DOI 10.20535/2411-1031.2024.12.1.306260
УДК 004.056

АНАСТАСІЯ ТОЛКАЧОВА,
ДАНИЇЛ ЖУРАВЧАК

АВТОМАТИЗАЦІЯ ПЕРЕВІРКИ АТРИБУТІВ СЕСІЙНИХ ФАЙЛІВ COOKIES

У цьому дослідженні ми зосереджуємо увагу на критично важливій темі веб-безпеки, а саме безпеці сесійних файлів cookies, які відіграють ключову роль у функціонуванні сучасних веб-додатків. Cookies, що є стандартним механізмом для зберігання даних на стороні клієнта, мають вирішальне значення для процесів автентифікації, авторизації та підтримки стану сесії користувача. Проте, не дивлячись на їхню необхідність і зручність, файли cookies також можуть становити серйозні ризики безпеки. Основна увага в нашому дослідженні приділяється аналізу та автоматизації перевірки атрибутів cookies, що є критично важливим для забезпечення захисту від різноманітних веб-атак. Виявлення і усунення слабких місць в атрибутах cookies може значно зменшити ризик зловмисних атак, таких як крадіжка сесій, атаки міжсайтового скриптингу (XSS) та атаки міжсайтового підроблення запитів (CSRF). Ми детально розглядаємо сучасні методи та інструменти для забезпечення безпеки cookies, включаючи впровадження строгих політик щодо атрибутів cookies, таких як Secure, HttpOnly та SameSite. Ці атрибути допомагають обмежити доступ до файлів cookies від неавторизованого використання через клієнтські скрипти, а також забезпечують додатковий захист від міжсайтових атак. Окрім того, ми розглядаємо важливість оновлення стандарту cookies, RFC6265bis, який пропонує поліпшені механізми безпеки, включно з атрибутом SameSite, що дозволяє контролювати надсилання cookies при перехресних запитах, тим самим знижуючи ризик CSRF атак. Наше дослідження також включає аналіз потенційних загроз і вразливостей, асоційованих з неправильним використанням або налаштуванням cookies, а також обговорення стратегій мінімізації цих ризиків. Ми демонструємо, як через детальну автоматизовану перевірку атрибутів cookies можна значно підвищити рівень безпеки веб-додатків. Результати дослідження вказують на необхідність постійного моніторингу та оцінки захисту сесійних файлів cookies, а також на важливість впровадження передових практик і стандартів безпеки для забезпечення надійності та безпеки веб-додатків.

Ключові слова: файли cookie, вразливості, CSRF, веб-додаток, безпека, сесія.

Постановка проблеми. Постановка проблеми у цьому дослідженні зосереджується на автоматизації виявлення та аналізі вразливостей веб-сесій cookie, які становлять значний ризик для конфіденційності та безпеки користувачів в Інтернеті. Незважаючи на розробку численних механізмів захисту, таких як атрибути Secure та SameSite для cookie та захист від міжсайтової фальсифікації запитів (CSRF), існуючі рішення не завжди ефективні проти усіх типів атак. Це створює потребу в детальному розумінні можливих векторів атак та розробці вдосконалених методів захисту, що враховують сучасні загрози і тенденції розвитку веб-технологій.

Аналіз останніх досліджень і публікацій. Вивчаючи найновіші наукові дослідження щодо сеансових файлів cookie та аспектів їх безпеки, слід зупинитись на кількох ключових моментах, які були отримані в ряді важливих публікацій.

Стаття J.S. Park і Ravi Sandhu “Secure Cookies on the Web” розглядає проблематику забезпечення безпеки cookie у веб-додатках. Автори обговорюють вразливість звичайних cookie, які зберігаються і передаються у незашифрованому вигляді, що робить їх доступними для зловмисників [1]. Для вирішення цієї проблеми, вони пропонують концепцію безпечних сеансів, які використовують криптографічні методи для забезпечення аутентифікації,

цілісності та конфіденційності даних. Такі cookie можуть містити зашифровану інформацію, таку як роль користувача або кредитну картку, і мають механізми перевірки, що запобігають несанкціонованому доступу та модифікації. Автори також розглядають різні сценарії застосування безпечних cookie, включно з електронною комерцією та контролем доступу на основі атрибутів [1].

