

# Honeypot

Vladimir Shelkovnikov  
Secure Systems and Network Engineering  
Innopolis University  
v.shelkovnikov@innopolis.university

## I. INTRODUCTION:

Honeypot is a resource that is bait for intruders. The task of the honeypot is to be attacked or unauthorized research, which will subsequently allow you to study the attacker's strategy and determine the list of means by which attacks on real security objects can be carried out. The implementation of a honeypot can be either a special dedicated server or a single network service whose task is to attract the attention of hackers.

A honeypot is a resource that does nothing without any impact on it. Any external interaction with this resource is considered as hacker activity. Honeypot collects a small amount of information, after analyzing which statistics are built on the methods used by hackers, and also determines the presence of any new solutions that will later be used in the fight against hackers.

## II. RESEARCH GOALS

- 1) To analyze honeypot as an information security tool.
- 2) Deploy an existing honeypot and analyze it.
- 3) Implement my honeypot.

## III. RELATED WORKS

John Erickson's book [1] is useful in my research because the author explains the work and the inner essence of exploits.

Review research papers [2, 3] are devoted to the issue of considering the use of honeypot. Since I want to focus on working with the Metasploit, I also need to take into account the available documentation on mute [4]. Research papers devoted to the development of modern Honeypot systems are valuable for my research [5, 6].

## IV. COMPLEX SOLUTION

To achieve the goal, I need to develop my honeypot, which must meet the following requirements:

- 1) Use exploit signatures to work;
- 2) Detected signatures should be entered into the honeypot database;

- 3) Honeypot should support working against Metasploit;
- 4) Support the automation of its work.

## V. HONEYPOT AS AN INFORMATION SECURITY TOOL

To study the actions performed by attackers after they have hacked the system. It should be configured in anticipation of hacking by intruders. But giving attackers a complete system to use is a potential security risk, and therefore it is better to use a honeypot system.

A honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource [2].

The honeypot system is intended solely for investigative purposes. It has no value other than providing services for attackers to interact with. Theoretically, all traffic to the honeypot is considered malicious or unauthorized, it has no legitimate activity [3].

There are many types of honeypots, but they are usually categorized based on the amount of interaction that is offered.

A honeypot with a low level of interaction will only emulate a service or operating system and thus provides a limited amount of interaction. An attacker can execute only those commands that are implemented in the emulation.

Medium-interaction honeypot (MIH) provides much more interaction to the adversaries. However, unlike an LIH, an MIH does not implement the TCP/IP stack by itself. Instead, MIHs (Cowrie [7]) bind to sockets and leave the OS to manage connections. In contrast to LIHs, which implement network protocols, the MIHs modeling algorithm is based on the emulation of logical responses of applications to incoming requests. Thus, the request coming to MIH will be monitored and studied, and fake responses will be created by the MIH security program.

Honeypot with a high level of interaction (HIH) is a fully functional system that can be completely compromised by opponents (Argos [8]). Since a fully functional honeypot can be compromised, HIH must equip security toolkits to monitor system activity and limit outgoing traffic.

Firewalls are often deployed around the perimeter of an organization to block unauthorized access by filtering certain ports [9] and content, but they do little to assess the traffic. They can block all access to a particular service to prevent malevolent traffic, but this also blocks any benevolent traffic that wants to access the service. Conversely, honeypots are aimed at opening ports to capture as many attacks as possible for subsequent data analysis.

