Generovanie honeypotov na základe dôkazov zneužiteľnosti zraniteľností.

Analýza a návrh riešenia

Bc. Filip Dvorský

1m, 2024 – 2025

Abstrakt. V tejto práci sa venujeme generovaniu honeypotov s nízkou úrovňou interakcie na základe dôkazov zneužiteľnosti zraniteľností. Cieľom je navrhnúť a overiť systém, pomocou ktorého bude následne možné vygenerovať honeypot, ktorý bude napodobňovať správanie zraniteľnej služby nachádzajúcej sa na portoch 443 a 80. Honeypot nám následne umožní cielene zbierať presnejšie dáta o útočníkoch a ich taktikách ako u štandardných honeypotoch. Práve pomocou týchto presných dát o útočníkoch by sme vedeli efektívnejšie nastaviť pravidlá a politiky týkajúce sa nových hrozieb v oblasti kybernetickej bezpečnosti.

Kľúčové slová: Kybernetická bezpečnosť, honeypot, honeynet, zraniteľnosti, automatizácia

1 Úvod

Nové technológie so sebou prinášajú nové typy útokov a kybernetických hrozieb, ktoré si vyžadujú nové flexibilné prístupy a nástroje k monitorovaniu a prevencii daných hrozieb. Jedným z týchto kľúčových nástrojov sú aj Honepoty. Honepoty sú zariadenia určené ako pasce na útočníkov, snažiacich sa získať prístup k zariadeniu, získať dáta z daného zariadenia atď., pričom honepot počas útoku monitoruje aktivitu útočníka, čo následne pomáha zlepšovať bezpečnostné opatrenia ako napríklad nastavenie firewall-u, Threat-Hunting a mnoho ďalších.

Tradičné honeypoty však čelia výzvam, ako je napríklad neschopnosť reagovať na nové zraniteľnosti v reálnom čase, ale aj generovanie veľkého množstva údajov a dát, z ktorých

podstatná časť má nízku výpovednú hodnotu pre bezpečnostného analytika. Tradičným príkladom sú tzv. “script kiddies“, sú to útočníci, ktorí využívajú základné nástroje na vykonávanie jednoduchých automatizovaných útokov. Výsledkom je teda zahltenie bezpečnostného analytika nepotrebnými informáciami, čo vo výsledku komplikuje a predlžuje prácu analytika.

2 Honeypot

V počítačovej bezpečnosti termín „Honeypot“ vieme definovať ako pascu na útočníkov/hackerov. Určenie tohto zariadenia je imitovať ciel za účelom získania informácii, postupov a technik využívaných útočníkmi pri získaní prístupu/informácii z daného cieľového systému.

Z náplne úlohy daného zariadenia teda jasne vyplýva, že útočník ma získať prístup k danému zariadeniu, preto teda môžeme o honeypotoch hovoriť ako o obetnom počítačovom systéme [1]. Pravé z tohto dôvodu sú dané honeypoty zväčša cielene nezabezpečené zariadenia, plne rôznych zraniteľnosti, ktoré môže útočník využiť na získanie prístupu k danému zariadeniu [2]. Pravé tato vlastnosť, spolu z tým, že bežný užívateľ ma nulovú pridanú hodnotu navštíviť, alebo interagovať z daným honeypotom, nám zabezpečuje, že všetky honeypotom získane data a aktivita sú získane z činnosti útočníka. A pravé tato nízka false-pozitivita je jedným z hlavných výhod uplatnenia honeypotu v infraštruktúre organizácie. Avšak záleží len na bezpečnostnom tíme akým spôsobom využije získane data z honeypotov. [3]

Honeypot ako taký vieme klasifikovať do jednotlivých kategórii, podlá módu interakcie, podľa typov dát, ktoré zbiera, podľa role v organizácii, typu honeypotu a podľa mnoho ďalších kategórii. Hlavne budeme teda honeypoty klasifikovať podľa ich módu interakcie z útočníkom, role a typu dát ktoré daný honeypot bude zbierať.

2.1 Klasifikácia honeypotov podľa role v infraštruktúre

Honeypoty vieme klasifikovať podľa ich roly v infraštruktúre organizácie. Budeme ich deliť na server-side a client-side honeypoty. Ako názvy napovedajú jednotlivé honeypoty budú simulovať zariadenia z daných kategórii (server , klientska stanica). Tento typ rozdelenia vychádza zo základného princípu klient-server infraštruktúry, kde honeypot reprezentuje správanie buď server zariadenia alebo klientského zariadenia.

Client-Side honeypoty, ktoré blízko napodobňujú správanie klientskych staníc, pričom tieto honeypoty umožňujú zistiť, aké typy útokov sú smerovane na používateľov / klientsku časť infraštruktúry organizácie.

Server-Side honeypoty sú oproti klient-side implementovane priamo v rámci serverov pripadne v ich sieťovej infraštruktúre a pomáhajú odhaliť a analyzovať pokusy o prienik do siete, analyzovať útoky, atď.

Klasifikácia honeypotov vzhľadom na ich rolu v infraštruktúre je dôležité z hľadiska ich nasadenia v infraštruktúre organizácie, ale aj pri návrhu spôsobu zberu dát, ktoré sledujú. V našej práci sa budeme zameriavať na server-side honeypoty, keďže cieľom našej práce je analyzovať a simulovať správanie zraniteľných služieb.

