VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

NÁSTROJ PRO DETEKCI PŘIHLAŠOVACÍCH FORMULÁŘŮ

LOGIN FORMS DETECTION TOOL

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. Jakub Sohr

AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. Zdeněk Martinásek, Ph.D.

SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Bc. Jakub Sohr ***ID:*** 198044

Ročník: 2 ***Akademický rok:*** 2022/23

NÁZEV TÉMATU:

Nástroj pro detekci přihlašovacích formulářů

POKYNY PRO VYPRACOVÁNÍ:

Hlavním cílem práce je návrh a implementace nástroje pro detekci přihlašovacích rozhraní webové aplikace při penetračním testu. Nástroj bude pomáhat testerovi při testování webových stránek velkého rozsahu a identifikovat stránky s možností přihlášení, na které se tester musí zaměřit podrobněji. V teoretické části práce realizujte analýzu současného stavu problematiky, zaměřte se na identifikaci přihlašovacích formulářů. Navrhněte a implementujte vlastní programové vybavení v jazyce Python. Program bude možné spouštět z příkazové řádky včetně podpory výstupu ve formátu JSON. Vlastní nástroj otestujte a přehledně analyzujte dosažené výsledky.

DOPORUČENÁ LITERATURA:

[1] SHAH, S.; MEHTRE, B. M.; CHU, B. T. B.; JONES, M. An overview of vulnerability assessment and penetration testing techniques. In Journal of Computer Virology and Hacking Techniques. 2015, 11(1), 27–49. ISSN 2263-8733.

[2] VATS, P.; MANDOT, M.; GOSAIN., A. A Comprehensive Literature Review of Penetration Testing & Its Applications. In 2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO). 2020, s. 674–680. ISBN 978-1-7281-7016-9.

Termín zadání: 2.2.2023 ***Termín odevzdání:*** 19.5.2023

Vedoucí práce: doc. Ing. Zdeněk Martinásek, Ph.D.

Konzultant: Jaroslav Nespěšný NÚKIB

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRAKT

Rychlý rozvoj internetu a rostoucí složitost webových aplikací vedly k rostoucí potřebě robustních a uživatelsky přívětivých nástrojů kybernetické bezpečnosti. Tato práce představuje návrh, implementaci a vyhodnocení webového nástroje pro detekci přihlašovacích rozhraní, jehož cílem je pomoci bezpečnostním specialistům při identifikaci a posuzování potenciálních přihlašovacích rozhraní na různých webových stránkách. Automatizací tohoto procesu se nástroj snaží zefektivnit identifikaci možných bezpečnostních zranitelností a pomoci při penetračním testování.

Webové rozhraní nástroje bylo vyvinuto s využitím moderních frontendových technologií a frameworků, jako jsou Vue.js a Socket.IO, aby poskytovalo intuitivní a responzivní uživatelské prostředí. Tato práce popisuje principy návrhu a prvky uživatelského rozhraní, které byly použity, stejně jako implementaci základních frameworků Vue.js a Socket.IO pro komunikaci mezi klientem a serverem v reálném čase.

Prostřednictvím komplexního zkoumání návrhu a implementace nástroje pro detekci přihlašovacích rozhraní tato práce demonstruje potenciál moderních technologií při vývoji pokročilých nástrojů kybernetické bezpečnosti.

KLÍČOVÁ SLOVA

penetrační testování, přihlašovací rozhraní, python, webová bezpečnost, detekce, sken zranitelností, websraping, spidering

ABSTRACT

The rapid growth of the internet and the increasing complexity of web applications have resulted in a rising need for robust and user-friendly cybersecurity tools. This thesis presents the design, implementation, and evaluation of a Web-based Login Interface Detection Tool, which aims to assist security professionals in identifying and assessing potential login interfaces on a variety of websites. By automating this process, the tool seeks to streamline the identification of possible security vulnerabilities and assist in penetration testing efforts.

The web interface for the Login Interface Detection Tool has been developed using modern frontend technologies and frameworks, such as Vue.js and Socket.IO, to provide an intuitive and responsive user experience. This thesis describes the design principles and user interface elements that have been employed, as well as the implementation of the underlying Vue.js and Socket.IO frameworks for real-time communication between the client and server.

Through a comprehensive examination of the design, implementation, and security considerations for the Web-based Login Interface Detection Tool, this thesis demonstrates the potential of modern technologies in the development of advanced cybersecurity tools.

KEYWORDS

penetration testing, login interface, python, web security, detection, vulnerability assessment, websraping, spidering

SOHR, Jakub. *Nástroj pro detekci přihlašovacích formulářů*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2023, 54 s. Diplomová práce. Vedoucí práce: doc. Ing. Zdeněk Martinásek, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Bc. Jakub Sohr
VUT ID autora:	198044
Typ práce:	Diplomová práce
Akademický rok:	2022/23
Téma závěrečné práce:	Nástroj pro detekci přihlašovacích formulářů

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu doc.Ing. Zdeňku Martináskovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci. Dále bych rád poděkoval Mgr. Lukáši Neudertovi za spolupráci, návrhy na zlepšení a testování nástroje.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

V dnešním digitálním světě se webové služby staly nedílnou součástí našeho každodenního života, od sociálních médií po online bankovnictví. S pohodlným přístupem k těmto službám odkudkoli však přichází riziko narušení bezpečnosti. Jedním z nejčastějších způsobů narušení uživatelských účtů je napadení přihlašovacích rozhraní. Útočníci využívají různé techniky, například phishing a útoky hrubou silou, aby získali neoprávněný přístup k uživatelským účtům využitím zranitelností na přihlašovacích stránkách.

Pro zmírnění těchto rizik se může ukázat jako cenný přínos softwarový nástroj pro detekci webových přihlašovacích rozhraní. Tento nástroj je určen ke skenování webových stránek a identifikaci potenciálních přihlašovacích rozhraní, která by mohla být cílem útočníků. Díky detekci přihlašovacích rozhraní může tento software pomoci organizacím i jednotlivcům přijmout proaktivní opatření k zabezpečení jejich webových služeb. Ta mohou zahrnovat zavedení silnějších ověřovacích opatření, jako je vícefaktorová autentizace, nebo sledování pokusů o přihlášení kvůli podezřelým aktivitám.

Celkově může softwarový nástroj pro detekci webových přihlašovacích rozhraní hrát zásadní roli při zvyšování bezpečnosti webových služeb, pomáhat předcházet neoprávněnému přístupu a chránit citlivé informace uživatelů.

Cíle práce

Hlavním cílem této práce je řešit problém zlepšení bezpečnosti webových aplikací, který je stále důležitější, protože stále více podniků a jednotlivců se spoléhá na internet při kritických úkonech, jako jsou finanční transakce, zdravotnické služby a komunikace. Jednou z klíčových oblastí zranitelnosti webových aplikací je přihlašovací rozhraní, které je často cílem útočníků snažících se získat neoprávněný přístup k citlivým údajům nebo službám. Cílem této práce je vyvinout softwarový nástroj, který dokáže automaticky detekovat webová přihlašovací rozhraní a identifikovat případné zranitelnosti nebo bezpečnostní rizika spojená s těmito stránkami.

K dosažení tohoto cíle bude softwarový nástroj navržen tak, aby analyzoval webové stránky a detekoval přihlašovací formuláře na základě jejich struktury HTML, obsahu a chování. Nástroj bude vyhodnocen pomocí kombinace ručního a automatizovaného testování, aby se určila jeho účinnost a přesnost při odhalování webových přihlašovacích rozhraní a identifikaci potenciálních bezpečnostních rizik.

Kromě vývoje softwarového nástroje si tato práce klade za cíl přispět do oblasti bezpečnosti webových aplikací tím, že určí potenciální oblasti pro budoucí výzkum a vývoj. To zahrnuje zkoumání využití pokročilých technik, jako je strojové učení a zpracování přirozeného jazyka, ke zvýšení přesnosti a účinnosti softwaru, a také zkoumání etických a právních důsledků používání nástroje v praxi. Práce se bude zabývat také výzvami spojenými s integrací softwarového nástroje se stávajícími nástroji a rámci pro zabezpečení webových aplikací a určí potenciální strategie pro překonání těchto výzev.

Konečným cílem této práce přispět do oblasti bezpečnosti webových aplikací vytvořením softwarového nástroje, který může pomoci odhalit a zmírnit potenciální bezpečnostní rizika spojená s webovými přihlašovacími rozhraními. Dosažením tohoto cíle práce přispěje k vývoji bezpečnějších a uživatelsky přívětivějších webových aplikací a pomůže chránit jednotlivce i podniky před rostoucí hrozbou kybernetických útoků.

1 Současný stav vědy a techniky

Tato kapitola obsahuje přehled literatury o zabezpečení webu a detekci přihlašovacích rozhraní a zabývá se existujícími nástroji a technikami pro detekci webových přihlašovacích rozhraní. Kapitola začíná přehledem zabezpečení webu a útoků na přihlašovací rozhraní, po němž následuje diskuse o důležitosti detekce přihlašovacích rozhraní a o výzvách, které jsou s tím spojeny. Kapitola poté obsahuje přehled stávajících přístupů k detekci webových přihlašovacích rozhraní a jejich výhod a omezení.

1.1 Penetrační testování

Penetrační testování je proces prováděný bezpečnostními profesionály s cílem identifikovat a využít zranitelnosti systému, sítě nebo aplikace. Cílem je simulovat reálný scénář útoku a posoudit odolnost zavedených bezpečnostních kontrol a opatření. Pomáhá určit proveditelnost útoku, potenciální dopad, který by mohl mít, a úroveň úsilí nebo zdrojů potřebných ke zneužití zranitelností [1, 14, 15].

V souvislosti s webovými aplikacemi nabývá penetrační testování jedinečného významu vzhledem ke složitosti a rozmanitosti těchto aplikací. S rostoucí závislostí na webových aplikacích pro obchodní operace se zajištění jejich bezpečnosti stalo prioritou. Penetrační testování webových aplikací pomáhá identifikovat zranitelnosti, které by mohli útočníci zneužít k získání neoprávněného přístupu, krádeži citlivých dat, narušení služeb nebo k jiným škodlivým činnostem[14, 15].

Penetrační testování webových aplikací se obvykle provádí pomocí kombinace automatizovaných nástrojů a manuálních technik. Proces začíná fází "průzkumu"nebo "sběru informací", kdy se shromažďují informace o cílové aplikaci. To může zahrnovat pochopení struktury aplikace, jejích technologií a potenciálních vstupních bodů pro útok.

Po průzkumu následuje fáze "skenování", která zahrnuje použití automatizovaných nástrojů k identifikaci potenciálních zranitelností. Ty by mohly zahrnovat místa pro *Structured query language* (SQL) injection, zranitelnosti *Cross-site scripting* (XSS) a potenciální oblasti pro *Cross-site request forgery* (CSRF)[1].

Po identifikaci potenciálních zranitelností začíná fáze "exploitace". V této fázi se penetrační tester pokusí zranitelnosti využít, aby pochopil jejich potenciální dopad. To může zahrnovat pokus o získání neoprávněného přístupu do systému, eskalaci oprávnění nebo získání citlivých dat[1].

Po úspěšném zneužití zranitelnosti může tester zůstat v systému neodhalen a simulovat to, co by mohl dělat skutečný útočník. To může zahrnovat vytvoření zadních vrátek nebo použití jiných technik k udržení přístupu do systému.

Poslední fáze je sepsání zprávy kde tester poskytne zadavateli všechny informace, které při testu získal. Cílem zprávy není pouhý výčet zranitelností zjištěných během testu, ale poskytnutí komplexní analýzy, která může organizaci nasměrovat k nápravě a zlepšit její celkovou bezpečnostní úroveň.

Každá z těchto fází probíhá v kontrolovaném prostředí a je pečlivě naplánována tak, aby nedošlo k neúmyslnému poškození nebo narušení provozu aplikace. Je velmi důležité, aby penetrační testy byly prováděny eticky a s patřičným oprávněním[1].