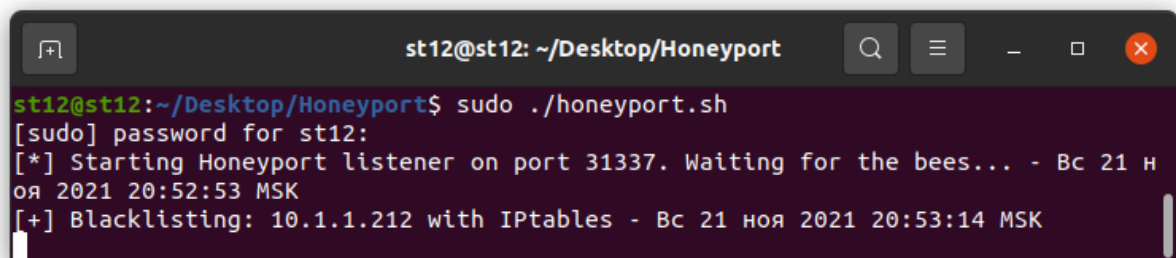
The Intrusion Detection System (IDS) is used to evaluate traffic and detect any inappropriate, incorrect, and abnormal actions. However, IDSs often have a "false alert problem", i.e. signature identifiers (rule-based) often generate false-negative alerts, while anomaly-based

identifiers generate false positive alerts. Compared to IDS, the honeypot has a big advantage in that it never generates false warnings, because any observed traffic to it is suspicious since the production service is not running on the honeypot. Consequently, the integration of bait with IDS can significantly reduce the number of false warnings [10].

## VI. EXCITING HONEYPOTS

For the second goal, I decided to test a few examples of honeypots and see how they are built. This will help me to understand the principles of Honeypots work and correct the conception of my solution.

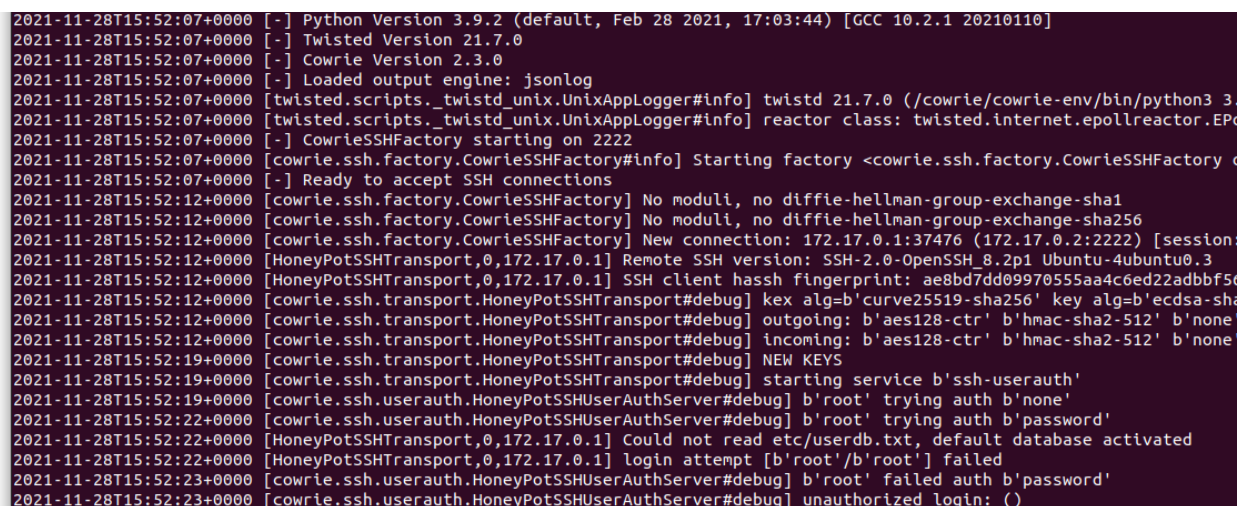
I started with the Honeypot [11]. This is a very simple version of the honeypot. It runs Ncat listener on \$PORT. Gets the client's IP. Check it against a whitelist, if it is not whitelisted - blacklist the IP using iptables or Dmcc and log. An example of its work is shown in Figure 1.



```
st12@st12: ~/Desktop/Honeyport
st12@st12:~/Desktop/Honeyport$ sudo ./honeypot.sh
[sudo] password for st12:
[*] Starting Honeyport listener on port 31337. Waiting for the bees... - Bc 21 ноя 2021 20:52:53 MSK
[+] Blacklisting: 10.1.1.212 with IPtables - Bc 21 ноя 2021 20:53:14 MSK
```