2.2 Klasifikácia honeypotov podľa módu interakcie

Podľa interakcie delíme honeypoty v základe podľa možnosti a schopnosti ktoré sú poskytovane útočníkovi daným honeypotom. Dane možnosti sú v rozmedzí od jednoduchej replikácie komunikácie reálneho zariadenia až po plnú interakciu z operačným systémom. Preto honeypoty delíme na tri úrovne:

Honeypot z nízkou úrovňou interakcie detegujú útočníkov pomocou základnej charakteristiky daného operačného systému a sieťových služieb na zariadení. Jednou z výhod tohto prístupu je podstatne lepšia kontrola nad útočníkovými aktivitami, dôvodu limitovaného prístupu útočníka k službám na danom zariadení. Avšak medzi nevýhody patri obmedzený počet relevantných dát získaných z daných honeypotov keďže útočník je limitovaný pravé na vybrane služby [4].

Honeypoty z strednou úrovňou interakcie sú mierne viac sofistikovanejšie ako honeypoty z nízkou úrovňou interakcie. Tieto honeypoty stále neposkytujú plný operačný systém útočníkovi, avšak obsahujú viac zraniteľností a viac simulovaných služieb, ktoré môže útočník využiť na získanie prístupu [4, 5].

Honeypot z vysokou úrovňou interakcie poskytuje útočníkovi plný prístup k operačnému systému spolu z jeho službami a softvérom [4]. Tento prístup produkuje podstatne viac dát ako predošlé prístupy keďže všetky akcie vykonane útočníkom sú zaznamenane, spracovane a následne analyzované [5].

2.3 Klasifikácia honeypotov podľa typu

Jednotlivé typy honeypotov sa líšia podľa rôznych dát, ktoré zhromažďujú, pričom tieto dáta závisia od konkrétnych hrozieb, pre ktoré je honeypot určený. Honeypot môže byt zameraný

na imitáciu jednotlivých webových stránok, zatiaľ čo iné sa zameriavajú na zber emailových adries útočníkov, atď.

Malvérový honeypot využíva už známe vektory útoku na „prilákanie“ malvéru. Ako príklad malvérový honeypot môže simulovať USB úložne zariadenie. V prípade napadnutia daného systému honeypot oklame malvér k zaútočeniu na simulovane USB zariadenie za účelom získania malvér súboru [6].

Spam-ové honeypoty sú navrhnute tak, aby prilákali útočníkov pomocou otvorených proxy serverov k rozposlaniu hromadnej nechcenej posty (spam). Ak sa útočník pokúsi o rozposlanie spam-u, honeypot danú aktivitu zablokuje a zozbiera relevantne dáta spojené z útočníkom [6].

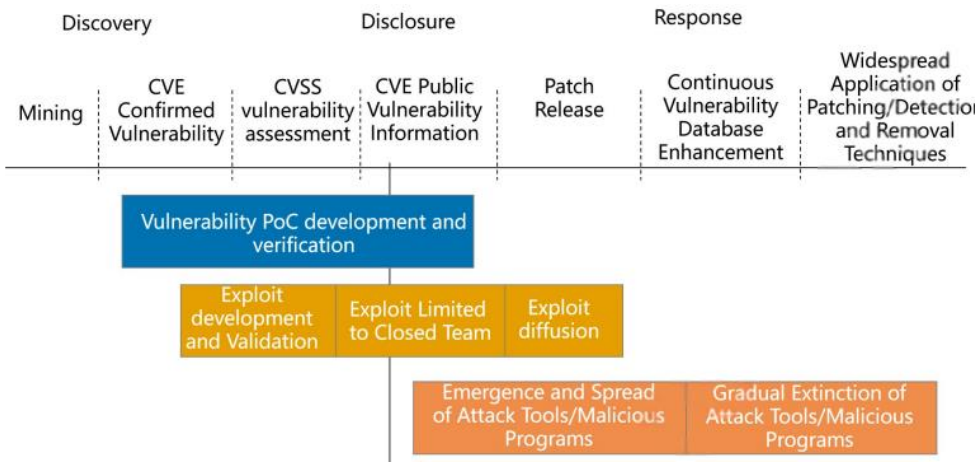
Databázový honeypot sa používa na vytvorenie falošných databáz za účelom nalákania útočníkov na využitie SQL-injection útoku špecifických pre danú databázu [6].

Klientský honeypot sa pokúša o nalákanie malicióznych serverov, ktoré využívajú útočníci. Dany honeypot sa tvári ako bežné klientske zariadenie a sleduje ako útočník modifikuje dané zariadenie za účelom získania prístupu k zariadeniu a získania prístupu k iným zariadeniam v sieti [6].

3 Zraniteľnosti

Zraniteľnosť je chyba/slabina v logike kódu, jednotlivých softvérových alebo hardvérových komponentoch, ktorej zneužitie ma negatívny vplyv na dôvernosť, integritu alebo dostupnosť [7]. Tieto vlastnosti predstavujú takzvanú CIA triadu (Confidentiality, Integrity, Availability), ktorá je základným princípom v kybernetickej bezpečnosti [8].

Zraniteľnosť sama o sebe nepredstavuje riziko, avšak v momente ak je objavená a existuje možnosť danú zraniteľnosti využiť, napríklad pomocou dôkazu zneužiteľnosti zraniteľnosti, tak sa stáva bezpečnostným problémom. V kybernetickej bezpečnosti je preto potrebné chápať aj praktickú využiteľnosť danej zraniteľnosti. Z tohto dôvodu potrebujeme aj chápať samotný životný cyklus zraniteľnosti a ako do daného cyklu vstupujú dôkazy zneužiteľnosti zraniteľností.



Obrázok 1: Životný cyklus zraniteľnosti [9].