У статті Hyunsoo Kwon та ін. "(In-)Security of Cookies in HTTPS: Cookie Theft by Removing Cookie Flags" досліджується проблематика безпеки cookie у середовищі HTTPS. Автори описують новий вид атаки, "rotten cookie", який дозволяє зловмисникам деактивувати флаги безпеки куків, навіть якщо вони захищені за допомогою TLS, використовуючи вразливості протоколу HTTP. Аналізуються недоліки в реалізації веб-браузерів, які ігнорують неправильні частини HTTP-відповідей та приймають формати без відкидання, що може призвести до видалення атрибутів безпеки і, як наслідок, до крадіжки cookie. Пропонується стратегія пом'якшення для захисту сеансу на транспортному рівні [2].

Khu-Smith і Mitchell (2002) "Enhancing the Security of Cookies". Ця робота розглядає безпеку cookie у веб-застосунках, ідентифікуючи ключові загрози, такі як конфіденційність даних у сеансових файлах, моніторинг поведінки користувачів та зловмисні дії. Автори обговорюють вимоги до безпеки, включаючи конфіденційність, цілісність та аутентифікацію, а також розглядають різні технічні підходи для забезпечення цих вимог, такі як застосування криптографії всередині сеансових файлів [3]. Подальший аналіз включає порівняння серверного керування шифруванням та керуванням шифруванням з боку користувачів, висвітлюючи переваги і недоліки кожного підходу. Автори також розглядають реалізацію протоколів для захисту cookie, що керуються користувачами, використовуючи симетричну та асиметричну криптографію [3].

Метою статті є глибокий аналіз вразливостей веб-сесій і безпеки cookie, що включає вивчення існуючих методів захисту та їх потенційних слабких місць. У контексті зростаючої складності веб-застосунків та високих вимог до їх безпеки, це дослідження спрямоване на виявлення нових векторів атак і розробку рекомендацій для їх нейтралізації. Дослідження охоплює аналіз сучасних підходів до захисту сесій та cookie, включаючи атрибути Secure, SameSite та механізми захисту від CSRF [4], з метою вдосконалення існуючих заходів безпеки. Особлива увага буде приділена розробленню автоматизованої системи, яка дозволяє перевіряти наявність необхідних атрибутів безпеки в cookies: Secure, SameSite та HttpOnly. Ця система буде сканувати HTTP-відповіді сервера на предмет наявності та правильності встановлення згаданих атрибутів, які є критично важливими для забезпечення конфіденційності та цілісності сесій користувачів.

Виклад основного матеріалу дослідження. HTTP cookies є найстарішим використовуваним механізмом для обміну станом між веб-клієнтами і серверами. Вони є важливою частиною веб-сесій та відіграють вирішальну роль у автентифікації та авторизації користувачів. Незважаючи на їх важливість у веб-додатках, cookie мають довгу історію вразливостей та декілька відомих недоліків. Існують цілі класи атак, пов'язаних з порушенням конфіденційності або цілісності кукісів. Наприклад, атаки перехоплення сесії спрямовані на розголошення значення куки сесії і використання його для несанкціонованого доступу до веб-сайту. Атаки на підробку міжсайтового запиту (CSRF), натомість, є типовою проблемою порушення цілісності сесії, де зловмисник видає міжсайтові запити з браузера жертви для виконання небажаних дій на веб-сайті, на якому користувач автентифікований.

У відповідь на ці атаки, було запропоновано нові механізми як на стороні клієнта, так і на стороні сервера. На стороні клієнта основні веб-браузери тепер підтримують оновлений стандарт RFC6265bis [5], який включає розширені засоби безпеки порівняно з оригінальним RFC від 2011 року. Прикладом є атрибут SameSite, який рекламується як надійний спосіб захисту від атак CSRF. Інші зміни спрямовані на посилення цілісності сеансових файлів щодо атак в межах того ж сайту та мережі, з покращеннями в атрибуті Secure та введенням префіксів імен __Host- та __Securecookie. На стороні сервера традиційні заходи захисту від атак CSRF

включають в себе використання секретного токена, який обмінюється між браузерами і серверами. Цей підхід був широко прийнятий популярними веб-фреймворками і розглядається як ефективний захист у варіанті схеми синхронізатора токену.

HTTP-файл cookie (веб-файл cookie, файл cookie браузера) – це невеликий фрагмент даних, який сервер надсилає веб-браузеру користувача. Браузер може зберігати файл cookie і надсилати його назад на той самий сервер при наступних запитах. Зазвичай HTTP-файл cookie використовується для того, щоб визначити, чи два запити надходять від одного і того ж браузера – наприклад, для того, щоб користувач увійшов в систему. Він запам'ятовує інформацію про стан для протоколу HTTP.