Penetrační testy webových aplikací poskytují cenné informace o stavu zabezpečení aplikace. Může odhalit jak technické zranitelnosti v kódu nebo konfiguraci aplikace, tak i nedostatky v bezpečnostních politikách nebo postupech. Díky simulaci reálného útoku poskytuje příležitost pochopit potenciální dopady útoku a otestovat postupy reakce na incident[14, 15].

Kromě toho může penetrační testování pomoci splnit požadavky na shodu s předpisy, protože mnoho předpisů a norem vyžaduje pravidelné testování bezpečnostních kontrol. Například směrnice NIS2 nařizuje pravidelné penetrační testování pro subjekty, které spadají do kategorie s vyššími povinnostmi.

Závěrem lze říci, že penetrační testování webových aplikací je důležitou součástí komplexní bezpečnostní strategie. Poskytuje realistické posouzení stavu zabezpečení aplikace a pomáhá identifikovat a odstranit zranitelnosti dříve, než je může útočník zneužít. Tím přispívá k ochraně citlivých údajů, kontinuitě obchodních operací a udržení dobrého jména organizace[14, 15].

1.2 Útoky na webové zabezpečení a přihlašovací rozhraní

Přihlašovací rozhraní jsou kritickými součástmi webových aplikací, protože umožňují uživatelům přístup k chráněným zdrojům a službám. Bohužel jsou také hlavním cílem útočníku, kteří využívají různé útočné techniky k získání neoprávněného přístupu k citlivým datům a systémům. V této kapitole budeme diskutovat o nejčastějších útocích na přihlašovací rozhraní, jejich mechanismech a možných důsledcích.

Útoky hrubou silou

Útoky hrubou silou spočívají v systematickém zkoušení různých kombinací uživatelských jmen a hesel, dokud nejsou objeveny správné přihlašovací údaje. Tento typ útoku může být časově náročný, zejména pokud má cílový systém silné zásady pro zadávání hesel. Útočníci však často používají automatizované nástroje a botnety, které tento proces urychlují a zvyšují pravděpodobnost úspěchu.

Pro zmírnění útoků hrubou silou mohou webové aplikace zavést opatření, jako je uzamčení účtu, prodlevy mezi pokusy o přihlášení, CAPTCHA a vícefaktorové ověřování (MFA). Kromě toho by si uživatelé měli vytvářet silná a jedinečná hesla, aby snížili pravděpodobnost úspěšného útoku hrubou silou[2].

Credential Stuffing

Útoky typu Credential stuffing využívají skutečnosti, že mnoho uživatelů opakovaně používá svá hesla na více platformách. Při tomto typu útoku útočníci používají dříve uniklé nebo ukradené přihlašovací údaje z jedné platformy k získání neoprávněného přístupu k jiným platformám, kde se používá stejná kombinace uživatelského jména a hesla. Útoky typu Credential stuffing jsou často automatizované, což útočníkům umožňuje rychle otestovat velké množství přihlašovacích údajů na různých přihlašovacích rozhraních.

Na obranu před útoky typu credential stuffing by si uživatelé měli vytvářet jedinečná hesla pro každý účet a používat správce hesel k jejich bezpečnému uložení. Webové aplikace mohou také implementovat vícefaktorové ověřování, kontrolní testy CAPTCHA a monitorovat pokusy o přihlášení kvůli podezřelým aktivitám[4].

Phishing

Phishingové útoky spočívají v tom, že na uživatele působí jako legitimní subjekt, např. banka, poskytovatel služeb nebo zaměstnavatel, a podvodně je nutí prozradit své přihlašovací údaje. To lze provést prostřednictvím e-mailu, sociálních médií nebo jiných komunikačních kanálů. Phishingové útoky často využívají techniky sociálního inženýrství, aby uživatele přiměly kliknout na škodlivé odkazy, stáhnout malware nebo zadat své přihlašovací údaje na falešné přihlašovací stránky.

Aby se uživatelé nestali obětí phishingových útoků, měli by být opatrní při klikání na odkazy, otevírání příloh nebo poskytování osobních údajů online[5].

Keylogging

Keylogging je metoda útoku, při níž útočníci pomocí škodlivého softwaru nebo hardwarových zařízení zachycují stisky kláves uživatelů, včetně jejich uživatelských jmen a hesel. Keyloggery mohou být do počítače oběti nainstalovány různými způsoby, například pomocí phishingu, malwaru nebo fyzického přístupu k zařízení.

Na ochranu před keyloggingem by uživatelé měli svá zařízení zabezpečit instalací renomovaného bezpečnostního softwaru, aktualizací operačních systémů a aplikací

a vyhýbáním se podezřelým souborům ke stažení. Webové aplikace mohou implementovat virtuální klávesnice nebo jednorázová hesla (OTP), aby se snížila účinnost útoků keyloggingu[6]).

Útoky *Man-in-the-middle* (MITM)

Při útoku man-in-the-middle zachytí útočníci komunikaci mezi uživatelem a webovou aplikací, což jim umožní odposlouchávat konverzaci, měnit přenášená data nebo vkládat škodlivý obsah. Útočníci tak mohou získat přihlašovací údaje, přebírat relace nebo manipulovat s akcemi uživatele.

Na obranu proti útokům MITM by webové aplikace měly používat šifrované komunikační kanály, například protokol *Hypertext Transfer Protocol Secure* (HTTPS) a šifrování *Secure Sockets Layer* (SSL). Uživatelé by také měli být obezřetní při připojování k veřejným sítím Wi-Fi a ověřovat pravost navštěvovaných webových stránek. Webové aplikace by také měly specifikovat hlavičku *HTTP Strict Transport Security* (HSTS)[7].

Password spraying

Password spraying je typ útoku, při kterém se útočníci pokoušejí získat neoprávněný přístup k více účtům vyzkoušením malého počtu běžně používaných hesel na velkém počtu uživatelských jmen. Tento přístup se liší od útoků hrubou silou, protože se zaměřuje spíše na zneužití slabých hesel než na vyzkoušení všech možných kombinací pro jeden účet.

Pro zmírnění útoků typu password spraying by webové aplikace měly prosazovat zásady silných hesel a vyžadovat, aby uživatelé vytvářeli složitá a jedinečná hesla. K zajištění další úrovně zabezpečení lze také zavést vícefaktorové ověřování. Uživatelé by se měli vyvarovat používání běžných nebo snadno uhodnutelných hesel, jako je "password", "123456"nebo "qwerty"[8].

SQL Injection

Útoky SQL injection využívají zranitelnosti ve webových aplikacích, které k ukládání uživatelských informací používají databáze SQL. Útočníci vkládají škodlivé dotazy SQL do přihlašovacích formulářů nebo jiných uživatelských vstupních polí, což může vést k neoprávněnému přístupu, krádeži dat nebo dokonce k úplnému ovládnutí napadeného systému.

Na obranu proti útokům SQL injection by měli vývojáři webových aplikací používat postupy bezpečného kódování, například parametrizované dotazy a ověřování

vstupů. Ve webových aplikacích lze také implementovat webové aplikační firewally (WAF), které odhalí a zablokují podezřelé požadavky[9].

Session hijacking

Session hijacking neboli únos relace zahrnuje krádež souboru cookie relace uživatele, které webové aplikace používají k udržování stavu přihlášeného uživatele. Získáním platného souboru cookie relace se může útočník vydávat za uživatele a získat neoprávněný přístup k jeho účtu, aniž by potřeboval jeho přihlašovací údaje.

Na ochranu proti únosu relace by webové aplikace měly implementovat techniky bezpečné správy relace, jako je používání zabezpečených souborů cookie, regenerace identifikátorů relace při přihlášení a implementace krátkých časových limitů relace. Uživatelé mohou riziko únosu relace snížit také tím, že se budou z webových aplikací odhlašovat, když je nepoužívají, a vyhýbat se veřejným nebo nezabezpečeným sítím Wi-Fi[10].

Sociální inženýrství

Útoky sociálního inženýrství využívají lidské psychologie k manipulaci uživatelů, aby prozradili své přihlašovací údaje nebo jiné citlivé informace. Toho lze dosáhnout různými taktikami, například vydáváním se za důvěryhodnou osobu, vyvoláním pocitu naléhavosti nebo využitím zvědavosti, strachu nebo touhy uživatele pomoci ostatním.

Aby se uživatelé mohli bránit útokům sociálního inženýrství, měli by být poučeni o rizicích a vyškoleni v rozpoznávání běžných taktik používaných útočníky. Webové aplikace mohou také implementovat další bezpečnostní opatření, jako je vícefaktorové ověřování, aby se snížil dopad kompromitovaných přihlašovacích údajů[11].

Závěr

Webová přihlašovací rozhraní jsou hlavním cílem útočníků a pochopení nejčastějších útoků proti nim je pro uživatele i vývojáře zásadní. Zavedením silných bezpečnostních opatření, jako jsou zásady silných hesel, vícefaktorové ověřování a postupy bezpečného kódování, mohou webové aplikace snížit riziko neoprávněného přístupu a ochránit citlivé údaje. Uživatelé by navíc měli být ostražití při zadávání svých přihlašovacích údajů online a používat osvědčené bezpečnostní postupy, aby snížili pravděpodobnost, že se stanou obětí těchto útoků.

1.3 Stávající přístupy k detekci webových přihlašovacích rozhraní

Detekce webového přihlašovacího rozhraní je důležitým procesem v různých oblastech, včetně kybernetické bezpečnosti a správy řízení přístupu. V průběhu let navrhli výzkumníci i odborníci z praxe řadu technik pro detekci přihlašovacích rozhraní. V této části je uveden podrobný přehled tří hlavních kategorií metod detekce webových přihlašovacích rozhraní: statická analýza, dynamická analýza a přístupy založené na strojovém učení.

Statická analýza

Statická analýza je technika, která analyzuje zdrojový kód webových stránek, aniž by došlo ke spuštění webové aplikace nebo interakci s ní. Tato metoda obvykle zahrnuje hledání specifických prvků *HyperText Markup Language* (HTML), atributů nebo vzorů ve zdrojovém kódu, které svědčí o přihlašovacím rozhraní. Mezi klíčové součásti statické analýzy patří:

- Rozbor jazyka HTML: Analýza HTML je proces zkoumání struktury dokumentu HTML a extrakce relevantních prvků a atributů. V kontextu detekce přihlašovacího rozhraní lze parsery použít k identifikaci formulářů, vstupních polí a dalších podstatných prvků, které mohou indikovat přítomnost přihlašovacího rozhraní.
- Regulární výrazy: Regulární výrazy jsou vzory používané k porovnávání konkrétních řetězců nebo sekvencí znaků v daném textu. Lze je využít při statické analýze k vyhledávání konkrétních vzorů ve zdrojovém kódu, například konkrétních klíčových slov nebo atributů spojených s přihlašovacími rozhraními (např. "username", "password", "login"atd.)[13].
- Heuristika: Přístupy založené na heuristice se při identifikaci přihlašovacích rozhraní spoléhají na soubor předem definovaných pravidel nebo vzorů. Tato pravidla mohou být založena na přítomnosti nebo nepřítomnosti konkrétních prvků HTML, atributů nebo textových vzorů. Heuristiky mohou být jednoduché nebo složité, v závislosti na požadované úrovni přesnosti a specifičnosti.

I přes svou jednoduchost a rychlost má statická analýza několik omezení. Za prvé, nemusí přesně odhalit přihlašovací rozhraní vytvořená nebo upravená pomocí skriptovacích jazyků na straně klienta, jako je JavaScript. Za druhé, statická analýza může poskytovat falešně pozitivní nebo negativní výsledky kvůli nedostatečnému porozumění kontextu. A konečně se může obtížně přizpůsobovat různým a vyvíjejícím se webovým technologiím, protože detekční pravidla mohou vyžadovat neustálou aktualizaci, aby zůstala účinná[12].