Samotné zraniteľnosti vieme kategorizovať do viacerých typov. Napríklad podľa technickej vrstvy výskytu. Tieto klasifikácie sú obzvlášť doležíte pri návrhu honeypotov, ktoré majú simulovať správanie zraniteľných systémov a komponentov.

3.1 Klasifikácie zraniteľností

Ako už bolo spomínané zraniteľností vieme klasifikovať a kategorizovať do jednotlivých skupín podľa rôznych atribútov. Avšak pre správne pochopenie závažnosti danej zraniteľnosti potrebujeme systematicky hodnotiť jednotlivé zraniteľnosti podľa určitej stanovenej metriky. Tieto klasifikácie sú kľúčové pre presné určenie reálnych dôsledkov danej zraniteľnosti a pre výber dôkazov zneužiteľnosti zraniteľností na základe, ktorých budeme následne vytvárať naše honeypoty.

V ďalších podkapitolách sa preto budeme podrobne venovať trom najvýznamnejším štandardom a systémom, ktoré sa aktuálne používajú na kategorizáciu daných zraniteľností podľa závažnosti a podľa slabín v systéme, spolu z jednoznačnou identifikáciou zraniteľností.

3.1.1 Závažnosť zraniteľností

Jednotlivé zraniteľnosti sa medzi sebou líšia viacerými faktormi, jedným z týchto faktorov môže byt aj ako závažná je daná zraniteľnosť. Pre účely kybernetickej bezpečnosti môžeme závažnosť popísať stručne ako vyjadrenie rizika nejakou definovanou formou. Pre vyriešenie a mitigovanie jednotlivých problémov je preto potrebné poznať pravé túto závažnosť pre efektívnu prioritizáciu riešenia daných zraniteľností.

Pre kvantifikovanie závažnosti danej zraniteľnosti sa v súčasnosti používa takzvane CVSS (Common Vulnerability Scoring System) skóre, skrátene CVSS skóre. Toto skóre kvalitatívne popisuje závažnosť, avšak samotné skóre nie je meradlom nebezpečenstva. Výstupná metrika je číselné skóre v rozsahu od 0 po 10. Dané číselné skóre môže byť následne reprezentované kvalitatívnym vyjadrením ako napríklad nízke, stredné, vysoké a kritické. CVSS skóre je definované viacero štandardami, pričom najnovší štandard je CVSS v4.0, avšak CVSS v3.1 je stále bežne používaný. Tento štandard kvalitnejšie popisuje faktory vystupujúce pri určení závažnosti pre danú zraniteľnosť a aké sú externé faktory a dopady na CIA triádu [10, 11].

Obrázok 2: CVSS skóre pre zraniteľnosť CVE- 2021-44228 [12].

Na Obr. Č. 2 môžeme vidieť samotné CVSS skóre pre zraniteľnosť CVE-2021-44228 spolu z jej vektorovým reťazcom. V rámci systému CVSS existuje takzvaný vektory reťazec, ktorý sumarizuje všetky parametre na základe, ktorých je dané skóre vyhodnotené. Tento vektorový reťazec ma štandardizovaný formát a začína identifikátorom verzií štandardu, napríklad „CVSS:3.1“. Následne medzi každou ďalšou položkou je oddeľovací znak „/“. Ďalej sa v danom vektorovom reťazci nachádzajú jednotlivé položky, ktoré bližšie popisujú dopad a rozsah danej zraniteľnosti. Medzi tieto patrí napríklad „AV“, ktorý popisuje vektor útoku a teda spôsob prístupu k zraniteľnosti, „AC“, ktorý popisuje náročnosť využitia danej zraniteľnosti, a ďalšie ako napríklad „C“, „I“ a „A“, ktoré popisujú dopad na CIA triádu [13].

3.1.2 Klasifikácia podľa slabiny v systéme

Slabinu môžeme definovať ako stav v softvéri, hardvéri alebo v nejakom komponente, ktorý by mohol prispieť k vzniku zraniteľnosti. Jednotlivé podmienky vzniku slabín sú teda priamo alebo nepriamo v réžii vývojára počas vývoja produktu. CWE (Common Weakness Enumeration) je verejný zoznam typov pravé takýchto slabín v systémoch.

Každý záznam v CWE zozname disponuje svojim identifikátorom vo forme „CWE-<ID>“, kde <ID> je unikátne číslo priradene v čase pridania do zoznamu. Nasleduje názov danej slabiny spolu z stručným popisom. Následne daný záznam disponuje bežnými dopadmi na bezpečnosť systému a teda dopad na CIA triádu a potencionálne možnosti zmiernenia dopadu. Potencionálne zmárnenia dopadu v tomto zazname figurujú ako rady pre vývojárov ako odstrániť danú slabinu zo systému [14].

3.1.3 Identifikácia zraniteľnosti

Spolu z kategorizáciou zraniteľnosti podľa jednotlivých slabín v systéme a podľa závažnosti potrebujeme dané zraniteľnosti aj nejak jednoznačne identifikovať. Pravé pre tento účel existuje viacero systémov pre jednoznačne identifikovanie zraniteľnosti.

Medzi najpoužívanejšie patri Common Vulnerabilities and Exposures, skrátene CVE zoznam. Tento zoznam je podobne ako CWE spravovaný firmou MITRE v spolupráci s CISA. Každá zraniteľnosť ma pridelený jedinečný identifikátor vo forme „CVE-yyyy-zzzzz“, kde „yyyy“ predstavuje rok zverejnenia a „zzzzz“ predstavuje poradové číslo v zozname CVE. Každá identifikovaná zraniteľnosť nachádzajúca sa v CVE zozname disponuje stručným popisom, zasiahnutým softvérom spolu z verziou daného softvéru, časovými pečiatkami identifikácie a upraví záznamu, CVSS skóre, CWE, a ďalšie ako je napríklad meno osoby, ktorá našla danú zraniteľnosť, či referencie [15].

