

Detekcia nepovoleného prístupu kustomizáciou nízko interaktívnych honeypotov

Jakub Perdek

Slovenská technická univerzita v Bratislave

Bratislava, Slovensko

perdek.jakub@gmail.com

Abstract—Prevenencia pri zabezpečení systému obvykle nemusí byť dostatočná. Z toho dôvodu je potrebné detegovať aktivity spojené hlavne s neoprávneným prístupom do systému, súborom alebo do intranetu. Ich skoré odhalenie môže pomôcť prijať vhodné opatrenia a zabrániť reálnemu útoku na systém. Vhodným nástrojom sú práve honeypoty. Tie navyše môžu slúžiť aj na spomalenie neoprávnených aktivít alebo na zmiatnutie útočníka. Často sú ale ľahko rozpoznateľné od reálnych aktív, a preto je ich kustomizácia nevyhnutná. Lákanie možno realizovať podľa viacerých stratégií, a podľa toho aj prispôbiť generovanie kustomizovateľných tokenov. Predstavujeme preto automatické riešenie pre tvorbu nízko interaktívnych honeypotov založené na čo najväčšej infiltrácii logiky pre sledovanie rozličných foriem manipulovania s nimi v rámci konkrétnej biznis logiky. Snahou je poskytnúť informácie pri záujme o konkrétny obsah. Analyzované sú preto rôzne spôsoby aplikovania uvedených mechanizmov v rámci webových dokumentov. Možnosti pre zamedzenie ich zneužitia a obmedzenie manipulácie s nimi by v rámci tohto druhu honeypotov mali byť ľahšie dosiahnuteľné a prispôbené pre konkrétny prípad použitia.

Index Terms—nízko interaktívne honeypoty, detekcia nepovoleného vstupu, kustomizácia honeypotov, webové honeypotokeny

I. Úvod

Honeypoty sú bezpečnostným zdrojom, ktorý generuje upozornenie pri zachytení nadviazania interakcie s ním [2]. Napríklad v podobe prieskumu, útoku alebo pri kompromitácii. Často lákajú infiltrovaný subjekt, a pri ich dobrom maskovaní a umiestnení môžu pomôcť odhaliť neoprávnený prístup, odhalené prístupové údaje, dokonca spomaliť aktivity útočníka vrátane odhalenie nultého útoku a rovnako aj zistiť niektoré ním aplikované postupy pri útoku. Bežní používatelia by nemali s honeypotom vôbec interagovať, ale ani vedieť o tejto funkcionalite. Prihlásenie sa do takéhoto systému alebo otvorenie a manipulácia s dokumentom, respektíve honey tokenom [1] je preto

automaticky podozrivá a malo by pomocou oznámení byť na ňu upozornené.

II. Honeypoty pre detekciu neoprávneného prístupu

III. Benefity nízko interaktívnych honeypotov pre detekciu neoprávneného prístupu

Medzi úrovňou interakcie honeypotov je často potrebné robiť trade-off. Napríklad nízko úrovňové honeypoty je jednoduché nainštalovať, ľahko emulujú niektoré služby. Riziko je tak nízke ale rovnako je limitovaná aj poskytnutá informácia o aktivite útočníka [4]. Často nevyžadujú zmeny v existujúcej topológii siete alebo na zariadeniach [3]. Nízko interaktívne honeypoty sú väčšinou produkčné honeypoty určené na ochranu organizácie [4].

IV. Návrh nízko interaktívneho honey tokenu z webových dokumentov

Nízko interaktívne honeypoty pre detekciu neoprávneného prístupu by mali zahŕňať mechanizmy pre generovanie informácie o príslušnej udalosti. Tie by mali byť dostatočne maskované, tak aby sa útočník o svojom odhalení najlepšie ani nedozvedel. V svojej práci sme sa zamerali na webové dokumenty.