Dynamická analýza

Dynamická analýza zahrnuje interakci s webovou aplikací, a to buď ručně, nebo pomocí automatizovaných nástrojů, za účelem identifikace přihlašovacích rozhraní. Tento přístup zohledňuje chování a vzhled webových stránek za běhu, což může pomoci překonat některá omezení statické analýzy. Mezi klíčové aspekty dynamické analýzy patří:

- Webové crawlery a scrapery: Webové crawlery a scrapery jsou automatizované nástroje, které mohou procházet webové aplikace a interagovat s nimi, simulovat interakce uživatelů a shromažďovat relevantní data. V kontextu detekce přihlašovacích rozhraní mohou tyto nástroje navštěvovat různé stránky, sledovat odkazy, odesílat formuláře a pozorovat výsledné chování za účelem identifikace přihlašovacích rozhraní.
- Automatizace prohlížeče: Rámce pro automatizaci prohlížeče, jako je například Selenium, umožňují automatizované ovládání webových prohlížečů, což umožňuje replikovat interakce uživatelů s webovými aplikacemi. Díky automatizaci akcí prohlížeče může dynamická analýza zachytit chování za běhu a vizuální aspekty webových stránek, což může být užitečné pro detekci přihlašovacích rozhraní.
- Spouštění JavaScriptu: Vzhledem k tomu, že se mnoho webových aplikací při vykreslování a úpravách obsahu spoléhá na JavaScript na straně klienta, může spuštění JavaScriptu během dynamické analýzy pomoci odhalit přihlašovací rozhraní, která mohou být skrytá nebo generovaná za běhu. Tento přístup může zvýšit přesnost detekce ve srovnání se statickou analýzou, která nemusí zohlednit skriptování na straně klienta.

Dynamická analýza může nabídnout přesnější výsledky a zohlednit chování na straně klienta, má však také svá omezení. Zaprvé může být ve srovnání se statickou analýzou časově i zdrojově náročnější, protože vyžaduje interakci s webovou aplikací. Zadruhé, stále se může potýkat s problémy při odhalování obfuskovaných nebo složitých webových aplikací, zejména pokud je přihlašovací rozhraní skryto za více vrstvami interakce s uživatelem[12, 16].

Přístupy založené na umělé inteligenci

Přístupy založené na strojovém učení využívají algoritmy strojového učení k detekci přihlašovacích rozhraní na základě učení vzorů z označené sady dat webových stránek. Tyto techniky mohou být ve srovnání s metodami statické a dynamické analýzy přesnější a odolnější vůči obfuskaci, protože mohou na základě naučených vzorů zobecňovat a identifikovat přihlašovací rozhraní i ve složitých nebo dříve neviděných

webových aplikacích. Mezi klíčové součásti přístupů založených na strojovém učení patří např:

Extrakce příznaků: Extrakce příznaků zahrnuje identifikaci a extrakci relevantních příznaků z webových stránek, které lze použít jako vstup pro algoritmy strojového učení. V kontextu detekce přihlašovacího rozhraní mohou rysy zahrnovat prvky HTML, atributy, textové vzory, vizuální aspekty nebo dokonce charakteristiky chování.

Trénování a ověřování: Po extrakci rysů se k trénování modelu strojového učení použije označený soubor dat webových stránek s přihlašovacími rozhraními a bez nich. Model je poté ověřen na samostatném souboru dat, aby se vyhodnotila jeho výkonnost a zajistilo se, že se dokáže dobře zobecnit na nová data.

Klasifikační algoritmy: Pro klasifikační úlohy lze použít různé algoritmy strojového učení, například metody podpůrných vektorů (SVM), rozhodovací stromy, náhodné lesy nebo neuronové sítě. Výběr algoritmu závisí na konkrétním problému, souboru dat a požadované úrovni přesnosti a výpočetní účinnosti.

Metriky hodnocení: K posouzení výkonnosti přístupu založeného na strojovém učení lze použít různé hodnotící metriky, včetně přesnosti, přesnosti, odvolání a skóre F1. Tyto metriky poskytují přehled o účinnosti metody a mohou pomoci určit oblasti, které je třeba zlepšit nebo optimalizovat.

Přístupy založené na strojovém učení jsou sice slibné, ale mají také svá omezení. Za prvé vyžadují značné množství trénovacích dat, jejichž získání může být náročné, zejména pro různorodé webové aplikace. Za druhé mohou být výpočetně náročné, zejména při použití složitých modelů nebo velkých souborů dat. A také může být výkonnost modelu ovlivněna kvalitou trénovacích dat, protože zašuměná nebo zkreslená data mohou vést ke špatné generalizaci.

Závěrem lze říci, že techniky detekce webového přihlašovacího rozhraní se v průběhu let vyvíjely, přičemž každá metoda nabízí svůj jedinečný soubor výhod a omezení. Statická analýza, dynamická analýza a přístupy založené na strojovém učení přispívají do této oblasti cennými poznatky, ale zároveň se potýkají s problémy, pokud jde o přesnost, účinnost a přizpůsobivost různým a vyvíjejícím se webovým technologiím. Vývoj komplexního, efektivního a přesného řešení pro detekci webových přihlašovacích rozhraní zůstává stále aktuální výzkumnou výzvou a skript v jazyce Python navržený v této práci si klade za cíl odstranit některá z těchto omezení a poskytnout spolehlivý a efektivní nástroj pro detekci webových přihlašovacích rozhraní[12].

1.4 Výhody a nedostatky stávajících přístupů

Stávající přístupy k detekci webových přihlašovacích rozhraní mají různé výhody a omezení. Heuristické přístupy jsou jednoduché a rychlé, ale mohou mít vysokou míru falešně pozitivních výsledků a nemusí dobře fungovat pro složitá přihlašovací rozhraní. Přístupy založené na strojovém učení se mohou naučit složité vzory a vlastnosti přihlašovacích rozhraní, ale vyžadují trénovací data a mohou být výpočetně nákladné.

K řešení těchto omezení bylo navrženo několik hybridních přístupů, které kombinují silné stránky přístupů založených na heuristice a strojovém učení. Cílem těchto hybridních přístupů je zvýšit přesnost a účinnost detekce přihlašovacích rozhraní a zároveň minimalizovat falešně pozitivní výsledky.

1.5 Současné nástroje pro detekci webových přihlašovacích rozhraní

Pro odhalování a analýzu jednotlivých komponent webových aplikací, včetně přihlašovacích rozhraní, byly vyvinuty různé skenery a nástroje. Tato kapitola se zaměřuje na přehled některých nejoblíbenějších a nejpoužívanějších skenerů, jako jsou Burp Suite, *Open Web Application Security Project* (OWASP) *Zed Attack Proxy* (ZAP) a Nmap, porovnává jejich funkce a hodnotí jejich vhodnost pro odhalování přihlašovacích rozhraní v různých typech webových aplikací. Cílem tohoto přehledu je poskytnout ucelený náhled do současného stavu technologie detekce přihlašovacích rozhraní webových aplikací a identifikovat potenciální oblasti pro zlepšení a inovace.

Burp Suite

Burp Suite je komplexní platforma pro testování zabezpečení webových aplikací vyvinutá společností PortSwigger. Poskytuje širokou škálu nástrojů pro různé úlohy testování zabezpečení, včetně skenování, mapování a analýzy webových aplikací. Jednou z jejích funkcí je Burp Spider, který je určen k procházení a mapování webových aplikací a při tom objevuje komponenty, jako jsou například přihlašovací rozhraní[18].

Mezi klíčové funkce sady Burp Suite, které se týkají detekce přihlašovacích rozhraní, patří:

- Spider: Nástroj, který automaticky prochází webové aplikace a objevuje odkazy, formuláře a další součásti, včetně přihlašovacích rozhraní.

- Target: Funkce, která umožňuje uživatelům zobrazit a spravovat mapu webu vytvořenou nástrojem Spider a poskytuje přehled objevených komponent a jejich vztahů.
- Proxy: Proxy server, který zachycuje všechny požadavky a odpovědi *Hypertext Transfer Protocol* (HTTP) mezi prohlížečem a cílovou aplikací, což může pomoci při ruční identifikaci přihlašovacích rozhraní.

OWASP ZAP

OWASP Zed Attack Proxy je open-source nástroj pro testování zabezpečení webových aplikací vyvinutý organizací Open Web Application Security Project (OWASP). ZAP je navržen tak, aby pomáhal testerům odhalovat a analyzovat součásti webových aplikací ve fázi vývoje a testování. Má snadno použitelné rozhraní a poskytuje širokou škálu funkcí, jako je pasivní skenování, aktivní skenování a spidering[19].

Mezi klíčové funkce nástroje OWASP ZAP důležité pro detekci přihlašovacího rozhraní patří:

- Spider: Nástroj, který automaticky prochází webové aplikace a zjišťuje odkazy, formuláře a další součásti, včetně přihlašovacích rozhraní.
- *Asynchronous JavaScript and XML* (AJAX) Spider: Slouží k vyhledávání webových stránek, které se nacházejí na webu: Funkce, která umožňuje ZAP procházet a objevovat komponenty ve webových aplikacích, které se při vykreslování a navigaci spoléhají na JavaScript, což může být užitečné zejména při detekci přihlašovacích rozhraní v moderních webových aplikacích.
- Site tree: Hierarchické zobrazení objevených komponent webové aplikace, které poskytuje přehled o struktuře aplikace a může pomoci při identifikaci přihlašovacích rozhraní.

Nmap

Nmap (Network Mapper) je síťový bezpečnostní skener s otevřeným zdrojovým kódem, který slouží k odhalování počítačů a služeb v počítačové síti. Ačkoli se jedná především o nástroj pro skenování sítě, Nmap obsahuje také různé skripty pro zjišťování a analýzu webových aplikací prostřednictvím nástroje Nmap Scripting Engine (NSE)[20].

Mezi klíčové funkce nástroje Nmap důležité pro detekci přihlašovacích rozhraní patří:

- Nmap Scripting Engine (NSE): NSE: výkonný skriptovací jazyk, který uživatelům umožňuje psát vlastní skripty pro specializované úlohy, včetně zjišťování a analýzy komponent webových aplikací. odpovědí.

- http-auth-finder Spideruje stránku od specifikovaného počátečního bodu a vyhledává přihlašovací formuláře a stránky vyzívající k HTTP-autentizaci.

Srovnání a diskuse

Sady Burp Suite, OWASP ZAP a Nmap poskytují funkce a možnosti, které lze využít pro detekci přihlašovacích rozhraní ve webových aplikacích. Jejich zaměření a silné stránky se však liší, takže každý nástroj je vhodnější pro konkrétní případy použití.

Sada Burp Suite nabízí ucelenou sadu nástrojů pro testování zabezpečení webových aplikací a její funkce Spider umožňuje efektivní odhalování přihlašovacích rozhraní. Integrace funkcí Proxy a Target z něj činí výkonné řešení pro automatizované i manuální odhalování přihlašovacích rozhraní.

OWASP ZAP je open-source nástroj pro testování zabezpečení webových aplikací se silným důrazem na použitelnost a podporu komunity. Jeho funkce Spider a AJAX Spider z něj činí dobrou volbu pro detekci přihlašovacích rozhraní v široké škále webových aplikací, včetně těch, které se při vykreslování a navigaci spoléhají na JavaScript. Strom webů navíc poskytuje jasný přehled o struktuře aplikace, což může pomoci při identifikaci přihlašovacích rozhraní.

Nmap je především skener zabezpečení sítě, ale jeho nástroj Nmap Scripting Engine (NSE) umožňuje uživatelům provádět odhalování a analýzu komponent webových aplikací. Nenabízí sice stejnou úroveň funkčnosti a granularity jako sady Burp Suite nebo OWASP ZAP pro detekci přihlašovacích rozhraní, ale Nmap je výkonný a flexibilní nástroj, který lze použít ve spojení s dalšími nástroji pro komplexnější posouzení komponent webových aplikací.