3.2 Dôkaz zneužiteľnosti zraniteľnosti

Dôkaz zneužiteľnosti zraniteľnosti, inak PoC exploit (Proof of Concept exploit), je publikáciou bezpečnostného analytika za účelom poskytnutia technických detailov, ktoré ilustrujú ako môžu jednotlivý útočníci využiť danú zraniteľnosť na dosiahnutie ich cieľa. Tento neškodný útok demonštruje potencionálny vplyv, ktorý môže mať konkrétna zraniteľnosť na bezpečnosť v organizácii. Dôkazy zneužiteľnosti zraniteľnosti pomáhajú tímom lepšie spravovať zraniteľnosti v organizácii a umožňujú rýchlejšiu tvorbu záplat pre dotknute zariadenia. Dôkazy zneužiteľnosti zraniteľnosti poukazujú na možnosť útočníkov využiť zraniteľnosti na získanie neoprávneného prístupu alebo na umožnenie neúmyselných akcii v rámci systémov. Aj keď cieľom dôkazu zneužiteľnosti zraniteľností je pomôcť určiť priority pre uplatňovanie bezpečnostných aktualizácií, verejné zverejnenie dôkazu zneužiteľnosti zraniteľností môže viesť útočníkov k vytvoreniu nových exploitov, ktoré môžu byt použite pri budúcich útokoch [16].

```java
public class log4j {
    private static final Logger logger = LogManager.getLogger(log4j.class);

    public static void main(String[] args) {
        //The default trusturlcodebase of the higher version JDK is false
        System.setProperty("com.sun.jndi.ldap.object.trustURLCodebase","true");
        logger.error("${jndi:ldap://127.0.0.1:1389/Exploit}");
    }
```

Obrázok 3: Ukážka dôkazu zneužiteľnosti zraniteľnosti (CVE-2021-44228) [17].

Na Obr. Č. 2 môžeme vidieť dokaz zneužiteľnosti zraniteľností pre zraniteľnosť CVE-2021-44228, lepšie známu ako log4j. Jedna sa o jednoduché overenie existencie výskytu zraniteľnosti na zariadení, pričom ako názov napovedá daná zraniteľnosť sa nachádza v logovacej knižnici log4j pre programovací jazyk Java.

3.3 Metasploit Framework

Metasploit je projekt, ktorý poskytuje data o bezpečnostných zraniteľnostiach a pomáha pri vykonávaní penetračného testovania. Vlastni ho americká spoločnosť Rapid7, ktorá je uznávaná v oblasti kybernetickej bezpečnosti. Významným nástrojom z projektu Metasploit je open-source Metasploit Framework.

Metasploit Framework je modulárny nastroj pre penetračné testovanie založený na jazyku Ruby, ktorý umožňuje písať, testovať a spúšťať jednotlivé dôkazy zneužiteľnosti zraniteľností na zariadeniach. Metasploit Framework obsahuje sadu nástrojov, ktoré sa dajú použiť na testovanie bezpečnostných zraniteľností, vykonávanie útokov a vyhýbanie sa detekcii na zariadení. Jadrom Metasploit Framework je kolekcia bežne používaných nástrojov, ktoré poskytujú kompletné prostredie na penetračné testovanie a vývoj nových dôkazov zneužiteľnosti zraniteľností [18].

```
def exploit
  validate_configuration!
  if datastore['HTTP_HEADER'].blank?
    targetinfo = (@checkcode&.details || []).reject { |ti| ti[:headers].blank? }.first
    http_header = targetinfo[:headers].keys.first if targetinfo
    fail_with(Failure::BadConfig, 'No HTTP_HEADER was specified and none were found automatically') unless http_header

    print_good("Automatically identified vulnerable header: #{http_header}")
  else
    http_header = datastore['HTTP_HEADER']
  end
```

Obrázok 4: Ukážka časti dôkazu zneužiteľnosti zraniteľnosti v Metasploit Framework pre zraniteľnosť CVE-2021-44228 [19].