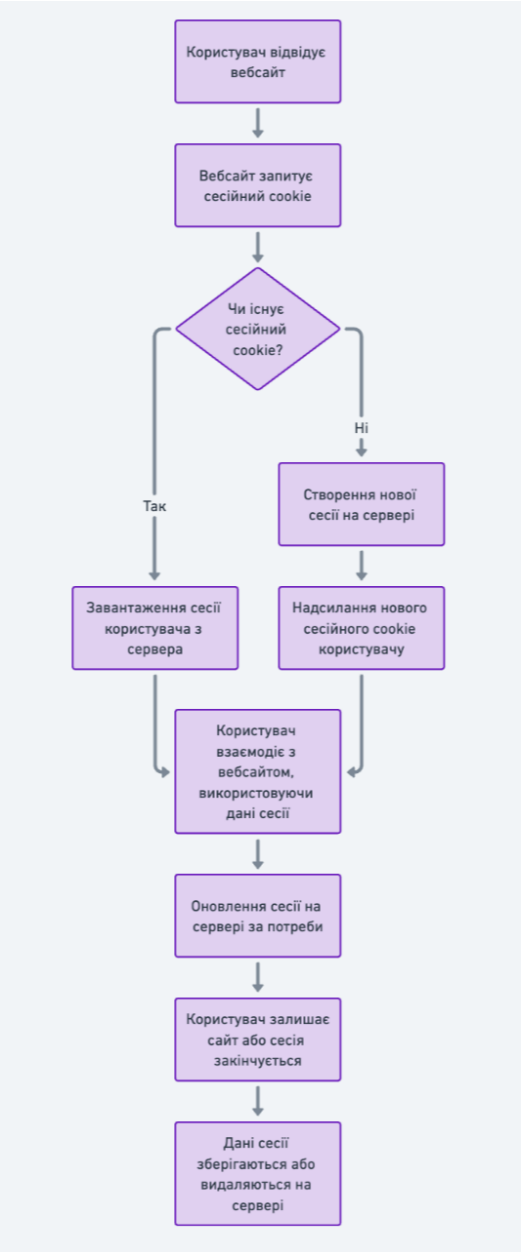

Рисунок 1 – Схема роботи сесійних файлів cookie

Файли cookie в основному використовуються для трьох цілей:

– *керування сеансами*: логіни, кошики для покупок, рахунки в іграх або будь-що інше, що має запам'ятовуватися сервером;

– *персоналізація*: уподобання користувача, теми та інші налаштування;

– *відстеження*: запис та аналіз поведінки користувача.

Колись файли cookie використовувалися для загального зберігання даних на стороні клієнта. Хоча це мало сенс, коли вони були єдиним способом зберігання даних на клієнті, зараз рекомендується використовувати сучасні API для зберігання даних. Файли cookie надсилаються з кожним запитом, тому вони можуть погіршити продуктивність (особливо для мобільних з'єднань) [6]. Сучасні API для зберігання даних на клієнті – це Web Storage API (localStorage і sessionStorage) і IndexedDB.

Створення файлів cookie. Після отримання HTTP-запиту сервер може надіслати один або декілька заголовків Set-Cookie у відповідь. Зазвичай браузер зберігає файл cookie і надсилає його разом із запитами на той самий сервер у HTTP-заголовку Cookie. Є можливість вказати дату закінчення терміну дії або період часу, після якого файл cookie не повинен надсилатися. Також можливо встановити додаткові обмеження на певний домен і шлях, щоб обмежити місце надсилання файлів cookie.

Заголовки Set-Cookie та Cookie. Заголовок HTTP-відповіді Set-Cookie надсилає файли cookie від сервера до агента користувача. Простий файл cookie встановлюється так:

```
Set-Cookie: <cookie-ім’я>=<cookie-значення>
```

Це вказує серверу, що надсилає заголовки, повідомити клієнту, щоб він зберіг пару файлів cookie:

```
HTTP/2.0 200 OK
Content-Type: text/html
Set-Cookie: cookie=abcdfgw
Set-Cookie: cookie=strawberry

[сторінка]
```

Потім, при кожному наступному запиті до сервера, браузер відправляє всі раніше збережені файли cookie назад на сервер за допомогою заголовка Cookie.

```
GET  /page.html HTTP/2.0
Host: www.example.org
Cookie: cookie= abcdfgw; cookie=strawberry
```

Визначення часу життя файлу cookie. Файли cookie можуть зберігатися протягом двох різних періодів, залежно від атрибутів, використаних у заголовку Set-Cookie при їх створенні:

– *Постійні файли cookie* видаляються в дату, зазначену в атрибуті Expires, або через період, визначений атрибутом Max-Age.