Závěr

Souhrnně lze říci, že stávající řešení pro detekci přihlašovacích rozhraní ve webových aplikacích, jako jsou Burp Suite, OWASP ZAP a Nmap, nabízejí řadu funkcí a možností, které lze využít pro identifikaci a analýzu těchto kritických komponent. Každý nástroj má své silné a slabé stránky a výběr vhodného nástroje závisí na konkrétních požadavcích a omezeních projektu.

Na základě přezkoumání a porovnání těchto existujících řešení můžeme identifikovat potenciální oblasti pro zlepšení a inovace při vývoji nástroje pro detekci webového přihlašovacího rozhraní v jazyce Python pro tuto magisterskou práci. Například zaměření na snadnost použití a přizpůsobitelnost by mohlo pomoci vytvořit nástroj, který bude přístupný širšímu okruhu uživatelů, včetně vývojářů, bezpečnostních výzkumníků a penetračních testerů. Integrace technik strojového učení a umělé

inteligence by navíc mohla zvýšit schopnost nástroje přesně detekovat a klasifikovat přihlašovací rozhraní, a to i ve složitých nebo obfuskovaných webových aplikacích.

1.6 Knihovny a nástroje pro extrakci dat z webů v jazyce Python

Python je široce používaný programovací jazyk, který nabízí rozsáhlý ekosystém nástrojů a knihoven pro scraping webových stránek, získávání dat a analýzu. Tyto nástroje a knihovny hrají zásadní roli při usnadňování vývoje řešení pro detekci webových přihlašovacích rozhraní. V této části je uveden ucelený přehled populárních nástrojů a knihoven založených na jazyce Python, které se běžně používají v úlohách scrapování a analýzy webu, se zaměřením na jejich klíčové funkce, možnosti a případy použití.

BeautifulSoup

BeautifulSoup je široce používaná knihovna jazyka Python pro parsování dokumentů HTML a *Extensible Markup Language* (XML). Poskytuje jednoduchý a efektivní způsob navigace, vyhledávání a úprav parsovacího stromu. BeautifulSoup si poradí se špatně formulovaným HTML a umožňuje uživatelům snadný přístup k prvkům a atributům HTML a manipulaci s nimi. Mezi klíčové funkce a případy použití BeautifulSoup patří[21]:

- Parsování HTML a XML: BeautifulSoup dokáže analyzovat dokumenty HTML i XML, což uživatelům umožňuje pracovat s různými typy webového obsahu.
- Procházení stromů: Knihovna poskytuje metody procházení a prohledávání stromu parsování, což usnadňuje vyhledávání konkrétních prvků nebo skupin prvků.
- Extrakce dat: BeautifulSoup umožňuje uživatelům extrahovat data z prvků a atributů HTML, což může být užitečné při úlohách web scrapingu a data miningu.
- Úprava a formátování: Knihovna také podporuje úpravu stromu parsování a přeformátování výstupu, což uživatelům umožňuje manipulovat s webovým obsahem podle potřeby.

Selenium

Selenium je výkonný rámec pro automatizaci prohlížeče, který uživatelům umožňuje programově ovládat webové prohlížeče. Podporuje různé webové prohlížeče, včetně prohlížečů Chrome, Firefox a Safari, a poskytuje rozhraní *Application Programming*

Interface (API) pro interakci s webovými stránkami, simulaci akcí uživatele a zachycení chování za běhu. Selenium je obzvláště užitečné pro úlohy detekce webového přihlašovacího rozhraní, které vyžadují dynamickou analýzu, protože dokáže spouštět JavaScript a komunikovat s webovými aplikacemi v reálném čase. Mezi klíčové funkce a případy použití nástroje Selenium patří[17]:

- Automatizace prohlížeče: Selenium umožňuje uživatelům automatizovat akce prohlížeče, jako je otevírání webových stránek, klikání na tlačítka, vyplňování formulářů a přecházení mezi stránkami.
- Spouštění JavaScriptu: Framework dokáže spouštět kód JavaScriptu, což umožňuje zachytit chování za běhu a komunikovat s dynamickými webovými aplikacemi.
- Interakce s webovými prvky: Selenium poskytuje rozhraní API pro vyhledávání a interakci s webovými prvky, což uživatelům umožňuje simulovat uživatelské akce a získávat informace z webových stránek.
- Zachycení snímků obrazovky: Knihovna podporuje pořizování snímků obrazovky webových stránek, což může být užitečné pro úlohy vizuální analýzy, například pro identifikaci přihlašovacích rozhraní na základě jejich vzhledu.

Scrapy

Scrapy je open-source framework pro procházení webových stránek a scraping pro jazyk Python, který poskytuje výkonný a flexibilní způsob získávání dat z webových stránek. Je určen pro rozsáhlé úlohy scrapování webových stránek a snadno si poradí se složitými webovými aplikacemi. Scrapy nabízí různé vestavěné funkce pro zpracování požadavků, zpracování dat a správu úložišť, díky čemuž je oblíbenou volbou pro projekty scrapingu webů a získávání dat. Mezi klíčové funkce a případy použití Scrapy patří[22]:

- Procházení webových stránek: Scrapy dokáže procházet více webových stránek současně, sledovat odkazy a efektivně extrahovat data, takže je vhodný pro rozsáhlé úlohy web scrapingu.
- Extrakce dat: Framework poskytuje vestavěnou podporu pro extrakci dat pomocí selektorů *Cascading Style Sheets* (CSS) a výrazů XPath, což uživatelům umožňuje přesné vyhledávání a extrakci informací z webových stránek.
- Zpracování požadavků a odpovědí: Scrapy nabízí robustní možnosti zpracování požadavků a odpovědí, včetně podpory zpracování přesměrování, souborů cookie a relací, což může být užitečné v úlohách detekce webového přihlašovacího rozhraní, které zahrnují interakci s webovými aplikacemi.
- Ukládání a výstup dat: Knihovna podporuje různé formáty ukládání a výstupu dat, například *JavaScript Object Notation* (JSON), *Comma Separated Values*

(CSV) a XML, což uživatelům umožňuje ukládat a spravovat získaná data podle potřeby.

Requests

Requests je oblíbená knihovna jazyka Python pro vytváření požadavků HTTP, která je nezbytnou součástí úloh scrapování webu a extrakce dat. Poskytuje jednoduché a uživatelsky přívětivé rozhraní API pro odesílání a přijímání požadavků HTTP a obsluhu různých aspektů webové komunikace, jako je ověřování, soubory cookie a přesměrování. Požadavky jsou užitečné zejména pro úlohy detekce webového přihlašovacího rozhraní, které zahrnují načítání webových stránek a extrakci jejich zdrojového kódu pro další analýzu. Mezi klíčové funkce a případy použití aplikace Requests patří[23]:

- HTTP požadavky: Requests podporuje různé metody HTTP, včetně GET, POST, PUT a DELETE, a umožňuje tak uživatelům efektivně komunikovat s webovými aplikacemi a rozhraními API.
- Ověřování: Knihovna poskytuje vestavěnou podporu pro různé mechanismy ověřování, například Basic, Digest a OAuth, což může být užitečné v úlohách detekce webového přihlašovacího rozhraní, které vyžadují přístup k ověřeným zdrojům.
- Zpracování relací: Knihovna podporuje zpracování relací, což uživatelům umožňuje udržovat stav napříč více požadavky, spravovat soubory cookie a bezproblémově zpracovávat přesměrování.
- Zpracování odpovědí: Knihovna nabízí praktické metody pro zpracování odpovědí HTTP, včetně podpory zpracování obsahu JSON, XML a HTML.

Závěr

Využitím možností těchto nástrojů a knihoven založených na jazyku Python mohou vývojáři vytvářet robustní a efektivní řešení pro úlohy detekce webového přihlašovacího rozhraní. Technologie BeautifulSoup, Selenium, Scrapy a Requests nabízejí jedinečné funkce a výhody, které lze využít v různých aspektech procesu detekce, například při scrapování webu, extrakci dat, dynamické analýze a interakci s webovými aplikacemi.

V kontextu detekce webového přihlašovacího rozhraní mohou vývojáři spojit silné stránky těchto nástrojů a knihoven a vytvořit komplexní řešení, které řeší omezení stávajících metod. Například BeautifulSoup lze použít k analýze jazyka HTML a extrakci relevantních prvků a atributů, zatímco Selenium si poradí s dynamickými aspekty webových aplikací, jako je spouštění JavaScriptu a automatizace prohlížeče.

Scrapy lze použít pro rozsáhlé úlohy scrapování webu a Requests může usnadnit efektivní komunikaci s webovými aplikacemi a rozhraními API.

Výběr nástrojů a knihoven bude záviset na konkrétních požadavcích úlohy detekce webového přihlašovacího rozhraní a také na požadované úrovni přesnosti, efektivity a škálovatelnosti. Pochopením schopností jednotlivých nástrojů a knihoven mohou vývojáři činit informovaná rozhodnutí a vytvořit účinné řešení, které řeší problémy detekce webového přihlašovacího rozhraní a přispívá k probíhajícímu výzkumu v této oblasti.

2 Praktická část

Detekce přihlašovacích rozhraní na webových stránkách je klíčovým úkolem v mnoha aplikacích kybernetické bezpečnosti. Detekce webových přihlašovacích rozhraní může pomoci penetračním testerům a výzkumníkům v oblasti bezpečnosti při identifikaci potenciálních vstupních bodů pro různé útoky. V této kapitole bude diskutován návrh a implementace nástroje pro detekci webových přihlašovacích rozhraní v jazyce Python, který efektivně skenuje webové stránky a detekuje přihlašovací rozhraní pomocí heuristiky a regexů. Skript dále využívá threadingu pro zvýšení výkonu.

Pro instalaci potřebných knihoven použijte následující příkaz: `pip install requests bs4 click` Tento příkaz nainstaluje knihovny Requests, BeautifulSoup a Click, které jsou nezbytné pro fungování nástroje. Po instalaci knihoven můžete nástroj spustit pomocí dodaného skriptu Python.

2.1 Podrobnosti o implementaci

Implementaci nástroje lze rozdělit do několika částí, z nichž každá má specifický účel:

Rozhraní příkazového řádku

Knihovna Click slouží k definování a analýze argumentů a voleb příkazového řádku. Uživatelé mohou zadat vstupní soubory obsahující seznamy webových stránek a seznamy slov, určit počet vláken, nastavit výstupní soubor pro ukládání výsledků, povolit funkci spideringu a konfigurovat další možnosti prostřednictvím argumentů příkazového řádku.

```
@click.command()
@click.option("--websites", type=click.File("r"),\
help="File␣containing␣list␣of␣websites␣to␣scan.")
@click.option("--single-url", type=str,\
help="A␣single␣URL␣to␣scan.")
@click.option("--wordlist", type=click.File("r"),\
help="File␣containing␣list␣of␣words␣to␣check.")
@click.option("--threads", type=int, default=100,\
help="Number␣of␣concurrent␣threads.")
@click.option("--output", type=str,\
help="Output␣file␣to␣store␣the␣results␣as␣JSON.")
@click.option("--max-depth", type=int, default=2,\
help="Maximum␣depth␣to␣follow␣links␣during␣spidering.")
```

```
@click.option("--timeout", type=float, default=2,\
help="Timeout␣in␣seconds␣for␣requests.")
@click.option("--spider", is_flag=True,\
help="Enable␣spidering␣functionality.")
```

2.2 Proces skenování

Proces skenování zahrnuje načítání webových stránek pomocí knihovny Requests, jejich analýzu pomocí BeautifulSoup a vyhledávání potenciálních přihlašovacích formulářů pomocí heuristiky a regulárních výrazů. Pokud je povoleno pavoukování, nástroj také sleduje odkazy na načtených webových stránkách až do zadané hloubky, aby našel další stránky ke skenování.

```
def check_url(url, spider, max_depth, timeout=1):

    if spider:
        visited_urls = spider_website\
        (url, max_depth,timeout)
    else:
        visited_urls = [url]

    for visited_url in visited_urls:
        try:
            response = requests.get\
            (visited_url, timeout=timeout)
            if response.status_code != 200:
                continue
            soup = BeautifulSoup\
            (response.text, "html.parser")
            score = detect_login_form(soup)
            if score > 1:
                return visited_url, score

        except KeyboardInterrupt as Ki:
            sys.exit(0)
        except Exception as e:
            print(e.message)

    return url, 0
```

2.3 Paralelizace

Vícevláknové zpracování je technika, která umožňuje programu provádět více úloh současně, což může vést k výraznému zvýšení výkonu a efektivity, zejména v případech, kdy lze úlohy provádět nezávisle. V kontextu našeho nástroje pro detekci přihlašovacích rozhraní lze multithreading využít k paralelizaci procesu skenování více webových stránek, čímž se zkrátí celkový čas potřebný k dokončení úlohy.