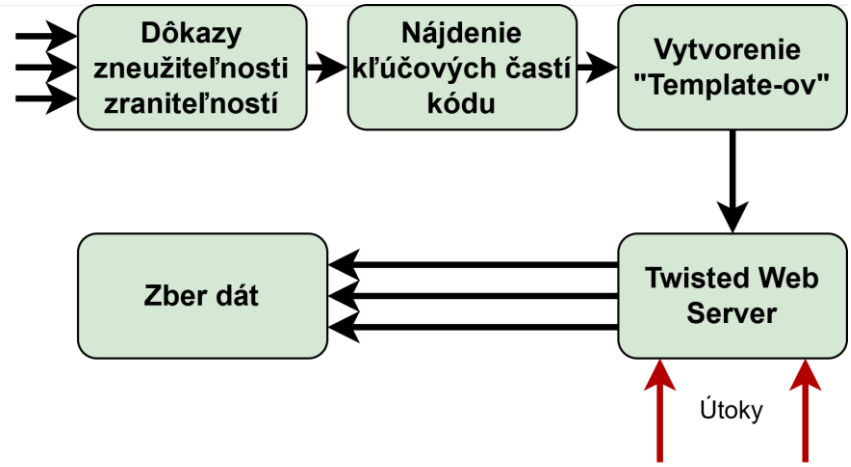
Samotné dôkazy zneužiteľnosti zraniteľností v nástroje Metasploit Framework sa nazývajú moduly a sú písane v jazyku Ruby. V Metasploit Framework predstavujú tieto moduly plnohodnotné dôkazy, že daná zraniteľnosť je zneužiteľná za určitých podmienok. Na obr. Č. 4 sa nachádza časť z dôkazu zneužiteľnosti zraniteľnosti pre zraniteľnosť Log4J. Každý modul disponuje jednotlivými časťami kde sú popísane základné informácie o danej zraniteľnosti, ako napríklad CVE identifikátor danej zraniteľnosti, mená autorov modulu, stručný popis zraniteľnosti a ďalšie klasifikácie zraniteľnosti pre účely Metasploit Framework-u. Následne každý modul disponuje funkciou s názvom „exploit“, v ktorej sa vykonáva hlavná logika kódu. Modul môže disponovať aj ďalšími funkciami avšak tieto funkcie sú určené len ako pomocné funkcie pre nastavenie potrebných parametrov, hodnôt a objektov potrebných pre úspešne otestovanie danej zraniteľnosti.

Pre účely našej práce sú tieto moduly nesmierne prospešne, z dôvodu presnej definície vstupných parametrov a hodnôt, presnej definície očakávanej odpovede na definovaný vstup, spolu z štruktúrovanou formou blokov, kde každý blok kódu ma špecificky účel. Pravé analýza týchto modulov nám umožni reverzne analyzovať dané dôkazy zneužiteľnosti zraniteľností pre vytvorenie logiky správania nášho honeypotu.

4 Návrh riešenia

Hlavným cieľom tejto práce je navrhnúť, implementovať a overiť systém, ktorý bude pomocou dôkazu zneužiteľnosti zraniteľností vytvárať honeypot z nízkou úrovňou interakcie pre daný dôkaz zneužiteľnosti zraniteľností. Cieľom teda je vytvoriť prostredie, ktoré dokáže detegovať pokusy o zneužitie špecifických vybraných zraniteľností. Pričom naše zameranie bude na dôkazy zneužiteľnosti zraniteľností pre služby fungujúce na protokole HTTPS a HTTP a teda na portoch 443 a 80, ktoré predstavujú najčastejšie ciele útočníkov. Vyber špecifických dôkazov zneužiteľnosti zraniteľností sme limitovali aj z dôvodu jednoduchšej implementácie HTTPS a HTTP servera. Návrh nášho systému je teda rozdelený do hlavných 4 sekcii.

- Analýza dôkazov zneužiteľnosti zraniteľností
- Vytvorenie šablón odpovedí
- Vytvorenie honeypotu s nízkou úrovňou interakcie
- Zber dát



Obrázok 5: Zjednodušená schéma návrhu riešenia.

Vstupom do nášho systému je samotný dôkaz zneužiteľnosti zraniteľnosti, ktorý reprezentuje scenár útoku na zariadenie. Tento dôkaz ďalej analyzujeme, kde hľadáme a určujeme jednotlivé sekcie kódu v danom dôkaze zneužiteľnosti zraniteľnosti, ktoré vystupujú ako overenia

výstupu zariadenia pripadne ďalšie parametre, ktoré musí zariadenie poslať útočníkovi ako jeho výstup. Medzi tieto parametre patria HTTP metódy, špecifické podmienky, overovacie reťazce, atď. Tieto parametre môžeme ďalej upraviť do jednotlivých šablón odpovedí, kde daná šablóna reprezentuje odpoveď zariadenia. Teda pre špecifický dôkaz zneužiteľnosti zraniteľnosti môže mať zariadenie niekoľko odpovedí, ktoré sa vyskytnú počas komunikácie s útočníkom. Tieto špecifické odpovede si vieme predpripraviť a poslať, ak dostaneme špecifickú požiadavku, presne určenú z analýzy daného dôkazu. Práve pomocou šablón odpovedí vieme ďalej prejsť k samotnému generovaniu nízko úrovňového honeypotu. Náš nízko úrovňový honeypot bude následne zbierať nami preddefinovane údaje o útočníkovi, počas celého útoku.

V nasledujúcich podkapitolách sa budeme detailne venovať nášmu návrhu postupu generovania honeypotov spolu z popisom potrebných nástrojov, návrhu logiky a zberu dát, ktoré sú potrebné pre korektné generovanie honeypotov.

4.1 Analýza dôkazov zneužiteľnosti zraniteľností

Prvé pri procese vytvorenie nášho honeypotu musíme analyzovať samotný dôkaz zneužiteľnosti zraniteľnosti, za účelom správneho generovania a správania nášho honeypotu. Pre účely našej práce sme sa rozhodli pre získanie dôkazov zneužiteľnosti zraniteľností pravé z už spomínaného Metasploit Framework-u, keďže dané dôkazy sú štruktúrované. Práve kvôli ich štruktúrovanej forme môžeme lepšie identifikovať špecifické vlastnosti v daných dôkazoch. Ako už bolo spomínané, každý Metasploit Framework modul disponuje špecifickými časťami kódu určené pre spustenie kódu potrebného pre inicializovanie premenných, vytvorenie objektov, popis daného dôkazu zneužiteľnosti zraniteľností, overenie splnenia podmienok pre vykonanie dôkazu a samotná logika daného dôkazu.