– *Сесійні файли cookie* – файли cookie без атрибутів Max age або Expires – видаляються після завершення поточного сеансу. Браузер визначає, коли закінчується "поточний сеанс", а деякі браузери використовують відновлення сеансу при перезапуску. Це може призвести до того, що сесійні файли cookie зберігатимуться нескінченно довго. Наприклад:

```
Set-Cookie: id=abababababa; Expires=Thu, 1 Oct 2029 07:28:00 GMT;
```

Обмеження доступу до файлів cookie. Можна забезпечити безпечне надсилання файлів cookie та недопущення доступу до них сторонніх осіб або скриптів одним із двох способів: за допомогою атрибута Secure та атрибута HttpOnly.

Файл cookie з атрибутом Secure надсилається на сервер лише із зашифрованим запитом за протоколом HTTPS. Він ніколи не надсилається незахищеним HTTP (окрім локального хосту), що означає, що зловмисники не зможуть легко отримати до нього доступ. Незахищені сайти (з http: в URL-адресі) не можуть встановлювати файли cookie з атрибутом Secure. Однак

не думайте, що атрибут Secure запобігає будь-якому доступу до конфіденційної інформації, що міститься у файлах cookie. Наприклад, хтось, хто має доступ до жорсткого диска клієнта (або JavaScript, якщо атрибут HttpOnly не встановлено), може прочитати і змінити інформацію.

Файл cookie з атрибутом HttpOnly недоступний для JavaScript Document.cookie API; він надсилається лише на сервер. Наприклад, файли cookie, які зберігаються в сеансах на стороні сервера, не повинні бути доступними для JavaScript [15] і повинні мати атрибут HttpOnly. Цей запобіжний захід допомагає зменшити ризик міжсайтових скриптових атак (XSS).

```
Set-Cookie: id=a3fWa; Expires=Thu, 21 Oct 2021 07:28:00 GMT; Secure; HttpOnly
```

Визначте, куди надсилатимуться файли cookie. Атрибути Domain і Path визначають сферу застосування файлів cookie: на які URL-адреси слід надсилати файли cookie. Domain вказує, який сервер може отримувати файли cookie. Якщо він вказаний, то файли cookie доступні на сервері та його субдоменах. Наприклад, якщо ви встановите Domain=nulp.gov, файли cookie будуть доступні на nulp.gov і його субдоменах, таких як main.nulp.gov.

Якщо сервер не вказує домен, файли cookie доступні на сервері, але не на його субдоменах. Тому вказівка домену є менш обмежувальною, ніж його відсутність. Однак це може бути корисно, коли субдомени повинні обмінюватися інформацією про користувача.

Атрибут шляху. Атрибут Path вказує шлях до URL-адреси, який повинен існувати в запитуваній URL-адресі, щоб відправити заголовок Cookie. Символ %x2F ("/") вважається розділювачем каталогів, і підкаталоги також збігаються.

Наприклад, якщо ви задасте Path=/docs, ці шляхи запиту співпадуть:

- /docs
- /docs/
- /docs/Web/
- /docs/Web/HTTP

Але ці шляхи запитів не збігаються:

- /
- /docsets
- /fr/docs

Шлях за замовчуванням. Якщо атрибут Path не задано, його значення за замовчуванням обчислюється з шляху URI, який встановив файл cookie, наступним чином:

- якщо шлях порожній, не починається з “/” або містить не більше одного символу “/”, то значенням за замовчуванням для Path буде “/”;
- в іншому випадку, значенням за замовчуванням для шляху є шлях від початку до останнього символу “/”, але не включаючи його.

Наприклад, якщо файл cookie було встановлено з “https://example.org/a/b/c”, то значенням за замовчуванням для Path буде “/a/b”.

Атрибут SameSite. Атрибут SameSite дозволяє серверам вказувати, чи надсилати файли cookie з міжсайтовими запитами (де сайт визначається зареєстрованим доменом і схемою: http або https). Це забезпечує певний захист від атак підробки міжсайтових запитів (CSRF). Він приймає три можливих значення: Strict, Lax і None.