V této části se budeme zabývat implementací multithreadingu v nástroji pro detekci přihlašovacích rozhraní a optimalizací výkonu dosaženou jeho použitím.

Knihovna Threading

K implementaci vícevláknového zpracování v tomto nástroji se využívá vestavěná knihovna pro zpracování vláken v jazyce Python. Tato knihovna poskytuje jednoduché rozhraní pro vytváření a správu vláken a také synchronizační mechanismy, jako jsou zámky a semafory, které zajišťují konzistenci dat ve více vláknech.

Vícevláknové skenování webových stránek

V nástroji pro detekci přihlašovacích rozhraní implementujeme vícevláknovost vytvořením samostatného vlákna pro každou webovou stránku v seznamu cílových webových stránek. Každé vlákno je zodpovědné za skenování jedné webové stránky, včetně detekce přítomnosti přihlašovacího rozhraní, prohledávání webu pavoukem pro další adresy *Uniform Resource Locator* (URL) (pokud je povoleno) a ukládání výsledků. To umožňuje skenovat více webových stránek současně, čímž se výrazně zkracuje celkový čas potřebný k dokončení úlohy.

Aby byla zajištěna konzistence výstupu a nedocházelo k "race conditions"nebo poškození dat, používáme pro ukládání výsledků datovou strukturu bezpečnou pro vlákna. V našem případě používáme vestavěnou třídu Queue.Queue jazyka Python, která poskytuje jednoduché rozhraní pro vkládání a načítání položek způsobem bezpečným pro vlákna.

Techniky optimalizace výkonu

Kromě implementace vícevláknového zpracování používáme v nástroji pro detekci přihlašovacích rozhraní několik technik optimalizace výkonu, jako např:

- Časové limity: Zahrnujeme parametr timeout pro volání requests.get(), abychom zabránili nekonečnému visení skriptu na pomalých nebo neodpovídajících webových stránkách. Tím je zajištěno, že skript může pokračovat ve skenování dalších webů, i když jeden web neodpoví v rozumném čase.

- Sdružování připojení: Knihovna requests, která se používá k provádění požadavků HTTP, ve výchozím nastavení podporuje sdružování připojení. To umožňuje nástroji opakovaně používat stávající spojení se serverem, čímž se snižuje režie spojená s navazováním nových spojení a zlepšuje celkový výkon.
- Selektivní spidering: Nástroj obsahuje volitelnou funkci spideringu, kterou lze povolit nebo zakázat podle preferencí uživatele. Vynecháním spideringu, pokud to není nutné, můžeme snížit počet provedených požadavků HTTP a zlepšit celkový výkon nástroje.

```
with ThreadPoolExecutor(max_workers=threads) as executor:
        future_to_url = {executor.submit(check_url, url, \
        spider,\
        max_depth, timeout): url for url in urls_to_check}

        with click.progressbar(length=len(urls_to_check),
        label="Scanning websites") as bar:
            for future in as_completed(future_to_url):
                url = future_to_url[future]
                try:
                    visited_url, score = future.result()
                    if score > 0:
                        results[visited_url] = score
                except KeyboardInterrupt as Ki:
                    sys.exit(0)
                except Exception as e:
                    pass
                bar.update(1)
```

2.4 Detekce přihlašovacích formulářů

Komponenta pro detekci přihlašovacích formulářů používá heuristiku a regulární výrazy k identifikaci potenciálních přihlašovacích formulářů na analyzovaných webových stránkách. Tato heuristika zahrnuje vyhledávání specifických atributů (např. "id", "class", "name"), které mohou indikovat přítomnost přihlašovacího formuláře. Kromě toho se k vyhledávání vzorů běžně se vyskytujících v přihlašovacích formulářích, jako jsou vstupní pole pro uživatelská jména a hesla, používá sada předdefinovaných regulárních výrazů.

```
def detect_login_form(html):
    score = 0
```

```
    soup = html
    forms = soup.find_all("form")

    form_keywords = ["login", "signin", "log-in",\
    "sign-in", "auth", "authentication", "session"]
    input_keywords = ["username", "user", "email",\
    "password", "pass", "pwd"]

    for form in forms:
        for keyword in form_keywords:
            if keyword in form.get("id", "").lower()\
            or keyword in "␣".join(form.get("class", [])).\
            lower() or keyword in form.get("name", "").lower():
                score += 1

        form_str = str(form).lower()
        for keyword in input_keywords:
            if keyword in form_str:
                score += 1

        email_pattern = re.compile(r'<input[^>]*type=["\']?
        email["\']?[^>]*>',␣re.IGNORECASE)
␣␣␣␣␣␣␣␣password_pattern␣=␣re.compile(r'<input[^>]*type=["\']?
␣␣␣␣␣␣␣␣password["\']?[^>]*>', re.IGNORECASE)

        form_str = str(form)
        if email_pattern.search(form_str):
            score += 2
        if password_pattern.search(form_str):
            score += 20

    return score
```

2.5 Podpora proxy serverů a anonymita

Anonymita je zásadním aspektem, který je třeba vzít v úvahu při vývoji nástroje pro scraping nebo crawling webu, zejména pokud se jedná o citlivé informace nebo se snaží obejít omezení stanovená webovými stránkami. Pomocí proxy serverů může

nástroj maskovat IP adresu uživatele a zachovat určitou úroveň anonymity při přístupu na webové stránky. Tím se snižuje pravděpodobnost, že budou cílové webové stránky zablokovány nebo že budou čelit právním důsledkům kvůli možnému porušení soukromí.

V této části je popsána implementace podpory proxy serverů v nástroji pro detekci přihlašovacích rozhraní a jeho úloha při zajišťování anonymity.

Implementace podpory proxy serverů

K implementaci podpory proxy v nástroji pro detekci přihlašovacích rozhraní je využito vestavěných funkcí proxy v knihovně requests. Knihovna requests umožňuje předat slovník proxy adres URL pro různé protokoly (např. HTTP a HTTPS) jako parametr ve funkci get(). To umožňuje nástroji směrovat své požadavky přes zadaný proxy server, čímž se účinně maskuje IP adresa uživatele.

V nástroji je přidána nová volba příkazového řádku –proxy, která přijímá adresu URL proxy serveru jako vstup od uživatele. Adresa URL proxy serveru by měla mít formát http://user:password@proxy.example.com:8080, kde uživatel, heslo, proxy.example.com a 8080 by měly být nahrazeny skutečnými přihlašovacími údaji a adresou proxy serveru. Pokud není zadána volba –proxy, bude nástroj pracovat bez použití proxy serveru.

Použití proxy serverů pro zajištění anonymity

Začleněním podpory proxy serverů do nástroje umožňujeme uživatelům zachovat určitou úroveň anonymity při skenování webových stránek pro přihlašovací rozhraní. To může pomoci chránit identitu uživatele a snížit riziko zablokování cílových webových stránek kvůli omezením na základě IP nebo mechanismům omezujícím rychlost.

Je nezbytné poznamenat, že ne všechny proxy servery poskytují stejnou úroveň anonymity a výběr proxy serveru může významně ovlivnit účinnost nástroje při zachování anonymity. Uživatelé by měli pečlivě vybírat renomovaný a spolehlivý proxy server, aby si zajistili co nejlepší ochranu.

Omezení a připomínky

Ačkoli podpora proxy serverů nabízí cennou vrstvu anonymity, není bez omezení a připomínek. Mezi potenciální problémy, kterých by si uživatelé měli být vědomi, patří např:

- Dopad na výkon: Směrování požadavků přes proxy server může přinést dodatečné zpoždění, které může ovlivnit výkon nástroje. Uživatelé by měli při volbě použití proxy serveru zvážit kompromisy mezi anonymitou a výkonem.

- Právní a etické otázky: Používání proxy serverů pro scraping nebo procházení webu může stále vést k právním nebo etickým problémům, zejména pokud mají cílové webové stránky přísné podmínky služby nebo pokud se uživatel pokouší získat přístup k informacím s omezeným přístupem. Uživatelé by měli vždy zajistit, aby jejich činnost byla v souladu s platnými zákony a předpisy.
- Spolehlivost proxy serverů: Spolehlivost zvoleného proxy serveru hraje významnou roli v celkové účinnosti nástroje. Nespolehlivé proxy servery mohou mít za následek přerušení spojení nebo pomalou odezvu, což negativně ovlivňuje výkon a přesnost nástroje.

Závěrem lze říci, že přidání podpory proxy serverů do nástroje pro detekci přihlašovacích rozhraní poskytuje uživatelům nezbytnou vrstvu anonymity, která pomáhá chránit jejich identitu a snižuje riziko zablokování cílových webových stránek. Zváží-li uživatelé omezení a potenciální problémy spojené s používáním proxy serverů, mohou se informovaně rozhodnout, zda do svých skenovacích aktivit zahrnout proxy servery, či nikoliv.

2.6 Podpora přípon

Jedním z kritických aspektů přesné detekce přihlašovacích rozhraní na webových stránkách je schopnost efektivně skenovat širokou škálu potenciálních adres URL přihlašovacích stránek. Ve výchozím nastavení používá skener přihlašovacích stránek ke generování možných cest k adresám URL pro kontrolu přihlašovacích formulářů předem definovaný seznam slov. Webové aplikace však mohou pro své přihlašovací stránky používat různé přípony souborů nebo mohou mít záložní kopie takových stránek s různými příponami. Tato různorodost přípon souborů může vést k vyšší pravděpodobnosti, že během procesu skenování budou chybět přihlašovací formuláře.

Pro vyřešení tohoto omezení a zlepšení účinnosti skeneru přihlašovacích formulářů byla implementována nová funkce, která umožňuje uživatelům zadávat do seznamu slov další přípony souborů za běhu. Připojením těchto přípon ke každému slovu v seznamu slov pro jedno spuštění může skener pokrýt širší rozsah potenciálních přihlašovacích stránek, aniž by musel trvale měnit soubor se seznamem slov.