A. Mechanizmy pre logovanie aktivity s webovým tokenom

Analyzovali sme rôzne spôsoby vloženia spomenutého mechanizmu do webového dokumentu. Zamerali sme sa na prípady, keď sa manipuluje s webovým dokumentom neposkytovaným priamo v rámci nejakého spusteného servera. Výsledný honeypotoken v podobe konkrétneho dokumentu bude poskytovať obmedzené možnosti interakcie s ním, a tým sa zabráni aj jeho zneužitiu. Identifikovali sme nasledovné prípady umožnenie sledovania interakcie s uvedeným dokumentom:

- 1) Použitie nástroja sledujúceho zmeny v rámci súborového systému, prípadne iného systému v rámci ktorého sa uvedené tokeny manažujú, pre vybrané súbory. Získanou informáciou potom je len informácia o záujme o tieto súbory. Detekovanie takejto zmeny môže byť aj jednoduchšie detegovať.
- 2) Vloženie, respektíve nahradenie alebo upravenie existujúceho iframe elementu novým. Infiltrovanej osobe sa tým priamo poskytne hľadaný obsah aj s odoslaním informácií pri dopytoch po zdrojových súboroch nového obsahu. Zároveň možno odpozorovať o aký obsah má osoba záujem a príslušne tento obsah kustomizovať. Aj napriek tomu, že sa iframy už neodporúča používať stále sa vyskytujú hlavne pri službách poskytujúcich letáky v internetových obchodoch.
- 3) Vytvorenie a odoslanie informujúcej správy, respektíve logu na server. Správu by malo byť potrebné maskovať. Oproti predchádzajúcemu spôsobu môže byť takáto správa odchytená a jej odoslanie na server útočníkom znemožnené, keďže sama o sebe spravidla neposkytuje pre útočníka relevantný obsah. Mala by preto byť prepojená s vyžiadaním konkrétnych kľúčov pre prístupenie tohto obsahu.
- 4) Využitie proxy, ktoré umožňuje získať a pozmeniť časť pravého obsahu pre nalákание útočníka.
- 5) Zahashovanie a zneprístupnenie obsahu až do momentu priameho vykonania skriptov na stránke. Inak by útočník mohol získať želaný obsah iba otvorením konkrétneho súboru.

Vytvorenie kópie webovej lokality sme realizovali pomocou balíčkov prístupných v jazyku python. Získali sme ale ich obmedzenú funkčnosť pri reálnom použití. V prípade balíčku pywebcopy ¹ to bolo časté zamrznutie skriptu kvôli zlúčeniu vlákien. Použili sme preto najnovšiu verziu priamo z githubu ² a zablokovali použitie multivláknového spracovania ale vyskytol sa nový problém s absolútnymi adresami skopírovaného dokumentu. Kvôli nim sa napríklad nezobrazia pôvodné obrázky na stránke. Ďalším nástrojom bolo vloženie podpory pre HTTrack ³, ktorý je efektívnejší a nemal komplikácie ako predchádzajúci balík. Jeho použitie je ale závislé na operačnom systéme a v súboroch necháva informácie o použití tohto nástroja.

Príklad použitia nástroja HTTrack pre získanie súborov do hĺbky 1, bez externých závislostí, a ich uloženie do priečinku ./downloads:

```
"C:\\Program Files\\WinHTTrack\\httrack.exe"
https://www.trony.it/online/store-locator -v -r1
-%e0 -O ./download
```

Obmedzenie na menované spôsoby je hlavne z dôvodu ochrán pred internetovými hrozbami, v rámci ktorých nie je povolené zapisovať na disk pomocou prehliadača štandardným spôsobom. Rovnako ukladanie je obmedzené na použitie cookies, ktoré si používateľ spravidla môže prezeráť. Pri aplikácii preto zostáva informovať s použitím naviazania na zdroje, ktoré sú ku dokumentu priradené.

B. Nástroje pre maskovanie logiky honeypotu a ich význam pre kustomizáciu

Pri tvorbe funkcionality umožňujúcej detekovať manipuláciu s týmito tokenmi a ich ďalšiu kustomizáciu sme použili nástroje pre:

- 1) Vytvorenie kópie danej webovej lokality, prípadne iba jej časti.
- 2) Manipulovanie so štruktúrou webového dokumentu a automatickú zmenu informácií v rámci nej.
- 3) Minifikáciu a zamlženie súborov pre kaskádové štýly, skripty pre javascript a hlavne HTML dokumenty.