При виборі значення Strict браузер надсилає файл cookie лише із запитами з сайту, з якого він був створений. Значення Lax схоже, за винятком того, що браузер також відправляє файл cookie, коли користувач переходить на сайт походження файлу cookie (навіть якщо користувач переходить з іншого сайту). Наприклад, при переході за посиланням із зовнішнього сайту. Значення None означає, що файли cookie надсилаються як на сайт-джерело, так і на міжсайтовий запит, але тільки в безпечному контексті (тобто, якщо SameSite=None, то атрибут Secure також повинен бути встановлений). Якщо атрибут SameSite не встановлено, файл cookie розглядається як Lax. Наприклад:

```
Set-Cookie: key=value; SameSite=Strict
```

– SameSite=Lax – нове значення за замовчуванням, якщо SameSite не вказано. Раніше файли cookie відправлялися для всіх запитів за замовчуванням;

– файли cookie з SameSite=None тепер повинні також вказувати атрибут Secure (вони вимагають безпечного контексту);

– файли cookie з одного і того ж домену більше не вважаються такими, що походять з одного і того ж сайту, якщо вони відправлені за іншою схемою (http: або https:).

Префікси файлів cookie. Через конструкцію механізму файлів cookie сервер не може підтвердити, що файл cookie був встановлений з безпечного джерела, або навіть сказати, де саме він був встановлений. Вразлива програма на субдомені може встановити файл cookie з атрибутом домену, який надає доступ до цього файлу на всіх інших субдоменах. Цим механізмом можна зловживати в атаці з фіксацією сеансу. Однак, в якості захисту ви можете використовувати префікси файлів cookie, щоб стверджувати певні факти про файл cookie. Доступні два префікси:

– __Host-. Якщо ім'я файлу cookie має цей префікс, він приймається в заголовку Set-Cookie тільки в тому випадку, якщо він також позначений атрибутом Secure, був відправлений з безпечного джерела, не містить атрибуту Domain і має атрибут Path, встановлений в значення /. Таким чином, ці файли cookie можна розглядати як “заблоковані доменом”.

– __Secure-. Якщо ім'я файлу cookie має цей префікс, він приймається в заголовку Set-Cookie, тільки якщо він позначений атрибутом Secure і був відправлений з безпечного джерела. Це слабший захист, ніж префікс __Host-.

Браузер відхилятиме файли cookie з цими префіксами, які не відповідають його обмеженням. Зверніть увагу, що це гарантує, що файли cookie з префіксами, створені в субдомені, або обмежуються субдоменом, або повністю ігноруються. Оскільки сервер додатків перевіряє наявність певного імені файлу cookie лише при визначенні автентичності користувача або правильності токену CSRF, це ефективно діє як засіб захисту від фіксації сеансу.

Доступ до JavaScript за допомогою Document.cookie. Ви можете створювати нові файли cookie через JavaScript за допомогою властивості Document.cookie. Ви також можете отримати доступ до існуючих файлів cookie з JavaScript, якщо не встановлено прапорець HttpOnly.

```
document.cookie = "yummy_cookie=choco";
document.cookie = "tasty_cookie=strawberry";
console.log(document.cookie);
// logs "yummy_cookie=choco; tasty_cookie=strawberry"
```

Файли cookie, створені за допомогою JavaScript, не можуть включати прапорець HttpOnly. Файли cookie, доступні для JavaScript, можуть бути викрадені за допомогою XSS [13].

Способи захисту від атак з використанням файлів cookie

Використовуйте атрибут HttpOnly, щоб запобігти доступу до значень файлів cookie через JavaScript. Файли cookie, які використовуються для конфіденційної інформації (наприклад, для автентифікації), повинні мати короткий термін дії, а атрибут SameSite повинен мати значення Strict або Lax.) У браузерах, які підтримують SameSite, це гарантує, що файл cookie для автентифікації не надсилається з міжсайтовими запитами. Це зробить запит фактично неавтентифікованим для сервера додатків [11], [12].

Файл cookie пов'язаний з певним доменом і схемою (наприклад, http або https), а також може бути пов'язаний з піддоменами, якщо встановлено атрибут Set-Cookie Domain. Якщо домен і схема файлу cookie збігаються з поточною сторінкою, вважається, що файл cookie походить з того ж сайту, що і сторінка, і називається первинним файлом cookie.

Якщо домен і схема відрізняються, файл cookie не вважається таким, що походить з того ж сайту, і називається стороннім файлом cookie. Хоча сервер, на якому розміщена веб-сторінка, встановлює файли cookie першої сторони, сторінка може містити компоненти, що зберігаються на серверах в інших доменах, наприклад, зображення або інші документи, вбудовані в <iframe>. Ці компоненти можуть встановлювати сторонні файли cookie.