Figure 1 – Honeypot

Now let's test something more interesting, like SSH honeypot, which is a common case. I decided to take Cowrie as an example[7]. It is a medium and high-level SSH and Telnet communication tool designed to register brute force attacks and shell interaction performed by an attacker. In medium interaction mode (shell), it emulates a UNIX system in Python, in high interaction mode (proxy), it functions as an SSH and telnet proxy server to monitor the behavior of an attacker in another system. An example of its operation is shown in Figure 2.



```
2021-11-28T15:52:07+0000 [-] Python Version 3.9.2 (default, Feb 28 2021, 17:03:44) [GCC 10.2.1 20210110]
2021-11-28T15:52:07+0000 [-] Twisted Version 21.7.0
2021-11-28T15:52:07+0000 [-] Cowrie Version 2.3.0
2021-11-28T15:52:07+0000 [-] Loaded output engine: jsonlog
2021-11-28T15:52:07+0000 [twisted.scripts._twisted_unix.UnixAppLogger#info] twisted 21.7.0 (/cowrie/cowrie-env/bin/python3 3.
2021-11-28T15:52:07+0000 [twisted.scripts._twisted_unix.UnixAppLogger#info] reactor class: twisted.internet.epollreactor.EPoll
2021-11-28T15:52:07+0000 [-] CowrieSSHFactory starting on 2222
2021-11-28T15:52:07+0000 [cowrie.ssh.factory.CowrieSSHFactory#info] Starting factory <cowrie.ssh.factory.CowrieSSHFactory o
2021-11-28T15:52:07+0000 [-] Ready to accept SSH connections
2021-11-28T15:52:12+0000 [cowrie.ssh.factory.CowrieSSHFactory] No moduli, no diffie-hellman-group-exchange-sha1
2021-11-28T15:52:12+0000 [cowrie.ssh.factory.CowrieSSHFactory] No moduli, no diffie-hellman-group-exchange-sha256
2021-11-28T15:52:12+0000 [cowrie.ssh.factory.CowrieSSHFactory] New connection: 172.17.0.1:37476 (172.17.0.2:2222) [session:
2021-11-28T15:52:12+0000 [HoneyPotSSHTransport,0,172.17.0.1] Remote SSH version: SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.3
2021-11-28T15:52:12+0000 [HoneyPotSSHTransport,0,172.17.0.1] SSH client hashfingerprint: ae8bd7dd09970555aa4c6ed22adbbf5e
2021-11-28T15:52:12+0000 [cowrie.ssh.transport.HoneyPotSSHTransport#debug] kex alg=b'curve25519-sha256' key alg=b'ecdsa-sha
2021-11-28T15:52:12+0000 [cowrie.ssh.transport.HoneyPotSSHTransport#debug] outgoing: b'aes128-ctr' b'hmac-sha2-512' b'none
2021-11-28T15:52:12+0000 [cowrie.ssh.transport.HoneyPotSSHTransport#debug] incoming: b'aes128-ctr' b'hmac-sha2-512' b'none
2021-11-28T15:52:19+0000 [cowrie.ssh.transport.HoneyPotSSHTransport#debug] NEW KEYS
2021-11-28T15:52:19+0000 [cowrie.ssh.transport.HoneyPotSSHTransport#debug] starting service b'ssh-userauth'
2021-11-28T15:52:19+0000 [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'root' trying auth b'none'
2021-11-28T15:52:22+0000 [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'root' trying auth b'password'
2021-11-28T15:52:22+0000 [HoneyPotSSHTransport,0,172.17.0.1] Could not read etc/userdb.txt, default database activated
2021-11-28T15:52:22+0000 [HoneyPotSSHTransport,0,172.17.0.1] login attempt [b'root'/b'root'] failed
2021-11-28T15:52:23+0000 [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'root' failed auth b'password'
2021-11-28T15:52:23+0000 [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] unauthorized login: ()
```