Pokiaľ je možné tak manipulujeme so štruktúrou webového dokumentu dynamicky s použitím knižnice BeautifulSoup. Typickou požiadavkou je upravenie konkrétnych iframe elementov a nastavenie ich atribútov. Niekedy je potrebné funkcionality využiť pre prepojenie aj upravených alebo vytvorených skriptov alebo kaskádových štýlov s webovým dokumentom. Pokiaľ načítanie tejto štruktúry nie je možné malo by byť zabezpečené pridanie uvedených častí v textovej podobe.

Do riešenia sme pridali niekoľko nástrojov pre zamlženie a minifikáciu súborov. Rôzne nástroje poskytovali funkcionality pre rozdielne typy súborov. Pre minifikáciu HTML sme použili pythonovskú knižnicu htmlmin ⁴ s nastavením pre vymazanie prázdneho miesta a odstránenie komentárov. Pre minimalizáciu

¹<https://pypi.org/project/pywebcopy/>

²<https://github.com/rajatomar788/pywebcopy>

³<https://www.httrack.com>

⁴<https://pypi.org/project/htmlmin>

ciu javascriptu sme použili Closure Compiler ⁵ od Googlu. V rámci nastavení sme použili compilation-level na SIMPLE_OPTIMIZATIONS, kvôli tomu, že pri pokročilejšom nastavení dochádzalo k odstráneniu všetkého kódu pokiaľ nebol priamo exportovaný. Redundantné funkcie určené na zneprehľadnenie kódu by tak boli odstránené. Následne sme pridali parameter pre spracovanie aj javascriptovských súborov v striktnom móde. Minimalizáciu kaskádových štýlov sme zabezpečili pomocou externého nástroja Yui Compressor ⁶ volaného podobne z príkazového riadku. Jednotlivé nástroje sme aplikovali na príslušné súbory v konkrétnej vybranej stromovej štruktúre. V praxi takýmito súbormi sú súbory klonovanej webvej lokality so zapracovanou sledovacou logikou.

Ďalšou voliteľnú časť tvorí použitie proxy pre poskytnutie časti relevantného obsahu s dodatočnou možnosťou úpravy. Funkcionalitu sme sa rozhodli pridať kvôli uľahčeniu procesu kopírovania časti namiesto celej webovej lokality, keďže v praxi táto činnosť vyžadovala kopírovanie obsahu aj z externých stránok. Reštriktívny je CORS hlavne pri zabezpečenejších aplikáciám. Namiesto proxy je preto výhodnejšie poskytovať celý fiktívny obsah pre honeypot priamo. Pri využití proxy možno reálne dáta z existujúcich služieb upravovať za behu, ale rastie riziko odhalenia takéhoto honeypotu.

Podstatnými sú nástroje pre maskovanie obsahu. Dáta je možné uložiť do viacerých častí, ktoré sa v priebehu vykonania zložia. To núti používateľa aby súbor reálne otvoril v prehliadači, nechal vykonať kód a tým zapríčinil aj vykonanie skrytej sledovacej logiky. Opäť by sprístupnenie obsahu malo spočívať na komunikácii so serverom, inak sledovacia logika neoznámi manipuláciu s tokenom. Rovnako je možné použiť base64 hash, ktorým ukryjeme obsah a hlavne sledovaciu logiku pred vyhľadaním zvonka. Alternatívou je prevod do hexadecimálneho formátu alebo použitie knižnice pre obalenie dát. Existujú rôzne formáty, používaným môže byť gzip.

V. Automatizácia kustomizácie honeytokenu

V rámci automatizácie a kustomizácie honey tokenu riešime dve stratégie. V rámci prvej používateľ špecifikuje počet výrazných honeytokenu a doménu z ktorej majú byť vytvorené. V rámci druhej stratégie automatizácia a kustomizácia nebude zvyrazňovať

určité charakteristiky cieľového súboru a jeho obsahu ale vytvorí dostatočný počet takýchto podobných tokenov.