Popis daného dôkazu zneužiteľnosti zraniteľnosti sa nachádza v bloku „initialize“, kde sa nachádzajú všetky potrebné informácie a dokumentácia daného dôkazu spolu s CVE identifikátorom, menami autorov, názvom cieľovej služby a z rozsahom zraniteľných verzii danej služby. Ako ďalší hlavný blok je „check“, v tomto bloku sa vykoná overenie všetkých pre rekvizít a podmienok, ktoré sú nutné pre úspešne overenie výskytu danej zraniteľnosti. Blok „exploit“ je samotné jadro logiky daného dôkazu. V tomto bloku sa nachádza hlavná logika kódu, a teda of vytvárania rôznych tokenov a reťazcov, ktoré sú následne očakávane na výstupe, až po naplnenie protokolových hlavičiek potrebnými dátami.

```
def exploit
  wmusecurity_id = find_wmusecurity_id[0]
  php_page_name = "#{rand_text_alpha(5..12)}.php"
  data = Rex::MIME::Message.new
  data.add_part('wmuUploadFiles', nil, nil, 'form-data; name="action"')
  data.add_part(wmusecurity_id, nil, nil, 'form-data; name="wmu_nonce"')
  data.add_part('undefined', nil, nil, 'form-data; name="wmuAttachmentsData"')
  data.add_part('1', nil, nil, 'form-data; name="postId"')
  data.add_part("GIF8#{payload.encoded}", 'image/gif', nil, "form-data; name=\"wmu_files[0]\"; filename=\"#{php_page_name}\"")
  post_data = data.to_s
```

Obrázok 6: Ukážka "exploit" sekcie dôkazu pre zraniteľnosť CVE-2020-24186 [20].

Na obr. Č. 6 môžeme vidieť časť „exploit“ sekcie z dôkazu zneužiteľnosti zraniteľností CVE-2020-24186. CVE-2020-24186 je zraniteľnosť umožňujúca vzdialene spustenie kódu pomocou nahratia škodlivého kódu ako .gif obrázka do Wordpress pluginu wpDiscuz. Na obr. Č. 6 je znázornená hlavná logika vytvárania škodlivej požiadavky, ktorá simuluje, ako už bolo spomínane, nahratie maliciozneho súboru cez webový formulár pre pridávanie komentárov do pluginu wpDiscuz [20]. Ako môžeme vidieť na začiatku funkcie „exploit“ premenná „php_page_name“ je náhodne generovaná, čo zabezpečuje, že daný dôkaz zneužiteľnosti funguje bez ohľadu na dĺžku či špecifickosť názvu súboru. Tieto prvky sú pre nás dôležité, pretože pravé tieto prvky budú tvoriť základ našich šablón odpovedi, ktoré bude náš honeypot využívať pre komunikáciu z útočníkom.

```
json_data = JSON.parse(res.body)
upload_url = json_data.dig('data', 'previewsData', 'images', 0, 'url')
fail_with(Failure::UnexpectedReply, "#{peer} - Upload was unsuccessful")
wp_shell_upload = upload_url.split('/').last
fail_with(Failure::NotFound, "#{peer} - Path not found in response body")
print_good("Payload uploaded as #{php_page_name}")
register_file_for_cleanup(php_page_name)
```

Obrázok 7: Ukážka vyhodnotenia odpovede z "exploit" sekcie dôkazu pre zraniteľnosť CVE-2020-24186 [20].

Na obr. Č. 7 sa nachádza vyhodnocovacia logika v „exploit“ bloku. Môžeme vidieť, že v tomto prípade korektná odpoveď v premennej „php_page_name“ bola vytvorená na začiatku bloku. Presnejšie vytvorenie danej premennej môžeme vidieť na obr. č. 6, kde daná premenná bola inicializovaná náhodným reťazcom. Nájdením týchto kľúčových časti kódu vieme následne vytvoriť špecifické šablóny, pomocou ktorý bude náš honeypot odpovedať.

4.2 Vytvorenie šablón odpovedí

Z analýzy dôkazov zneužiteľnosti zraniteľností, konkrétne modulov v Metasploit Framework-u, je našim ďalším krokom návrh a vytvorenie šablón odpovedí. Cieľom šablón je napodobnenie komunikácie a správania zraniteľného zariadenia.

Šablóny budeme vytvárať na základe získaných parametrov z analýzy, konkrétne sa zameriame na tie časti kódu, ktoré buď generujú alebo očakávajú nejaké špecifické reťazce, objekty, HTTP hlavičky, odpovede servera a podobne. Napríklad ak daný dokaz zneužiteľnosti zraniteľnosti využíva špecifickú HTTP hlavičku na to aby bol prenesený špecificky vygenerovaný reťazec, tak náš honeypot bude mať predpripravené šablóny pre tuto komunikáciu. Každá šablóna musí disponovať nasledujúcimi komponentami. Štruktúrou požiadavky na, ktorú ma honeypot odpovedať. Teda definovaná požiadavka ideálne cela HTTP hlavička spolu z ďalšími potrebnými informáciami. Definovanú odpoveď, ktorá by v ideálnom prípade mala byť taktiež celá HTTP hlavička.