Figure 2 - Work of Cowrie honeypot

Cowrie is a honeypot that tries to impersonate an SSH server, in particular, with weak and easily hacked login credentials. Once an attacker logs in, they will have access to a fake Linux shell where they can run commands and get realistic responses, but the attacker will never be able to actually execute those real commands outside of the honeypot sandbox environment. That's because this Cowry "shell" isn't really a Linux shell at all. The command-line environment is fully implemented in Python.

Like other honeypots, tricking an attacker into thinking that he is on the server, Cowry will also log or analyze attacks that have been committed against him. This allows the honeypot administrator to get an idea of what attacks are being attempted, their overall success or failure rate, as well as the geographical location of the IP address from which this attack occurs. Cowry is also capable of trying to capture information about a specific attacker, not just metadata about their attack, such as randomly detected SSH fingerprints.

## VII. IMPLEMENTING MY SOLUTION

To set up an FTP service on a honeypot, we need to emulate the entire server and implement a response to all commands that the user is allowed to execute [12]. So I created a medium interactive honeypot that emulates an FTP server and sends responses to malicious requests. To be able to identify and extract the exploit signature, I used the longest common string (LCS), which is effective for several examples of exploit requests. Figure 3 shows the algorithm for my honeypot solution.

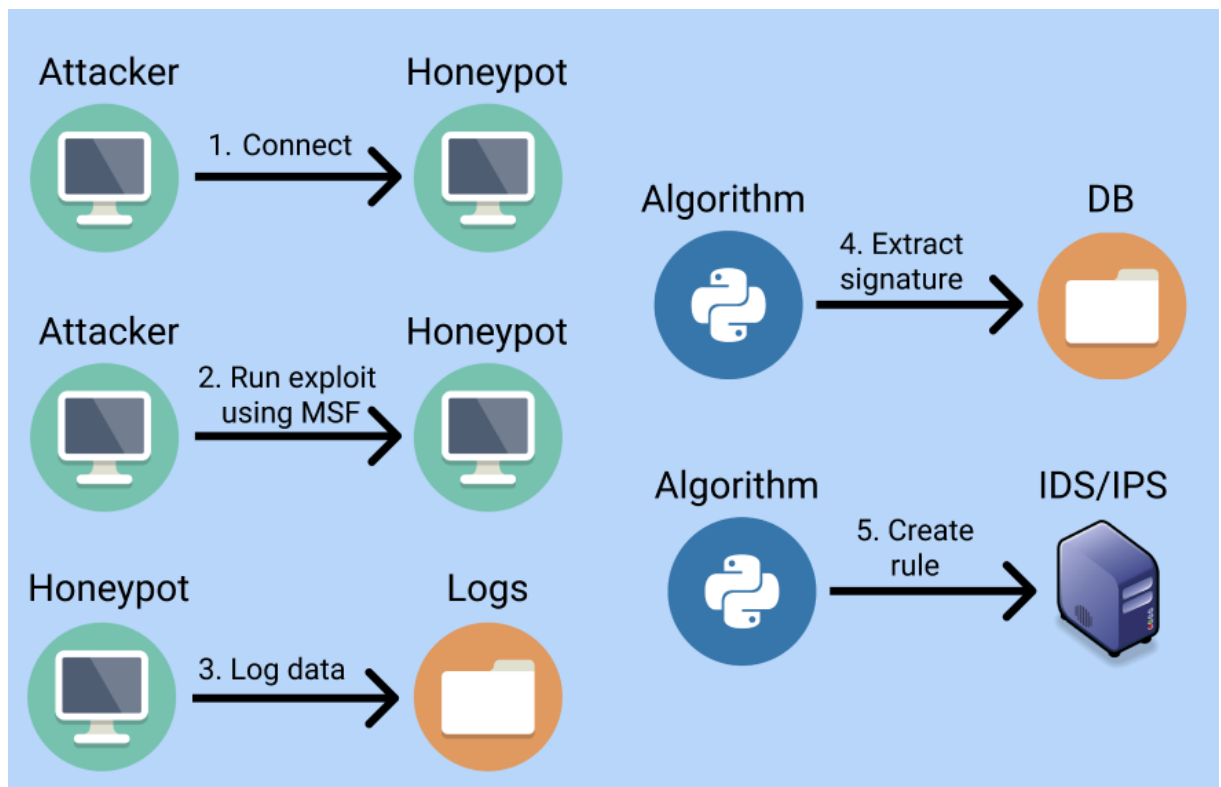


Figure 3 – Work algorithm of my honeypot solution

So the first step here is creating emulation of FTP server (TCP socket on port 21) using socketserver module of python. Then I run exploit using the Metasploit framework. Requests from attackers are saved in logs as shown in Figure 4.



Figure 4 - Logs

Then my algorithm which is based on LCS extracts signature from it as shown in Figure 5.

```
Finfing LCS signature for 2 flows on request nr 3
Hex signature 919f97
```

Figure 5 – Extracted signature

After that my honeypot should create a rule for Snort to prevent its attack, but I struggled with implementing this rule into IDS/IPS so this step is not working at this moment.

Figure 6 shows the topology that I’m used.

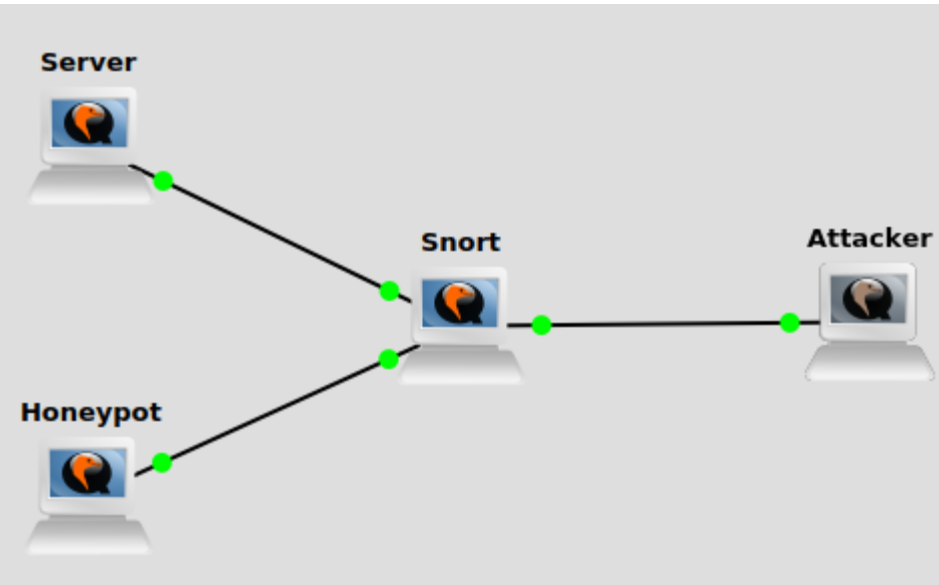


Figure 6 – Network topology

The Snort here is configured to block the traffic to the server, but not from it.

After that, I tested my honeypot with exploits from the Metasploit framework. The result is shown in Table 1.