V rámci samotnej automatizácie sa vykonávajú kroky v nasledovnej postupnosti:

- 1) Klonovanie webových dokumentov z danej domény. Zváži sa tu aj klonovanie nepovolených stránok v rámci súboru robots.txt a hĺbka tohto klonovania.
- 2) Pre každý generovaný prvok sa rovnako vykoná nasledovný postup. Vloží sa element so sledovacím iframe, pri ktorom server loguje požiadavky pre dopytovanie sa po jeho obsahu alebo sa nahradí existujúci iframe aj spolu s presunutím poskytovania tohto obsahu na server. V prípade nepoužitia iframu sa vygeneruje a vloží kód odosielať informácie pri vyžiadaní konkrétneho obsahu.
- 3) Klonovanie webovej lokality na adrese, na ktorú smeruje iframe a tvorba API pre spracovanie funkcionality poskytovanej konkrétnym iframom. Obsah v rámci biznis domény uložený v rámci konkrétnych súborov by mal byť ďalej kustomizovaný. V prípade nepoužitia iframu sa vygeneruje kostra servera poskytujúceho kľúče pre sprístupnenie obsahu z klienta a odmaskovanie správy pre jej zalogovanie.
- 4) Ďalšie hashovanie a zneprístupňovanie obsahu v rámci súborov. Zabezpečenie funkčnosti pri vykreslení stránky a aplikovaní kľúčov zo serveru.
- 5) Prípadné pridanie redundantného kódu alebo spájanie a vykonanie zamlženej verzie.
- 6) Rovnako pre každý generovaný honeytokenu prebehne minifikácia a prípadné zamlženie skriptov, html súborov a asociovaných kaskádových štýlov.

VI. Návrh a implementácia nástrojov pre utajenie obsahu

Ukrytie obsahu konkrétnych správ ale aj celých skriptov môže byť v rámci statických súborov nevyhnutné. Zároveň je potrebné aplikovať takýto postup na každú zahrnutú biznis informáciu, ktorá by neskôr mala byť sprístupnená a prezentovaná v rámci konkrétneho honeytokenu. Sprístupnia sa potom už iba konkrétne fiktívne biznis informácie. Útočník by si konkrétnych logov a ich obsahu potom nemal všimnúť. Rovnako by nemal ani neblokovať požiadavky smerujúce na server kvôli jeho potencionálnemu

⁵<https://github.com/google/closure-compiler>

⁶<https://github.com/yui/yuicompressor>

záujmu o biznis obsah. Uvedená funkcionálnosť by mala zabezpečiť rôzne druhy informácií odosielané z rôznych častí webového dokumentu a zvýšiť tak možnosti informovania o činnosti útočníka.

A. Utajenie obsahu kompresiou a kódovaním

Pre budúci generátor honey tokenov sme navrhli funkcionálnosť, ktorá bude obsah niekoľko ráz maskovať s použitím náhodne vybraných metód uplatnených aj niekoľko krát. Výsledkom bude maskovaný obsah a kľúčom bude postupnosť identifikátorov metód, aby následne mohli byť použité pre navrátenie textu do čitateľnej podoby. Funkcionálnosť sme zostrojili tvorbou triedy podporujúcej dve základné metódy pre maskovanie a odmaskovanie. Tie konkrétne implementácie dediace od nej prekryli vlastnými využívajúcimi svoju funkcionálnosť a umožnili tak náhradu za túto základnú triedu, čo dopomohlo k náhodnému výberu konkrétnej metódy. V základnej triede sme implementovali nástroje pre náhodný výber metódy, získanie metódy podľa identifikátora, funkcionálnosť umožňujúcu maskovať a rovnako odmaskovať konkrétny obsah.