Tuto funkci podpory rozšíření lze aktivovat pomocí volby `-x` nebo `-extension` , za kterou následují přípony souborů oddělené mezerami. Například:

```
python3 loginsscanner.py -x "php␣html␣bak"
```

Skener vytvoří rozšířený seznam slov pro aktuální běh tak, že ke každému slovu v původním seznamu slov přidá ".php", ".html"a ".bak". Tento rozšířený seznam slov zvyšuje počet potenciálních adres URL přihlašovacích stránek, které skener

kontroluje, a zvyšuje tak pravděpodobnost odhalení přihlašovacích formulářů, ale i délku běhu skenu. Mezi hlavní výhody funkce podpory rozšíření patří:

- Větší pokrytí: Díky tomu, že uživatelé mohou zadat další přípony souborů, může skener pokrýt širší rozsah možných přihlašovacích stránek, což vede k přesnějším a komplexnějším výsledkům skenování.
- Flexibilita: Tato funkce umožňuje uživatelům přizpůsobit proces skenování specifickým vlastnostem cílových webových stránek, čímž se zvyšuje pravděpodobnost odhalení přihlašovacích formulářů, které mohou používat netradiční přípony souborů nebo mají záložní kopie s různými příponami.
- Efektivita: Funkce podpory rozšíření připojí poskytnutá rozšíření k původnímu seznamu slov pro jedno spuštění, aniž by došlo k trvalé úpravě souboru se seznamem slov. Tento přístup umožňuje uživatelům použít různé sady přípon souborů pro různé relace skenování bez nutnosti udržovat více seznamů slov.
- Snadné použití: Díky začlenění funkce podpory rozšíření jako volby příkazového řádku mohou uživatelé tuto funkci rychle a pohodlně aktivovat, aniž by museli měnit zdrojový kód skeneru nebo soubor seznamu slov.

Souhrnně lze říci, že funkce podpory rozšíření výrazně zvyšuje schopnost skeneru přihlašovacích údajů detekovat přihlašovací rozhraní na webových stránkách tím, že rozšiřuje jeho pokrytí na potenciální přihlašovací stránky s různými příponami souborů. Toto zlepšení přesnosti, flexibility a efektivity činí ze skeneru přihlášení ještě výkonnější a univerzálnější nástroj pro identifikaci přihlašovacích formulářů v různých webových aplikacích.

2.7 Bodovací systém

Místo pouhého vrácení logické hodnoty označující, zda je přítomen přihlašovací formulář, nástroj poskytuje skóre představující pravděpodobnost, že daná stránka obsahuje přihlašovací rozhraní. Systém bodování je založen na počtu nalezených shod pomocí heuristiky a regulárních výrazů, přičemž vyšší skóre znamená vyšší pravděpodobnost, že obsahuje přihlašovací formulář.

2.8 Spidering

Pokud uživatel povolí funkci spideringu, nástroj bude sledovat odkazy na načtených webových stránkách až do zadané hloubky, aby našel další stránky ke kontrole. Tato funkce zvyšuje šanci na objevení přihlašovacích rozhraní díky prozkoumání více stránek na cílových webových stránkách.

2.9 Ukazatel průběhu

Nástroj obsahuje ukazatel průběhu, který se aktualizuje v průběhu skenování a poskytuje uživatelům vizuální informaci o postupu nástroje a zbývající práci, kterou je třeba provést.

2.10 Výstup a serializace

Výsledky skenování, včetně adres URL zjištěných přihlašovacích rozhraní a souvisejících skóre, se ukládají do souboru JSON zadaného uživatelem. Tento výstupní formát umožňuje snadné rozebírání a další analýzu výsledků.

2.11 Ukončení za běhu

Nástroj lze ukončit stisknutím kláves Ctrl+C v příkazovém řádku. Po přijetí signálu ukončení nástroj uloží dosud získané výsledky a až poté se vypne.

2.12 Návod na instalaci

Před použitím nástroje pro detekci webových přihlašovacích rozhraní se ujistěte, že máte v systému nainstalovaný jazyk Python a verzovací systém git. Nástroj byl vyvinut a testován s použitím jazyka Python 3.10, ale měl by být kompatibilní i s jinými verzemi jazyka Python 3.x.

Nástroj stáhnete pomocí příkazu

`git clone https://github.com/Sohrj00/loginscanner`, všechny potřebné knihovny pomocí příkazu

`pip install -r loginscanner/requirements.txt`. Pak už můžete nástroj spustit ve standardní verzi například

`loginscanner/loginscanner.py -u https://vut.cz` nebo ve verzi s webovým rozhraním pomocí

`loginscanner/loginscanner.py -server`.

2.13 Testování a diskuze

V této sekci je popsán proces testování nástroje pro detekci přihlašovacího rozhraní na virtuálních počítačích hostovaných na platformě TryHackMe. TryHackMe je online platforma, která poskytuje reálné, praktické prostředí pro učení a testování konceptů a nástrojů kybernetické bezpečnosti. Virtuální počítače na této platformě

jsou navrženy tak, aby simulovaly různé zranitelnosti a konfigurace, což z ní činí ideální testovací prostředí pro nástroj detekce přihlašovacích rozhraní.

Nastavení testovacího prostředí

Nejprve se nastaví testovací prostředí. To zahrnuje vytvoření účtu na stránce TryHackMe, připojení k VPN, vstoupení do příslušných místností a nasazení cílových virtuálních počítačů. Každý virtuální počítač simuluje jiný typ webové aplikace a je umístěn v izolovaném prostředí.

Provedení testu

Test se provádí zadáním adresy URL virtuálního počítače do webového rozhraní nástroje a spuštěním skenování. Průběh a výsledky nástroje jsou sledovány v reálném čase. Veškeré anomálie nebo problémy, které se během testování vyskytnou, jsou zaznamenány pro další zkoumání a ladění.

Výsledky testů a analýza

Výsledky skenování se porovnávají se známou konfigurací virtuálního počítače, aby se vyhodnotila přesnost nástroje. Mezi parametry, které se při analýze berou v úvahu, patří čas potřebný k dokončení skenování a přesnost detekce. Výsledky je možné vidět v tabulce 2.1.

Stroj	% detekovaných rozhraní	čas
Overpass	100	5:44
Pickle Rick	100	6:21
Internal	0	5:32
Ignite	0	5:56
Brute it	100	6:13
Boiler cti	50	5:48
Daily bugle	100	5:11

Tab. 2.1: Výsledky skenování

Vyskytnuté problémy

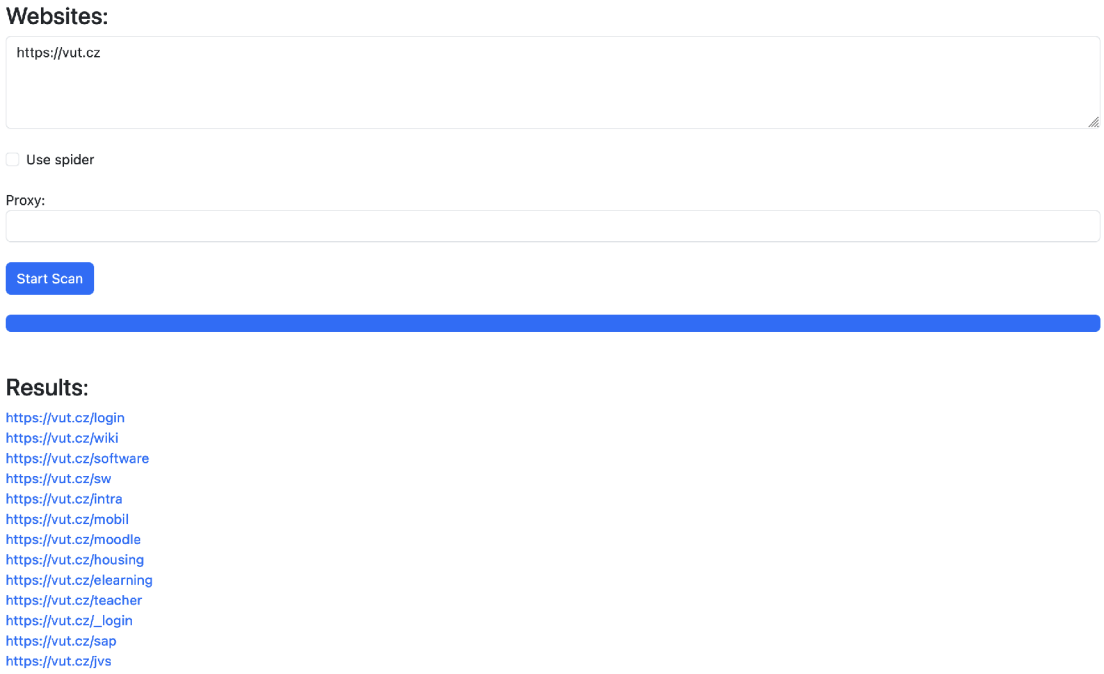
Na stroji Internal nástroj selhal, protože přihlašovací rozhraní bylo schované na URL `/blog/wp-login.php`, jelikož nástroj nepoužívá rekurzi, nebyl schopný najít

takto zanořenou stránku. Po manuální enumeraci a opětovném spuštění nástroje vůči `httt://internal/blog` již přihlašovací rozhraní správně detekoval.

Na stroji Ignite nastala podobná chyba, přihlašovací rozhraní bylo zanořené v `/fuel/login`. Po manuálním prozkoumání stránky byl nalezen CMS fuel, nástroj znovu spuštěn a přihlašovací rozhraní již bylo detekováno.

Nástroj byl také testován přes webové rozhraní vůči `https://vut.cz`. Výsledky tohoto testu jsou zobrazeny na obrázku 2.1.

Obr. 2.1: Výsledky testu ve webovém rozhraní



2.14 Závěr

V této kapitole byl poskytnut ucelený přehled o návrhu a implementaci nástroje pro detekci webového přihlašovacího rozhraní. Pomocí populárních knihoven jazyka Python, threadingu, heuristiky, regulárních výrazů a skórovacího systému nástroj efektivně skenuje webové stránky a detekuje přihlašovací rozhraní. Modularita a snadné použití nástroje z něj činí cenný přínos pro aplikace v oblasti kybernetické bezpečnosti, včetně penetračního testování a skenování zranitelností.

3 Webové rozhraní

Nástroj pro detekci přihlašovacího rozhraní je výkonná a efektivní aplikace určená k identifikaci přihlašovacích rozhraní na seznamu zadaných webových stránek. Vyhledáním těchto rozhraní mohou bezpečnostní odborníci a výzkumníci posoudit zabezpečení webových aplikací, identifikovat potenciální zranitelnosti a přijmout vhodná opatření na ochranu uživatelských dat.

Aby byl tento nástroj přístupnější a uživatelsky přívětivější, hraje zásadní roli dobře navržené webové rozhraní. Díky intuitivnímu webovému rozhraní mohou uživatelé snadno zadávat požadované informace, sledovat průběh procesu skenování a pohodlně si prohlížet výsledky. Responzivní design zajišťuje, že webovou aplikaci lze používat na různých zařízeních, včetně stolních počítačů, notebooků, tabletů a chytrých telefonů.

V této kapitole budou probrány technologie použitýmé pro vytvoření webového rozhraní, včetně Vue.js pro vývoj frontendů, Bootstrap pro responzivní design a stylování a Socket.IO pro komunikaci mezi klientem a serverem v reálném čase. Také bude popsán proces návrhu a implementace a integrace s nástrojem pro detekci přihlašovacího rozhraní.

3.1 Flask

Flask je lehký a flexibilní webový framework v jazyce Python, který vývojářům umožňuje rychle a efektivně vytvářet webové aplikace. Flask poskytuje jednoduché a snadno použitelné rozhraní API pro zpracování požadavků a odpovědí HTTP a také vestavěnou podporu pro obsluhu statických souborů, šablonování a směrování. Flask byl pro tento projekt vybrán, protože se snadno nastavuje a integruje se stávajícím skriptem Pythonu, což umožnilo rychle vytvořit webové rozhraní pro skener přihlášení.

V tomto webovém rozhraní se Flask používá k obsluze statických souborů (CSS, JavaScript a HTML) a poskytuje rozhraní API pro spouštění nových skenů a načítání výsledků. Aplikace Flask je nastavena s jedinou trasou, která vykresluje hlavní soubor index.html. Tento soubor obsahuje aplikaci Vue.js, která pohání interaktivní prvky webového rozhraní.

3.2 Flask-SocketIO

Flask-SocketIO je rozšíření pro Flask, které zjednodušuje integraci WebSocketů do aplikací Flask. WebSockets umožňují obousměrnou komunikaci v reálném čase mezi

klientem (prohlížečem) a serverem, což nám umožňuje aktualizovat průběh a výsledky skenování přihlášení v reálném čase bez nutnosti dotazování nebo obnovování stránky.