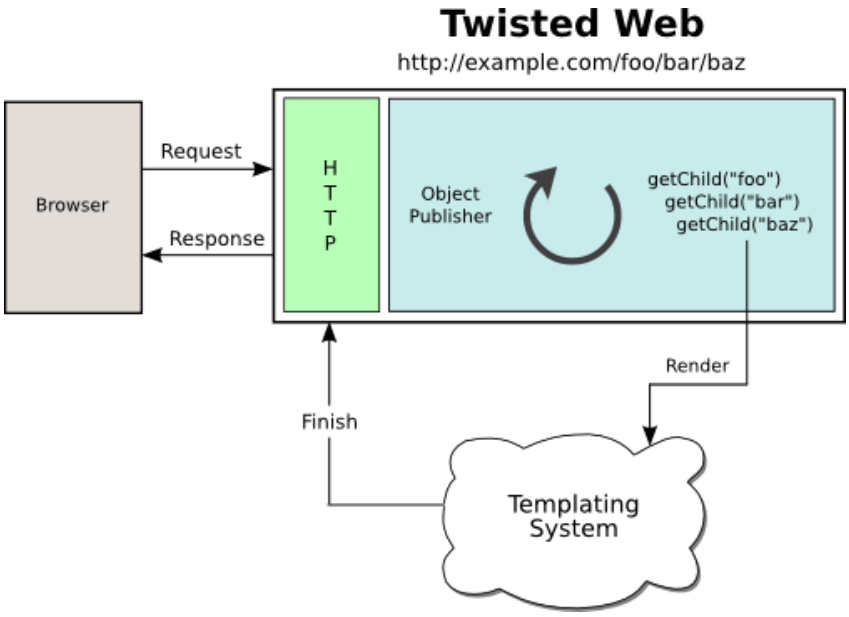
Samotné vytvorenie šablón budeme realizovať pomocou rule-based prístupu a teda z analýzy dôkazu zneužiteľnosti zraniteľnosti pomocou preddefinovaných pravidiel vyberieme špecifické úseky kódu, o ktorých vieme, že sú podstatne pre korenú imitáciu zraniteľného zariadenia. Následne tieto úseky upravíme do podoby šablón spracovateľných našim honeypotom. Vo výsledku náš systém bude umožňovať jednoduché rozšírenie o ďalšie šablóny, čo nám umožni rýchle nasadiť honeypot pre nové zraniteľnosti bez toho aby sme museli drasticky meniť, alebo zasahovať do konfigurácie honeypotu a zariadenia na ktorom bude daný honeypot fungovať.

4.3 Vytvorenie honeypotu z nízkou úrovňou interakcie

Pre samotné vytvorenie nášho honeypotu použijeme Twisted Web Server, ktorý je súčasťou frameworku Twisted pre programovací jazyk Python. Twisted Web Server nám umožňuje jednoduché a rýchle vytvorenie vlastného HTTP servera, pričom si vieme daný server upraviť podľa našich potrieb. Vďaka svojej modularite predstavuje Twisted Web Server vhodný nástroj na implementáciu honeypotu pre naše účely [21].

Keďže náš honeypot bude s nízkou úrovňou interakcie, nebudeme musieť implementovať plnú funkcionalitu servera, ale zameriame sa primárne na vytvorenie hlavnej logiky pre výber odpovedí. S využitím našich šablón budeme môcť presne definovať jednotlivé odpovede, ktoré bude náš Twisted Web Server odosielať.

Konfiguráciou hlavičiek vieme upraviť špecifické sekcie s cieľom vytvoriť dojem, že náš server je napríklad Apache Server s konkrétnou verziou. To nám umožní prispôsobiť hlavičky na základe analýzy dôkazov o zneužiteľnosti zraniteľnosti, ktorá už bola vykonaná. Takýmto spôsobom vieme navodiť dojem, že požiadavka bola spracovaná reálnou zraniteľnou službou, aj keď sa v skutočnosti jedná o simulované prostredie.



Obrázok 8: Proces spracovania požiadavky v rámci Twisted Web servera [21].

Z obr. Č. 8 vyplýva, že Twisted Web Server je priamo určený pre konfiguráciu odpovedí pre dané požiadavky, čo priamo vyhovuje našim požiadavkám. Vďaka tejto architektúre je Twisted Web Server dostatočne flexibilný na širokú škálu možných nastavení potrebných pre imitáciu zraniteľného zariadenia.

4.4 Zber dát

Našim cieľom pre túto prácu je vytvorenie honeypotu s nízkou úrovňou interakcie na základe dôkazu zneužiteľnosti zraniteľností. Z dôvodu, že náš honeypot je z nízkou úrovňou interakcie a z predpokladu, že náš honeypot sa budú pokúšať kompromitovať len útočníci, keďže bežný užívateľ nemá potrebu pristúpiť k danému zariadeniu, nie ešte pokúsiť sa využiť špecifickú zraniteľnosť. Z tohto dôvodu budeme primárne zbierať sieťové data, ako sú napríklad IP adresa útočníkovho zariadenia, aký user-agent / prehliadač bol použitý, časové pečiatky, celé HTTP hlavičky a rôzne iné parametre zanechané útočníkom. Keďže bežné nízko úrovňové honeypoty sa taktiež zameriavajú na mapovanie a monitorovanie útočných vektorov než na nejakú hlbokú analýzu krokov a stop zanechaných útočníkom.

Z nášho pohľadu sú pre účely ďalších analýz, hlavne dôležité údaje ako je IP adresa pre možnosti korelácie IP adresy útočníka a ďalšej monitorovanej aktivity v sieti organizácie. Tieto dáta budú ukladane v štruktúrovanej forme, čo nám umožní vykonať ďalšie analýzy nad týmito dátami, bez ďalších zbytočných uprav.

