# Detekcia nepovoleného prístupu kustomizáciou nízko interaktívnych honeypotov

Jakub Perdek
*Slovenská technická univerzita v Bratislave*
Bratislava, Slovensko
perdek.jakub@gmail.com

***Abstract*—Prevencia 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 naviac 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 prispôsobiť 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 záujeme 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 prispôsobené pre konkrétny prípad použitia.**

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

## 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

## 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 honey token 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, ža 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 predchádajúcemu 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 sprístupnenie tohto obsahu.

Obmedzenie na menované spôsoby je hlavne z dôvodu ochrán pred internetovými hrozbami,v rámci ktorých nie je povolené zapisovať na disk pomocou prehliadaču štandardným spôsobom. Rovnako ukladanie je obmedzené na použitie cookies, ktoré si používateľ spravidla môže prezerať. 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.
4) Využitie proxy, ktoré umožňuje získať a pozmeniť časť pravého obsahu pre nalákanie útočníka.
5) Zahashovanie a zneprístupnenie obsahu až do momentu priameho vykonania skriptov na tránke. Inak by útočník mohol získať želaný obsah iba otvorením konkrétneho súboru.

Vytvorenie kópie webovej lokality sme realizovali pomocou balíčkov prístupných v jazyku python. Zistili 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
```

Pokiaľ je možné tak manipulujeme so štruktúru webového dokumentu dynamicky s použitím knižnice Beautifulsoup. Typickou použiadavkou je upravenie konkrétnych iframe elementov a nastavenie ich atribútov. Niekedy je potrebné funkcionalitu 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 minfikáciu súborov. Rôzne nástroje poskytovali funkcionalitu pre rozdielne typy súborov. Pre minifikáciu HTML sme použili pythonovskú knižnicu htmlmin [4] s nastavením pre vymazanie prázdneho miesta a odstránenie komentárov. Pre minimalizáciu 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 odstránenie 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 Compres-

[1] https://pypi.org/project/pywebcopy/
[2] https://github.com/rajatomar788/pywebcopy
[3] https://www.httrack.com
[4] https://pypi.org/project/htmlmin
[5] https://github.com/google/closure-compiler

sor [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 honeytokenov a doménu z ktorej majú byť vytvorené. V rámci druhej stratégie automatizácia a kustomizácia nebude zvýrazňovať určité charakteristiky cieľového súboru a jeho obsahu ale vytvorí dostatočný počet takýchto podobných tokenov.

V rámci samotnej automatizácie sa vykonajú 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 odosielajúci 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) Rovnako pre každý generovaný honeytoken prebehne minifikácia a prípadné zamlženie skriptov, html súborov a asociovaných kaskádových štýlov.

[6] https://github.com/yui/yuicompressor

## VI. 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.