Build a Web based Honeypot in Python

Rudra Kadel

March 10, 2025

Abstract

This document details the development of a Python-based SSH honeypot system designed to attract, monitor, and analyze malicious access attempts. The honeypot simulates a vulnerable SSH server environment, logs authentication attempts, and captures commands executed by attackers. Through this project, we gain insights into attack patterns, common intrusion techniques, and can develop more effective defensive strategies.

Contents

1 HoneyPie: Overview

1.1 What is HoneyPie?

HoneyPie is an SSH honeypot designed to lure, log, and analyze unauthorized login attempts. It simulates a vulnerable SSH server, capturing attacker activities for security analysis. By using deception, it provides insights into attack techniques and trends, helping defenders improve security strategies.

1.2 Technologies Used

HoneyPie utilizes the following technologies:

- **Python**: Core programming language
- **Paramiko**: SSH protocol handling
- **Socket**: Network communication
- **Threading**: Multi-threaded client handling
- **Logging**: Event tracking
- **Flask**: Web-based honeypot
- **Argparse**: Command-line argument parsing

1.3 Installation and Execution

To install and run HoneyPie, follow these steps:

```
# Clone the repository
git clone https://github.com/rudrakadel/HoneyPie.git
cd HoneyPie

# Create a virtual environment
python -m venv honeypot_env
source honeypot_env/bin/activate

# Install dependencies
pip install -r requirements.txt

# Generate SSH keys
ssh-keygen -t rsa -b 2048 -f static/server.key -N ""

# Run the honeypot
python honeypy.py -a 0.0.0.0 -p 2222 -s
```

Listing 1: Installation and Running

1.4 Dependencies and Requirements

HoneyPie requires:

- Python 3.x
- Required libraries (listed in `requirements.txt`)
- Linux environment (recommended for running SSH services)
- SSH key generation tools
- Flask for the web-based honeypot

1.5 Honeypy.py: Main Honeypot Controller

The `honeypy.py` file is the main script that interfaces with the honeypot system. It supports both SSH and HTTP honeypots, allowing users to specify different modes using command-line arguments.

```
# Import necessary libraries
import argparse
from ssh_honeypot import *
from web_honeypot import *
from dashboard_data_parser import *
from web_app import *

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument('-a','--address', type=str, required=True)
    parser.add_argument('-p','--port', type=int, required=True)
    parser.add_argument('-s', '--ssh', action="store_true")
    parser.add_argument('-wh', '--http', action="store_true")
    args = parser.parse_args()

    try:
        if args.ssh:
            print("[-] Running SSH Honeypot...")
            honeypot(args.address, args.port)
        elif args.http:
            print("[-] Running HTTP Honeypot...")
            run_app(args.port)
        else:
            print("[!] Specify either SSH (-s) or HTTP (-wh) mode."
    )
    except KeyboardInterrupt:
        print("\nProgram exited.")
```

Listing 2: honeypy.py

1.6 Web Honeypot: Wordpress Login Emulator

The `web_honeypot.py` script simulates a fake Wordpress login page using Flask. Attackers entering credentials have their data logged for analysis.

```python
from flask import Flask, render_template, request
import logging
from logging.handlers import RotatingFileHandler

app = Flask(__name__)
logger = logging.getLogger('HTTPLogger')
logger.setLevel(logging.INFO)
funnel_handler = RotatingFileHandler("http_audit.log", maxBytes
    =2000, backupCount=5)
logger.addHandler(funnel_handler)

@app.route('/')
def index():
    return render_template('wp-admin.html')

@app.route('/wp-admin-login', methods=['POST'])
def login():
    username = request.form['username']
    password = request.form['password']
    ip_address = request.remote_addr
    logger.info(f'IP: {ip_address} - Username: {username}, Password
    : {password}')
    return "Invalid login. Try again."

def run_app(port=5000):
    app.run(debug=True, port=port, host="0.0.0.0")
```

Listing 3: web$_h$*oneypot.py*

1.7 Contributing to the Project

Developers can contribute by:

- Submitting pull requests on GitHub
- Reporting issues and suggesting features
- Enhancing logging, analysis, and deception techniques

1.8 Project License

HoneyPie is released under the MIT License. See `LICENSE` for details.