V tomto webovém rozhraní se k navázání spojení WebSocket mezi backendem Flask a frontendem Vue.js používá Flask-SocketIO. Když uživatel zahájí nové skenování, frontend odešle backendu prostřednictvím spojení WebSocket událost `start_scan`. Backend poté spustí skript pro skenování přihlášení a vyšle události `scan_progress` a `scan_complete` zpět do frontendu, který aktualizuje ukazatel průběhu a zobrazí výsledky.

3.3 Vue.js

Vue.js je progresivní framework JavaScriptu pro vytváření uživatelských rozhraní. Vue.js je navržen tak, aby se dal snadno integrovat do stávajících projektů a umožnil vývojářům přidávat na webové stránky reaktivní a interaktivní prvky s minimálním úsilím. Vue.js poskytuje jednoduché a výkonné rozhraní API pro vazbu dat, zpracování událostí a vývoj založený na komponentách, což z něj činí vynikající volbu pro vytvoření webového rozhraní pro náš přihlašovací skener. V tomto webovém rozhraní se Bootstrap používá ke stylování různých prvků uživatelského rozhraní, jako jsou vstupní pole, tlačítka, ukazatele průběhu a oblasti výsledků. CSS Bootstrap je obsažen v CDN a příslušné třídy Bootstrap jsou aplikovány na prvky HTML v šabloně Vue.js. Díky tomu lze snadno vytvořit vizuálně atraktivní a responzivní webové rozhraní.

3.4 Návrh uživatelského rozhraní

Uživatelské rozhraní nástroje pro detekci přihlašovacího rozhraní je navrženo tak, aby bylo jednoduché a intuitivní. Zahrnuje různé prvky a komponenty, které usnadňují interakci s uživatelem a organizovaně prezentují potřebné informace. Tento oddíl popisuje klíčové aspekty návrhu uživatelského rozhraní.

- Rozvržení a struktura: Rozložení webového rozhraní je strukturováno tak, aby poskytovalo jasnou vizuální hierarchii a vedlo uživatele procesem skenování. Rozhraní obsahuje záhlaví obsahující název nástroje, za nímž následují vstupní pole pro webové stránky, zaškrtávací políčko pro volbu povolení spideringu a pole pro zadání proxy serveru. Pod formulářem se zobrazuje ukazatel průběhu a po dokončení skenování jsou výsledky prezentovány v seznamu. Toto rozvržení zajišťuje, že uživatel nástroj snadno pochopí a může s ním pracovat.

- Prvky formuláře: Pro zahájení procesu skenování je uživatel povinen zadat seznam webových stránek, vybrat, zda chce použít spidering, a v případě potřeby zadat proxy server. Prvky formuláře jsou navrženy tak, aby byly uživatelsky přívětivé a vizuálně konzistentní, což uživatelům usnadňuje zadávání požadovaných informací. Formulář používá validaci, která zajišťuje správnost a úplnost vstupních údajů před jejich odesláním na server.
- Lišta průběhu: Protože proces skenování může nějakou dobu trvat, je součástí formuláře ukazatel průběhu, který uživatelům vizuálně znázorňuje průběh skenování. Lišta průběhu se aktualizuje v reálném čase pomocí Socket.IO pro příjem aktualizací průběhu ze serveru. Tato funkce udržuje uživatele informované při čekání na dokončení skenování.
- Zobrazení výsledků: Výsledky skenování se zobrazují v seznamu pod ukazatelem průběhu. Každá položka v seznamu je odkazem na zjištěné přihlašovací rozhraní, což uživatelům umožňuje snadný přístup ke zjištěným přihlašovacím stránkám a jejich prohlížení. Použití Vue.js umožňuje dynamickou aktualizaci seznamu výsledků při zjištění nových přihlašovacích rozhraní, což uživatelům poskytuje aktualizace v reálném čase.
- Odezva: Díky využití frameworku Bootstrap je uživatelské rozhraní navrženo tak, aby bylo responzivní a přívětivé pro mobilní zařízení. To zajišťuje, že nástroj je přístupný a použitelný na široké škále zařízení a velikostí obrazovky, a poskytuje tak konzistentní uživatelský zážitek bez ohledu na použité zařízení.

Uživatelské rozhraní nástroje pro detekci přihlašovacího rozhraní je navrženo tak, aby uživatelům poskytovalo bezproblémové a poutavé prostředí. Díky začlenění intuitivních formulářových prvků, aktualizacím průběhu v reálném čase a responzivnímu designu rozhraní zajišťuje, že uživatelé mohou s nástrojem snadno pracovat a rychle se dostat k požadovaným informacím.

3.5 Implementace Vue.js a Socket.IO

Webové rozhraní nástroje pro detekci přihlašovacího rozhraní využívá Vue.js a Socket.IO k vytvoření dynamického uživatelského prostředí v reálném čase. Tato část se zabývá implementací těchto technologií a tím, jak přispívají k celkové funkčnosti nástroje.

Integrace Vue.js

Vue.js je progresivní framework JavaScriptu, který se používá k vytváření uživatelských rozhraní. V tomto nástroji je Vue.js použit ke správě stavu frontendu, zpracování uživatelských vstupů a dynamické aktualizaci rozhraní. Aplikace Vue.js je vytvořena a připojena k prvku *Document Object Model* (DOM) s ID "app", kde

spravuje data a metody související s procesem skenování. Reaktivní datový systém Vue.js zajišťuje, že se rozhraní automaticky aktualizuje, kdykoli se změní podkladová data, a poskytuje tak uživateli zpětnou vazbu v reálném čase.

Zpracování uživatelských vstupů

Vue.js umožňuje nástroji zachycovat vstupy od uživatele a ukládat je do datových vlastností aplikace. Pomocí direktivy v-model se prvky formuláře vážou na odpovídající datové vlastnosti, což zajišťuje, že vstup uživatele je automaticky aktualizován ve stavu aplikace. Když uživatel odešle formulář, spustí se metoda submitForm, která na server vyšle událost 'start_scan' se shromážděnými vstupními daty.

Integrace Socket.IO

Socket.IO je knihovna, která umožňuje komunikaci mezi klientem a serverem v reálném čase. V tomto nástroji se Socket.IO používá k navázání spojení se serverem, což klientovi umožňuje přijímat aktualizace o průběhu skenování a zjištěných přihlašovacích rozhraních. Klient naslouchá různým událostem vysílaným serverem, například "scan_progress", "room_joined"a "scan_complete", a podle toho aktualizuje vlastnosti dat aplikace.

Aktualizace průběhu v reálném čase

Jak server zpracovává skenování, vysílá události "scan_progress"obsahující aktuální průběh a dílčí výsledky. Klient naslouchá těmto událostem a aktualizuje datové vlastnosti aplikace "progress"a "results", které jsou vázány na ukazatel průběhu a seznam výsledků v rozhraní. Díky tomu se lišta průběhu aktualizuje v reálném čase a poskytuje uživatelům vizuální informaci o průběhu skenování.

Zobrazení výsledků

Po přijetí události "scan_complete"ze serveru klient aktualizuje datovou vlastnost "results"s konečným seznamem zjištěných přihlašovacích rozhraní. Systém reaktivity Vue.js zajišťuje, že se seznam výsledků v rozhraní automaticky aktualizuje a zobrazí uživateli zjištěné přihlašovací stránky.

3.6 Závěr

V této kapitole byly probrány různé technologie a techniky použité k vytvoření webového rozhraní pro aplikaci přihlašovacího skeneru. S využitím technologií Flask,

Flask-SocketIO, Vue.js a Bootstrap bylo vytvořeno interaktivní a uživatelsky přívětivé rozhraní, které uživatelům umožňuje konfigurovat a spouštět několik skenů přihlášení, zobrazovat aktualizace průběhu v reálném čase a získávat výsledky.

Integrace Flask a Flask-SocketIO v backendu umožňuje aplikaci efektivně zpracovávat požadavky HTTP a připojení WebSocket, což zajišťuje efektivní komunikaci mezi klientem a serverem. Tato komunikace v reálném čase je klíčová pro poskytování aktualizací průběhu a poskytování výsledků skenování uživateli bez nutnosti neustálého dotazování nebo obnovování stránky.

Na frontendové straně zjednodušuje Vue.js proces vytváření dynamické jednostránkové aplikace, která reaguje na vstupy uživatele a podle toho aktualizuje prvky uživatelského rozhraní. Pomocí Vue.js lze snadno spravovat stav aplikace, zpracovávat události uživatele a aktualizovat DOM výkonným a udržovatelným způsobem. Architektura Vue.js založená na komponentách také podporuje znovupoužitelnost a modularitu kódu, což usnadňuje budoucí rozšíření nebo úpravy aplikace.

Bootstrap jako framework CSS nám pomáhá vytvořit vizuálně atraktivní a responzivní webové rozhraní s minimálním úsilím. Využitím předpřipravených stylů a komponent frameworku Bootstrap můžeme dosáhnout profesionálně vypadajícího designu a zajistit, aby naše webové rozhraní vypadalo dobře na různých zařízeních a při různých velikostech obrazovky.

Díky implementaci Vue.js a Socket.IO poskytuje nástroj pro detekci přihlašovacích rozhraní dynamické uživatelské prostředí v reálném čase, které uživatelům umožňuje získat okamžitou zpětnou vazbu o procesu skenování a přístup k detekovaným přihlašovacím rozhraním. Tyto technologie hrají klíčovou roli v celkové funkčnosti nástroje a zajišťují uživatelům bezproblémový a poutavý zážitek.

Celkově je možno říci, že kombinace těchto technologií umožnila vytvořit robustní a uživatelsky přívětivé webové rozhraní pro aplikaci přihlašovacího skeneru, díky čemuž je přístupnější a použitelnější pro širší publikum. Webové rozhraní nejen zjednodušuje proces spouštění skenování přihlašovacího systému, ale také poskytuje platformu pro budoucí rozšiřování a vylepšování, například přidání podpory pro další funkce skenování nebo integraci s dalšími webovými bezpečnostními nástroji.

4 Výzvy a omezení

Přestože nástroj pro detekci webového přihlašovacího rozhraní nabízí výkonné řešení pro detekci přihlašovacích formulářů, není bez problémů a omezení. Uvědomění si těchto problémů a pochopení omezení nástroje je nezbytné pro efektivní využití nástroje a interpretaci jeho výsledků.

Dynamický obsah webu: Stále častější výskyt dynamického webového obsahu, zejména díky používání jazyka JavaScript, může nástroji ztížit přesnou detekci přihlašovacích formulářů. Nástroj může mít problémy se stránkami, které načítají obsah asynchronně nebo používají složitou logiku JavaScriptu. Tento problém by mohla pomoci vyřešit budoucí vylepšení, například začlenění bezhlavého prohlížeče nebo funkce vykreslování JavaScriptu.

Falešně pozitivní a falešně negativní výsledky: Použití heuristiky a regulárních výrazů pro detekci přihlašovacích formulářů může mít za následek falešně pozitivní výsledky (tj. nepřihlašovací stránky nesprávně identifikované jako stránky obsahující přihlašovací formuláře) a falešně negativní výsledky (tj. přihlašovací stránky, které nástroj nezjistil). Přestože systém bodování pomáhá tento problém zmírnit tím, že poskytuje skóre pravděpodobnosti, je nutné si uvědomit, že výsledky nástroje nemusí být vždy dokonalé.

Vlastní přihlašovací formuláře: Některé webové stránky mohou používat vlastní přihlašovací formuláře, které se nedrží běžných vzorů nebo používají jedinečné atributy. Tyto vlastní formuláře může být obtížné odhalit pomocí předdefinovaných heuristik a regulárních výrazů. Umožnění uživatelům definovat vlastní heuristiky a regulární výrazy by mohlo pomoci toto omezení vyřešit.