Exploit	Signature (hex)
3cdaemon_ftp_user	fcae75fd89f989de8a0630074766813f
ability_server_stor	53544f5220
cesarftp_mkd	4d4b44200a
comsnd_ftpd_fmtstr	5040ac7125343233303432327825702570
dreamftp_format	eb29253878253878253878253
easyfilesharing_pass	a342535d8f3e9345aa72f76691ff6aa9ffb6e762ca19dab69351f4198
easyftp_cwd_fixret	89e781ef14feffff890f81c714fffffffe7
easyftp_list_fixret	81c724ffffff81c404fffffffe7
easyftp_mkd_fixret	81c714fffffffe74343434343434343434343
filecopa_list_overflow	4c495354204120
freefloatftp_user	81c454f2ffff
freelftpd_user	81c454f2ffff
globalscapeftp_input	eb0359eb05e8f8ffffff565458363
goldenftp_pass_bof	d97424f4
httpdx_tolog_format	6681caff0f42526a0258cd2e3c055a74efb85730305489d7af75eaaf7
ms09_053_ftpd_nlst	fd7ffd7ffd7ffd7ffd7ffd7ffd7feb243d77f d7f f d7f
netterm_netftpd_user	5553455220c0
oracle9i_xdb_ftp_pass	5553455220c0
oracle9i_xdb_ftp_unlock	81c4ffeffff44
proftpd_133c_backdoor	42424242424424242424242424242424242c8af0408f4590b08
proftp_sreplace	c8af0408f4590b08ccccccccccccc
proftp_telnet_iac	ff
ricoh_dl_bof	5954c377
sami_ftpd_list	81c454f2ffff
sami_ftpd_user	81c454f2ffff
servu_chmod	61caff0f42526a0258cd2e3c055a74efb85730305489d7af75eaaf75e
servu_mdmtm	94631079e18ca44fdc3ba13b9466a470547f981c9477ad50e0
slimftpd_list_concat	d97424f4
turboftp_port	2c3133332c32352c36352c302c34322c31303
vermillion_ftpd_port	2c2c2c2c2c2c2c2c2c2c2c2c2c2c2c2c2c3137312c3438

## CONCLUSION

As a result of this project, a honeypot simulating an FTP server was developed. It allows you to receive the attacker's requests and his IP address. The algorithm is based on the analysis of executed requests, their contents, and quantity, which serve as a signal of suspicious activity. The performance of the developed honeypot has been tested on MSF exploits. As a result, information was obtained about exploit signatures that can be used to prevent such attacks.

## FUTURE WORK

Due to time constraints, some points could not be researched in this research project.

Improving compatibility with Snort as IDS/IPS. Extending the functionality of honeypot to SSH, TCP.

Enhancing the interaction by saving the state of the honeypot for all users. This will give an attacker a more realistic environment to interact with.

Using other algorithms and/or methods to process the stored network traffic might increase performance compared to the Longest Common Substring algorithm used within this research.

## REFERENCES

- [1] Hacking: the art of exploitation by Jon Erickson. <https://redirect.is/bohdp3>
- [2] Meshvez S.Z. Classification of Honeypot systems. <https://elibrary.ru/>
- [3] Rusakov D.V., Pankratova E.A. Classification of honeypot cheating systems. <https://elibrary.ru/>
- [4] Metasploit official documentation. <https://docs.rapid7.com/metasploit/>
- [5] Rusakov D.V., Pankratova E.A. Principles of construction of deceptive systems such as honeypot. <https://elibrary.ru/>
- [6] Arzhakov A.V., Egupov A.A., Silnov D.S. Development and implementation of SSH honeypot traps. <https://elibrary.ru/>
- [7] Cowrie. <https://github.com/cowrie/cowrie>
- [8] Argos <https://github.com/Cyb3r-Jak3/Argos>
- [9] S. Peisert, M. Bishop, and K. Marzullo, "What do firewalls protect? An empirical study of firewalls, vulnerabilities, and attacks," UC Davis CS, Technical Report CSE-2010-8, 2010
- [10] K. G. Anagnostakis, S. Sidiroglou, P. Akritidis, K. Xinidis, E. P., Markatos, and A. D. Keromytis, "Detecting targeted attacks using shadow honeypots," in Usenix Security, 2005.
- [11] Honeyport. <https://github.com/securitygeneration/Honeyport>
- [12] My honeypot. <https://github.com/MrRahmat/ftphoneyport-py>