5 Záver

V tejto práci sme sa venovali problematike honeypotov, ktoré sú zamerane na imitáciu zraniteľných zariadení podľa existujúcich dôkazov zneužiteľnosti zraniteľností. V prvej časti práce sme sa zamerali a teoretickú stránku honeypotov, aké sú ich jednotlivé klasifikácie a na aké účely sa honeypoty používajú. Ďalej sme sa podrobnejšie venovali zraniteľnostiam a to presne ich klasifikácii podľa viacerých faktorov. Vysvetlili sme jednotlivé klasifikačné systémy ako sú napríklad CVSS, CVE a CWE a aj samotnú štruktúru modulov v nastrojí Metasploit Framework, ktoré sú písane v programovacom jazyku Ruby.

V časti návrhu riešenia sme vysvetlili proces vytvorenia honeypotu s nízkou úrovňou interakcie, ktorý je postavený na základe dôkazov zneužiteľnosti zraniteľností. Podrobne sme rozpísali jednotlivé fázy analýzy dôkazov na základe štruktúrovaných dôkazov z nástroja Metasploit Framework. Taktiež sme sa venovali procesu vytvárania šablón podľa vykonanej analýzy, ktoré budú použité honeypotom pre korektnú imitáciu zraniteľného zariadenia. V návrhu sme následne popísali samotnú implementáciu honeypotu pomocou využitia Twisted Web Servera, ktorý je písaný v programovacom jazyku Python. Twisted Web Server nám umožňuje jednoduchú a dynamickú konfiguráciu nášho nízko úrovňového honeypotu. Nakoniec sme podrobne vysvetlili metódy zberu dát výhodné pre náš honeypot a požiadavky potrebne pre dostatočne rozsiahle zoznámenia dát získaných z nášho honeypotu.

Aktuálne sa práca nachádza v stave návrhu riešenia, kde sme si určili najbližšie kroky pre realizáciu analýzy dôkazov zneužiteľnosti zraniteľností. Ďalej plánujeme rozšíriť návrh systému o konkrétnu implementáciu šablón a honeypotu pre vybrane dôkazy. Výsledkom našej práce by mal byť systém, ktorý umožní organizácii generovať honeypoty výhradne pomocou aktuálnych dôkazov zneužiteľnosti zraniteľností, čo prispeje k efektívnejšiemu monitorovaniu hrozieb v sieti organizácie.

Použitá literatúra

[1] What is a honeypot? [online] Dostupné z: https://www.kaspersky.com/resource-center/threats/what-is-a-honeypot

[2] Honeypots in cybersecurity explained. [online] Dostupné z: https://www.crowdstrike.com/en-us/cybersecurity-101/exposure-management/honeypots/

[3] Spitzner, L. 2002. Honeypots: Tracking Hackers. 1st ed. Boston, MA, USA: Addison Wesley.

[4] Sokol P, Pekarčík P, Bajtoš B. Data Collection and Data Analysis in Honeypots and Honeynets. [online] Dostupné z: https://web.archive.org/web/20180421062733id_/http://spi.unob.cz/papers/2015/2015-19.pdf

[5] Iyatiti Mokube and Michele Adams. 2007. Honeypots: concepts, approaches, and challenges. In Proceedings of the 45th annual southeast regional conference (ACM-SE 45). Association for Computing Machinery, New York, NY, USA, 321–326. https://doi.org/10.1145/1233341.1233399

[6] What Are Honeypots (Computing)? [online] Dostupné z: https://www.fortinet.com/resources/cyberglossary/what-is-honeypot

[7] Vulnerabilites. [online] Dostupné z: https://nvd.nist.gov/vuln

[8] CIA Triad. [online] Dostupné z: https://www.fortinet.com/resources/cyberglossary/cia-triad

[9] LU, Jiangtao, GU, Min, HU, Jinchang, BAI, Tongtong, HONG, Binsheng a HUANG, Song. The Anatomy of Vulnerability Proof-of-Concept: A Systematic Mapping Study [online]. [s.l.]: SSRN, 2024 [cit. 2025-06-19]. Dostupné na: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5193688

[10] Vulnerability Metrics. [online] Dostupné z: https://nvd.nist.gov/vuln-metrics

[11] Common Vulnerability Scoring System SIG. [online] Dostupné z: https://www.first.org/cvss/

[12] CVE-2021-44228 Detail. [online] Dostupné z: https://nvd.nist.gov/vuln/detail/cve-2021-44228

[13] Common Vulnerability Scoring System v3.1: Specification Document. [online] Dostupné z: https://www.first.org/cvss/v3-1/specification-document

[14] New to CWE. [online] Dostupné z: https://cwe.mitre.org/about/new_to_cwe.html

[15] What is a CVE. [online] Dostupné z: https://www.redhat.com/en/topics/security/what-is-cve

[16] Understanding Exploit Proof-of-Concept. [online] Dostupné z: https://vulncheck.com/blog/understanding-exploit-proof-of-concept

[17] CVE-2021-44228 Metasploit Framework. [online] Dostupné z: https://github.com/tangxiaofeng7/CVE-2021-44228-Apache-Log4j-Rce/blob/main/src/main/java/log4j.java

[18] Metasploit Framework. [online] Dostupné z: https://docs.rapid7.com/metasploit/msf-overview/

[19] log4shell Metasploit Framework. [online] Dostupné z: https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/multi/http/log4shell_header_injection.rb

[20] wpdiscuz Metasploit Framework. [online] Dostupné z: https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/unix/webapp/wp_wpdiscuz_unauthenticated_file_upload.rb#L59

[21]Configuring and Using the Twisted Web Server. [online] Dostupné z: https://twisted.org/documents/15.0.0/web/howto/using-twistedweb