Etické aspekty: Použití nástroje pro detekci webového přihlašovacího rozhraní ke skenování webových stránek bez souhlasu jejich vlastníka může vyvolat etické obavy a potenciálně porušit platné zákony nebo předpisy. Při používání nástroje pro hodnocení zabezpečení, penetrační testování nebo jiné aplikace v oblasti kybernetické bezpečnosti je nezbytné získat řádné oprávnění a dodržovat etické zásady.

Uvědomíme-li si tyto problémy a omezení, mohou uživatelé nástroje pro detekci webového přihlašovacího rozhraní činit informovanější rozhodnutí při interpretaci výsledků nástroje a aplikaci jeho zjištění na reálné scénáře. Pochopení těchto problémů navíc může být vodítkem pro budoucí vylepšování a vývoj s cílem zlepšit schopnosti nástroje a odstranit jeho omezení.

5 Návrhy pro budoucí práci

Přestože nástroj pro detekci webového přihlašovacího rozhraní poskytuje robustní řešení pro detekci přihlašovacích formulářů, existují potenciální vylepšení a rozšíření, která by mohla dále zlepšit jeho funkčnost a výkon.

Přístup strojového učení: Začlenění technik strojového učení, například trénování klasifikátoru pomocí prvků extrahovaných z webových stránek, by mohlo zvýšit přesnost a odvolatelnost nástroje. Ke klasifikaci webových stránek jako obsahujících nebo neobsahujících přihlašovací rozhraní by mohly být použity algoritmy učení s dohledem, například logistická regrese, stroje s podpůrnými vektory nebo modely hlubokého učení.

Lepší práce se stránkami náročnými na JavaScript: Moderní webové stránky se při vykreslování obsahu a vytváření dynamických uživatelských rozhraní často ve velké míře spoléhají na JavaScript. Vylepšení nástroje o funkce vykreslování JavaScriptu, například integrace bezhlavého prohlížeče, jako je Selenium, nebo použití knihovny, jako je Pyppeteer, by mohlo zlepšit detekci přihlašovacích formulářů na stránkách náročných na JavaScript.

Integrace se skenery webových aplikací: Integrace nástroje pro detekci webového přihlašovacího rozhraní se stávajícími skenery webových aplikací nebo nástroji pro hodnocení zranitelností by mohla poskytnout komplexnější bezpečnostní analýzu. Spojením detekce přihlašovacích rozhraní se skenováním zranitelností by výzkumníci v oblasti bezpečnosti mohli efektivněji identifikovat potenciální vstupní body a související zranitelnosti.

Přizpůsobitelné heuristiky a regulární výrazy: Umožnění uživatelům definovat vlastní heuristiky a regulární výrazy by mohlo nástroji umožnit přizpůsobit se konkrétním případům použití nebo se zaměřit na konkrétní typy přihlašovacích formulářů. Toto přizpůsobení by mohlo zlepšit přesnost a výdrž nástroje, zejména v situacích, kdy výchozí heuristika a regulární výrazy nestačí.

Omezení rychlosti: Přidání funkcí omezování rychlosti by mohlo nástroji pomoci lépe zvládat situace, kdy webové servery zavádějí omezení počtu nebo frekvence požadavků. Díky tomuto vylepšení by byl nástroj odolnější a dokázal by si poradit s širší škálou webových stránek a webových aplikací.

Kontinuální integrace a testování: Zavedení systému kontinuální integrace (CI), který by automaticky testoval nástroj na různých webových stránkách, by zajistilo jeho trvalou spolehlivost a účinnost. Pravidelná aktualizace souboru dat a základních údajů používaných pro testování by rovněž pomohla udržet přesnost a výkonnost nástroje.

Začleněním těchto vylepšení a rozšíření by se nástroj pro detekci webových přihlašovacích rozhraní mohl stát ještě výkonnějším a univerzálnějším řešením pro detekci

přihlašovacích formulářů a podporu úsilí o kybernetickou bezpečnost.

Závěr

Nástroj pro detekci webového přihlašovacího rozhraní představuje výkonné a všestranné řešení pro detekci přihlašovacích formulářů na webových stránkách a ve webových aplikacích. Díky použití heuristiky, regulárních výrazů a skórovacího systému dokáže nástroj přesně identifikovat přihlašovací rozhraní a podporovat různé aplikace kybernetické bezpečnosti, jako je testování průniku, hodnocení zranitelnosti a bezpečnostní audity.

Modulární konstrukce nástroje a použití populárních knihoven jazyka Python, jako jsou Requests, Beautiful Soup a Click, usnadňují jeho rozšíření a přizpůsobení pro konkrétní případy použití. Jeho rozhraní příkazového řádku a podpora vstupních souborů, jednotlivých adres URL a volitelných funkcí pavoukování navíc umožňují přizpůsobení různým potřebám a preferencím uživatelů.

Navzdory své účinnosti se nástroj pro detekci webového přihlašovacího rozhraní potýká s několika problémy a omezeními, včetně zpracování dynamického webového obsahu, falešně pozitivních a falešně negativních výsledků, vlastních přihlašovacích formulářů, omezení rychlosti a etických aspektů. Pokud si tyto problémy uvědomíme a budeme se je snažit řešit prostřednictvím aktualizací, vylepšení a průběžné údržby, může se nástroj dále vyvíjet a zůstat cenným zdrojem v oblasti kybernetické bezpečnosti.

Začlenění technik strojového učení, zlepšení možností vykreslování v jazyce JavaScript, integrace se skenery webových aplikací, umožnění přizpůsobitelných heuristik a regulárních výrazů a přidání omezení rychlosti a podpory proxy serverů jsou potenciální vylepšení, která by mohla dále zlepšit funkčnost a výkonnost nástroje.

Nástroj pro detekci webového přihlašovacího rozhraní nakonec slouží jako cenný zdroj informací pro odborníky na kybernetickou bezpečnost, výzkumníky i studenty. Pochopením jeho schopností, omezení a potenciálních aplikací mohou uživatelé nástroj efektivně využívat pro podporu svých bezpečnostních snah, získat přehled o zabezpečení webových aplikací a přispět k neustálému vývoji osvědčených postupů a řešení v oblasti kybernetické bezpečnosti.

Literatura

[1] OWASP: *WSTG* [online]. [cit. 28. 4. 2023] Dostupné z URL: <https://owasp.org/www-project-web-security-testing-guide/latest/3-The_OWASP_Testing_Framework/1-Penetration_Testing_Methodologies>.

[2] OWASP: *Broken Authentication* [online]. [cit. 28. 4. 2023] Dostupné z URL: <https://owasp.org/www-project-top-ten/2017/A2_2017-Broken_Authentication>.

[3] OWASP: *Authentication Cheat Sheet* [online]. [cit. 1. 5. 2023] Dostupné z URL: <https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html>.

[4] OWASP: *Credential stuffing* [online]. [cit. 1. 5. 2023] Dostupné z URL: <https://owasp.org/www-community/attacks/Credential_stuffing>.

[5] Furnell, Steven and Millet, Kieran and Papadaki, Maria, Fifteen years of phishing: can technology save us?, *Computer Fraud & Security*, vol. 2019, no. 7, pp. 11–16, 2019. https://doi.org/10.1016/S1361-3723(19)30074-0

[6] *Keyloggers: How they work and how to detect them (Part 1)* [online]. [cit. 1. 5. 2023] Dostupné z URL: <https://securelist.com/keyloggers-how-they-work-and-how-to-detect-them-part
36138/>.

[7] CloudFlare: *Monsters in the Middleboxes: Introducing Two New Tools for Detecting HTTPS Interception* [online]. [cit. 1. 5. 2023] Dostupné z URL: <https://blog.cloudflare.com/monsters-in-the-middleboxes/>.

[8] CrowdStrike: *Password spraying* [online]. [cit. 1. 5. 2023] Dostupné z URL: <https://www.crowdstrike.com/cybersecurity-101/password-spraying/>.

[9] OWASP: *SQL Injection* [online]. [cit. 1. 5. 2023] Dostupné z URL: <https://owasp.org/www-community/attacks/SQL_Injection>.

[10] OWASP: *Session hijacking* [online]. [cit. 1. 5. 2023] Dostupné z URL: <https://owasp.org/www-community/attacks/Session_hijacking_attack>.

[11] OWASP: *Social engineering* [online]. [cit. 1. 5. 2023] Dostupné z URL: <https://owasp.org/www-pdf-archive/Presentation_Social_Engineering.pdf>.

[12] Zheng, Shuyi and Song, Ruihua and Wen, Ji-Rong and Wu, Di, Joint Optimization of Wrapper Generation and Template Detection, In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pp. 894–902, 2007, Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/1281192.1281287

[13] Aho, Alfred V.. van Leeuwen, Jan. Algorithms for finding patterns in strings. Handbook of Theoretical Computer Science, volume A: Algorithms and Complexity. The MIT Press. pp. 255–300.

[14] Shah, S., Mehtre, B.M. An overview of vulnerability assessment and penetration testing techniques. J Comput Virol Hack Tech 11, 27–49 (2015). https://doi.org/10.1007/s11416-014-0231-x

[15] Vats, Prashant and Mandot, Manju and Gosain, Anjana, A Comprehensive Literature Review of Penetration Testing & Its Applications, In *2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, pp. 674–680, 2020. https://doi.org/10.1109/ICRITO48877.2020.9197961

[16] Castillo, Carlos, Effective web crawling, *SIGIR Forum*, vol. 39, pp. 55–56, 2005. https://doi.org/10.1145/1067268.1067287

[17] *Dokumentace selenium* [online]. Dostupné z URL: https://www.selenium.dev/documentation/webdriver/

[18] PortSwigger: *Dokumentace Burp Suite* [online]. Dostupné z URL: https://portswigger.net/

[19] OWASP: *Dokumentace OWASP ZAP* [online]. Dostupné z URL: https://www.zaproxy.org/docs/

[20] NMAP: *Dokumentace nmap* [online]. Dostupné z URL: https://nmap.org/docs.html

[21] CRUMMY: *Dokumentace knihovny BeautifulSoup* [online]. Dostupné z URL: https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[22] SCRAPY: *Dokumentace nástroje Scrapy* [online]. Dostupné z URL: https://docs.scrapy.org/en/latest/

[23] Read the Docs: *Dokumentace knihovny Requests* [online]. Dostupné z URL: `https://requests.readthedocs.io/en/latest/`

Seznam symbolů a zkratek

HTML	HyperText Markup Language
SQL	Structured query language
XSS	Cross-site scripting
CSRF	Cross-site request forgery
CAPTCHA	completely automated public Turing test to tell computers and humans apart
MFA	Multi-factor Authentication
MITM	Man-in-the-middle
OTP	One-time password
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
SSL	Secure Sockets Layer
HSTS	HTTP Strict Transport Security
WAF	Web Application Firewall
ZAP	Zed Attack Proxy
OWASP	Open Web Application Security Project
AJAX	Asynchronous JavaScript and XML
Nmap	Network Mapper
XML	Extensible Markup Language
API	Application Programming Interface
CSS	Cascading Style Sheets
JSON	JavaScript Object Notation
CSV	Comma Separated Values
URL	Uniform Resource Locator
DOM	Document Object Model

Seznam příloh

A Obsah elektronické přílohy

Elektronická příloha obsahuje samotný skript, šablonu html stránky webového rozhraní a výchozí slovník pro enumeraci.

- `/` ..... kořenový adresář přiloženého archivu
  - `loginscanner.py` ..... skript pro detekci přihlašovacích rozhraní
  - `raft` ..... Výchozí slovník
  - `requirements.txt` ..... Verze požadovaných knihoven jazyka pyzhon
  - `templates` ..... Složka obsahující šablonu pro webové rozhraní
    - `index.html` ..... Šablona pro webové rozhraní