Pri implementácii konkrétnych maskovacích metód sme využili knižnice tretích strán. Okrem základnej funkcionality umožňujúcej vytvoriť z textu hash ako napríklad base64, base32, base16 alebo base85 (knižnica base64 ⁷) sme implementovali aj podporu pre kompresné metódy ako je gzip (knižnica gzip ⁸), deflate (knižnica zlib ⁹) alebo brotli (knižnica brotli ¹⁰). Vstupom ako aj výstupom každej z týchto metód je textový reťazec. Problémom kompresných metód je vrátenie výslednej hodnoty v binárnej podobe. Bolo preto zaručiť konverziu do textovej podoby. To ale v rámci kódovania utf-8 nebolo kvôli rôznym nevyhovujúcim znakom možné. Pri vynechaní chýb nebolo spätné odmaskovanie úplne úspešné a uplatnenie viacerých metód ani možné. Rovnako pretypovanie spôsobovalo chyby pri integrácii spomenutých kompresných metód. Nakoniec sa nám problém podarilo vyriešiť nástrojom pre quotovanie url (knižnica urllib ¹¹), ktorý to dokázal urobiť aj z binárnej podoby (metóda pre prevod z binárnej podoby na text má podobu `urllib.parse.quote_from_bytes(compressed_bytes)`

⁷<https://docs.python.org/3/library/base64.html>

⁸<https://docs.python.org/3/library/gzip.html>

⁹<https://docs.python.org/3/library/zlib.html>

¹⁰<https://pypi.org/project/Brotli>

¹¹<https://docs.python.org/3/library/urllib.html>

a metóda pre spätný prevod má podobu `urllib.parse.unquote_to_bytes(encoded_text)`).

Ako ďalšie metódy pre maskovanie textu sme použili funkcionálnosť pre quotovanie url (`urllib.parse.quote_plus(pure_text)`) a jeho spätné dekódovanie (`urllib.parse.unquote_plus(encoded_text)`).

Nakoniec sme implementovali spomenuté metódy pre maskovanie a odmaskovanie. Metóda pre maskovanie získa na vstupe zvolený text, pole maskovacích metód a počet uplatnených cyklov. Následne náhodne vyberie z tohto poľa maskovaciu metódu a uplatní ju na zvolený text. Zaznamená sa aj druh tejto metódy podľa identifikátora tejto metódy. Postup sa opakuje zvolený počet krát vždy s predtým maskovaným textom. Výsledkom je maskovaný text a identifikátor všetkých použitých maskovacích metód ako kľúč. Ten sa môže umiestniť na serveri a zabezpečiť tak utajenie obsahu, alebo zaručiť odmaskovanie už na klientovi s nevyhnutnou potrebou maskovania tohto kľúču. V rámci uvedenej funkcionality je predpoklad odmaskovania na klientovi dôležitý, inak by postačovala aj iná forma šifrovania, pri ktorej by server bol schopný sprístupniť konkrétne správy, respektíve informácie.

Metóda na odmaskovanie na vstupe požaduje maskovaný text a identifikátor všetkých použitých metód pre maskovanie. V rámci tejto metódy sa vykoná reverzná aplikácia uvedených metód v opačnom poradí ako boli použité pri maskovaní. Výsledkom by mal byť odmaskovaný text v podobe v akej bol pred jeho utajením.

B. Zamlženie kódu jeho rozdelením a vykonaním v rámci druhého kódu

V praxi okrem štandardnej minifikácie kódu zahŕňajúcej náhradu premenných a odstránenia prázdneho miesta môže byť užitočné aj rozdelenie kódu na viaceré časti. Ak sa pred rozdelením aplikuje maskovanie obsahu z predchádzajúcej časti dostávame tak dômyselne utajený obsah. Týmto obsahom by mala byť najčastejšie správa o postupe útočníka ale môže ním byť aj zdrojový kód samotný. V prípade takéhoto spracovania zdrojového kódu samotného ako reťazca je potrebné vykonanie metódy eval. V rámci nej sa zdrojový kód zapísaný ako textový reťazec vykoná. Použitie tejto metódy je stále veľkým bezpečnostným riskom, a preto by sa malo takému použitiu vyhnúť. Ideálne kód predĺžiť rozdelením jednotlivých príkazov, lepším prepletením kódu s bizis logikou alebo pridaním nepotrebných a redundantných častí.

