This tab provides a payloads report. The records are listed along with priority, classification and signatures. Search filter is also provided.

(5) Rules

This tab is used for rules management. This tab allows the user to list, filter, edit, remove, activate or deactivate individual rules used in the configuration of Snort. It lets the user change the default source "http://rules.emergingthreats.net/open/snort-2.9.0/emerging.rules.tar.gz". Thus giving the user the ability to configure or reconfigure Snort.

(6) Sensors

This tab provides a list of sensors deployed by MHN Server. Sensors deployed by other servers can also be added using sensor and host names which allows for creation of various honeynets and resource/data sharing between servers.

(7) Charts

This tab provides a direct integration with Kippo. Charts are provided are,

- Top passwords
- Top users
- Top user/passwords
- Top attackers

(8) Settings

This tab provides the current user information. It also lists the users of MHN Server and allows the user to add remove users and change passwords.

3.6 HPFeeds-Logger

HPFeeds-Logger is a simple utility used to log events from HPFeeds in formats compatible with third party integration apps like Splunk and ArcSight. HPFeeds-Logger works with the HPFeeds user which is able to subscribe to all channels.

3.7 Splunk

Splunk has been developed by Michael Baum, Rob Das and Erik Swan. Splunk is a data processor. Splunk is a horizontal technology that deals with raw machine data for application management, security and compliance. Splunk is also user for data mining from machine-generated big data. Splunk is also used for business and web analytics.

Splunk can index structured and/or unstructured machine-generated data in textual form. Splunk can perform real-time and historical search on data and perform statistical analysis and generate reports. Splunk uses Search Processing Language which is derived from SQL and Unix Piping. Using SPL, Splunk can search, filter, modify, insert, delete or manipulate machine-generated big data.

Splunk is used in MHN Server as third party integration to HPFeeds. HPFeeds-Logger is used to index data from HPFeeds into Splunk. Splunk then performs data analysis and generates dashboards and visualization of web based interface which can be used to generate alerts, reports and graphs.

Splunk app integration with MHN was released in 2015, making MHN largest crowd-sourced honeynet.

4. CONCLUSION

Modern Honey Network is explained in the paper above. The architecture, different components and features are discussed. The process flow is explained.

MHN Server has truly overcome the demerits of standard honeypots and provided following unique merits.

(1) Easy to deploy, configure, maintain and use honeypot software.
(2) Risk or exposure is decreased as each sensor is pre-configured.
(3) Field of view and interaction is wide as multiple sensors and honeypots are deployed together.
(4) The risk of identification is decreased as multiple sensors and logical as well as geographical partition between components make discovery and fingerprinting very difficult.

Thus, I conclude that MHN has been able to overcome the demerits of honeypots which have been majorly influential in lack of adoption of honeypots by the security community of the world.

REFERENCES

[1] Spitzner, L. (2001): "The value of honeypots, part one: definitions and values of honeypots"
[2] Bringer, M; Chelmecki, C; Fujinoki, H. (2012): "A survey: recent advances and future trends in honeypot research", International Journal of Computer Network

and Information Security (IJCNIS Vol.4, No.10, September 2012)

[3] Seifert, C; Welch, I, Komisarczuk, P. (2006): "Taxonomy of honeypots", CS Technical Report TR-06-12, School of Mathematics, Statistics and Computer Science, Victoria University of Wellington, New Zealand

[4] The honeynet project (http://honeynet.org/)

[5] MHN, Threatstream (https://github.com/threatstream/mhn)