V rámci našej implementácie sme umožnili rozdeliť textový refazec na viaceré časti s použitím premenných a polí. Do polí sú tieto časti pridávané priamo alebo vo forme premenných, ku ktorým sú predtým priradené. Následne je predtým rozdelený refazec znovu zložený. Jednotlivé príkazy sú generované zaradom a ukladané do poľa. Náhodne sa pritom aplikujú implementované metódy pre generovanie príkazov ako napríklad tvorba premennej vo forme textového poľa alebo textového refazca a zlúčenie premenných. Dodatočne je možné pridávať medzi tieto príkazy aj iné redundantné príkazy alebo príkazy prislúchajúce ku konkrétnej biznis logike. Funkcionalitu sme testovali s aplikovaním maskovania obsahu a testovali s vykonaním javascriptu pomocou knižnice js2py¹². Výstupné texty boli zhodné. Identifikovali sme aj problém s použitím deklarácií s použitím let, pri ktorých bolo potrebné konkrétny kód obaliť do funkcie inak mohlo dochádzať ku kolízii premenných. Použitie var namiesto let by problém vyriešilo, ale mohlo by spôsobiť aj problémy a malo by byť používané len výnimočne pre globálne deklarácie.

VII. Vloženie detekčnej logiky

Detekčná logika je hlavným obsahom honeytokenu. Do dokumentu musí byť vhodným spôsobom pridaná a následne je potrebné zabezpečiť zneprístupnenie informácií a zamedziť jej odhalenie.

A. Vloženie detekčného skriptu

Podstatným pre využitie a prepojenie predtým realizovaných metód pre klonovanie, minimalizáciu a utajovanie obsahu je samotné vloženie správy alebo skriptu. To musí byť maskované vzhľadom na požiadavku utajene informovať o konkrétnom obsahu. Celý proces realizujeme v niekoľkých krokoch, pričom využívame náhodné generátory pre vygenerovanie rôznych parametrov určujúci finálny výsledok:

- 1) Príprava skriptu pre odosielanie hlásení na server. Dodatočne je možné pripraviť aj zoznam správ, ktoré sa budú posielať.
- 2) Príprava webových služieb pre manažovanie logov a samotnej kostry serveru.
- 3) Klonovanie príslušnej časti webovej lokality.
- 4) Načítanie skriptu pre detegovanie podozrivej aktivity, prípadne aj samostatných správ.
- 5) Zamaskovanie buď celého skriptu, pričom v neskorších fázach bude potrebné použiť funkciu

eval pre jeho vykonanie po odmaskovaní. Dôležité je maskovanie jednotlivých správ samostatne kvôli ich utajeniu. Tie budú následne počas vykonávania odosielané.

- 6) Vloženie metód pre dešifrovanie konkrétneho obsahu s kódom metódy, ale nie samotných správ s dôrazom na maskovanie identifikátora použitých metód. Implementovanie tejto funkcionality priamo na serveri z dôvodu zistenia obsahu logov a ich následného spracovania.
- 7) Rozdelenie obsahu do niekoľkých riadkov kódu, prípadne metód. Vhodné je aj zahrnúť tento obsah do kódu pre biznis logiku alebo pridať redundantný kód.
- 8) V prípade maskovania celého skriptu je potrebné po domaskovaní zabezpečiť volanie funkcie eval pre jeho vykonanie.
- 9) Vloženie celého kódu do script elementu a pripojenie ho do honeytokenu, respektíve webového dokumentu.
- 10) Otestovanie spustením servera a otvorením honeytokenu. Odoslané hlásenia by mali byť zaznamenané.

B. Injekcia iframe elementu s konkrétnym obsahom

Chýbajúci dynamický obsah možno v rámci statického obsahu honeytokenu vložiť ako vnorený dokument pomocou iframe elementu. V konkrétnom dokumente konkrétnej webovej lokality môžeme nahradiť už nejaký existujúci alebo vložiť nový. Pri nahradzovaní existujúceho by sme mali nahradiť aj celú poskytovanú biznis logiku. V niektorých prípadoch pre tieto účely možno použiť proxy server. Druhou možnosťou je propojiť iframe element, ktorý sa nemusí ani zobrazovať. V tomto prípade vzrastá potencionálna hrozba detegovateľnosti honeypotu. Poskytnutie reálneho obsahu tak môže byť spoľahlivejšie a zaručí to aj zabránenie blokovaniu požiadaviek na server. Aj v tomto prípade uvádzame postup tvorby takéhoto honeytokenu s dôrazom na náhodné určenie jednotlivých parametrov:

- 1) Klonovanie príslušnej časti webovej lokality s webovým dokumentom.
- 2) Vyhľadanie iframe elementu a zamenenie src atribútu za vlastnú webovú službu. Ak taký element neexistuje, alebo je komplikované napodobniť inú lokalitu, potom sa vytvorí nový iframe element s nastavenými rozmermi na 0.
- 3) Pokiaľ bol nahradzovaný obsah iframe elementu, potom sa zrealizuje klon webovej lokality na ktorú smeroval jeho pôvodný obsah a vloží sa

¹²<https://pypi.org/project/Js2Py/>

do zdrojov servera alebo sa prispôsobí server tak aby robil proxy medzi pôvodnou lokalitou. V rámci proxy sa niektoré informácie môžu pozmeniť, aby útočník nedostal reálne biznis dáta.

- 4) Doplnenie potrebného obsahu pre logovanie a ďalších potrebných služieb (napríklad prípadné proxy) na server.
- 5) Maskovanie zvyšného obsahu vyššie opísanými technikami pre zmiatnutie útočníka.
- 6) Otestovanie funkcionality spustením servera a otvorením honeytokenu. Správa o prístupe k danému obsahu by mala byť zaznamenaná.

VIII. Automatizované generovanie honey tokenov

Po návrhu kustomizovateľného honey tokenu bolo potrebné realizovať aj ich masové generovanie zosúladené so zvolenou stratégiou. Vhodné je zabezpečiť aj rozšíriteľnosť pre aplikovanie ľubovoľnej stratégie. Ich tvorbu sme preto rozdelili na niekoľko častí. V prvej fáze sa podľa špecifických kritérií vytvorí JSON konfiguračný súbor s parametrami určujúcimi základné komponenty a ich orientáciu. Túto fázu je možné automatizovať a po vytvorení príslušného súboru manuálne zrevidovať a upraviť výsledné hodnoty. Druhou fázou je samotné vytvorenie konkrétnych inštancií honey tokenov a serverov podľa vytvorenej konfigurácie. Následne rovnako možno manuálne upraviť výsledné časti, tak aby pôsobili čo najdôveryhodnejšie.

Zabezpečenie kustomizácie API pre komunikáciu klienta a serveru v rámci zvolenej stratégie vyžaduje určenie konkrétnych parametrov v konfigurácii pre obidve časti zároveň. Príkladom môže byť konkrétna URL na ktorej bude server získavať konkrétne informácie od klienta. Server môže na tejto adrese sprístupňovať zvolený obsah, odmaskovávať skrytý obsah správ a rovnako logovať prichádzajúce správy. Klient musí byť schopný odoslať správu vo vopred navrhnutom formáte na túto URL. Pre každý honey token sa zvolí reprezentujúci odkaz na existujúci webový dokument podľa ktorého sa má vytvoriť. Už v tejto fáze je možné rozhodnúť o minifikácii vybraného obsahu a ďalších doplnkoch a zmenách. Okrem samotného prispôsobenia a falšovania webového dokumentu je pri konkrétnych stratégiách potrebné zvoliť aký skript sa má použiť pri komunikácii so serverom a zároveň pomocou ďalších metód aj zamaskovať. Skript by mal byť písaný v Javascripte a obsahovať pre framework rozpoznateľné pomenovania dôležitých častí, ktoré v rámci

konkrétneho použitia budú za behu nahradené za hodnoty z konfiguračného súboru. Príkladom takejto časti môže byť už spomínaná URL adresa, ale aj reťazec identifikujúci aplikované maskovacie metódy. Podobným spôsobom sa vytvorí aj funkcionality samotného serveru. Server môže byť vytvorený v ľubovoľnom programovacom jazyku podporujúcom nástroje pre komunikáciu s týmito dokumentami. Viacero honey tokenov by malo byť schopné komunikácie s jedným serverom. V každom z dokumentov najvyššej úrovne v konfiguračnom JSON súbore sa špecifikujú honey tokeny a práve jeden server s príslušnými rozhraniami pre ne. Vzhľadom na potrebu odovzdania informácií výhradne sieťovou komunikáciou je prítomnosť servera nevyhnutná.

Nami vytvorené riešenie podporuje automatické generovanie častí NodeJS servera. V jednoduchom prototype sa skopíruje nevyhnutná funkcionality a postupne sa pridávajú ďalšie časti, ktoré sa podľa potreby upravujú podľa hodnôt v konfiguračnom súbore. Samotné metódy pre spracovanie prichádzajúcich dopytov by mali byť rovnako napísané v oddelenom súbore s rozpoznateľnými hodnotami pre používaný rámec. Kód z týchto súborov rámec podľa konfigurácie zahrnie na príslušné miesto. Pri realizácii konkrétnej stratégie je tak potrebné napísať niekoľko takýchto súborov s príslušnou funkcionality.

Po vygenerovaní príslušných častí je potrebné ešte doinštalovať potrebné balíčky pre spustenie servera a prípadne vytvorené tokeny vhodne umiestniť. Samotné umiestnenie tokenov a servera je možné nastaviť aj v konfiguračnom súbore. V prípade NodeJS servera je pre inštaláciu možné použiť príkaz

```
npm install
```

a pre jeho spustenie

```
npm start
```

Vhodná konfigurácia môže pomôcť hlavne pri tvorbe výrazných honey tokenov lákajúcich svojím vzhľadom. Môže to byť napríklad dodatočné generovanie odkazov na konkrétny token alebo napísanie názvu súboru veľkými písmenami. Prevažná väčšina ostatných súborov by potom už nemala mať podobné prvky a pôsobiť tak pre útočníka nezaujímavo. Iná stratégia založená na podobnosti väčšiny zo súborov je obvykle priamo podporená v tvorbe konfiguračného súboru, keďže existujúce dokumenty obvykle možno rýchlo získať priamo z webovej domény, a zároveň na

nich netreba uplatniť kustomizáciu vo vyššej miere. Obsah týchto dokumentov by ale mal byť fiktívny, aby dokumenty nemohli byť ďalej zneužívané.

V rámci testovania automatickej tvorby honey tokenov sme napísali krátke konfiguračné súbory pre obidve stratégie. Generátor tak vytvoril 3 tokeny, pričom sa pre každý uplatnil osobitný postup jeho tvorby. Postup je opísaný v predchádzajúcich kapitolách. Zároveň sa pre zadanú skupinu vytvoril server s možnosťou prijímať a spracovať dopyty od týchto tokenov. Vytvorenému serveru sme doinštalovali potrebné balíčky a spustili ho. Následne sme otvárali jednotlivé honey tokeny v prehliadači. V konzole serveru následne po každom otvorení takéhoto súboru pribudol jeden záznam. Po otvorení súborov bol obsah skriptov roztrieštený v kóde a na obsah boli aplikované niektoré maskovacie metódy. Tokeny vyzerali rovnako ako pôvodné webové dokumenty.

IX. Zhrnutie a budúca práca

References

- [1] C. K. Ng, L. Pan, and Y. Xiang. Honeypot Frameworks and Their Applications: A New Framework. SpringerBriefs on Cyber Security Systems and Networks. Springer Singapore, Singapore, 2018.
- [2] C. Sanders. Intrusion detection honeypots: detection through deception. 2020. OCLC: 1293961192.
- [3] C. Scott. Designing and Implementing a Honeypot for a SCADA Network. page 39, 2014.
- [4] L. Spitzner. Honeypots: Tracking Hackers. Addison-Wesley Longman Publishing Co., Inc., USA, 2002.