Сторонні файли cookie іноді називають міжсайтовими. Можливо, це більш точна назва, оскільки сторонні файли cookie передбачають, що вони належать сторонній компанії або організації. Однак поведінка і потенційні проблеми однакові, незалежно від того, чи володієте ви всіма залученими сайтами, чи ні.

Типові випадки використання сторонніх файлів cookie включають обмін інформацією про профіль користувача або збір аналітики в різних пов'язаних доменах. Вони також часто використовуються для реклами та відстеження користувачів в Інтернеті.

Сторонній сервер може створити профіль історії переглядів і звичок користувача на основі файлів cookie, надісланих йому тим самим браузером під час відвідування різних сайтів. Firefox [16] за замовчуванням блокує сторонні файли cookie, які, як відомо, містять трекери. Сторонні файли cookie також можуть бути заблоковані за допомогою інших налаштувань або розширень браузера. Блокування файлів cookie може призвести до того, що деякі сторонні компоненти (наприклад, віджети соціальних мереж) не працюватимуть належним чином.

Для розробників, які бажають поважати конфіденційність користувачів і мінімізувати відстеження третіми сторонами, є кілька корисних функцій:

– Сервери можуть встановлювати атрибут файлу cookie SameSite, щоб вказати, чи можна надсилати файли cookie третім особам.

– Cookies Having Independent Partitioned State (CHIPS) [14] дозволяє розробникам зберігати файли cookie в розділеному сховищі, з окремим кошиком для файлів cookie для кожного сайту верхнього рівня. Це дозволяє використовувати файли cookie третіх сторін без відстеження і продовжувати працювати в браузерах, які не дозволяють використовувати файли cookie для відстеження третіх сторін.

Автоматизація перевірки атрибутів cookie за допомогою програми на Python. Вибір мови Python [9] для написання скрипта сканування атрибутів файлів cookie веб-сайту обумовлений кількома ключовими факторами, які роблять Python особливо підходящим для такого завдання:

1. Простота і читабельність: Python відомий своїм чистим і читабельним синтаксисом, який робить код легким для розуміння та модифікації. Це особливо корисно для скриптів, які можуть потребувати подальшого розширення або налаштування іншими розробниками.

2. Широка підтримка бібліотек: Python має велику екосистему сторонніх бібліотек, які можуть спростити реалізацію багатьох завдань. Для роботи з HTTP-запитами і файлами cookie, наприклад, доступні бібліотеки `requests` і `http.cookies`, які значно спрощують роботу з мережевими запитами і обробкою файлів cookie.

3. Портативність: Python є кросплатформенною мовою, що означає, що скрипти, написані на Python, можуть виконуватися на різних операційних системах, включаючи Windows, macOS та Linux, без необхідності внесення змін до коду. Це робить Python ідеальним вибором для розробки інструментів, які потрібно використовувати в різних середовищах.

4. Спільнота та підтримка: Python має велику і активну спільноту розробників, що забезпечує велику кількість ресурсів для навчання та вирішення проблем.

5. Безпека та надійність: Python регулярно оновлюється і підтримується, що гарантує виправлення вразливостей та забезпечує надійність використання мови для розробки безпечних застосунків.

Загалом, Python є відмінним вибором для написання скриптів, які виконують сканування веб-сайтів, аналізують файли cookie та інші мережеві взаємодії завдяки його гнучкості, потужності та доступності.

Лістинг 1. Програма для автоматизації

```
import argparse
import requests
from http.cookies import SimpleCookie
from tabulate import tabulate

RED = '\033[91m' # Додаємо ANSI escape codes для кольорів

ENDC = '\033[0m'
CROSS = '\u274C'

# Функція для перевірки відсутніх атрибутів та форматування тексту
def check_missing_attributes(morsel):
    issues = []
    if 'secure' not in morsel:
        issues.append(f"{RED}{CROSS} Missing Secure Flag{ENDC}")
    if 'httponly' not in morsel:
        issues.append(f"{RED}{CROSS} Missing HttpOnly Flag{ENDC}")
    if 'samesite' not in morsel:
        issues.append(f"{RED}{CROSS} Missing SameSite Flag{ENDC}")
    return "\n".join(issues) if issues else "None"

def analyze_cookies(url): # Функція для аналізу файлів cookie
    response = requests.get(url)
    cookies = response.cookies
    cookies_data = []

    for cookie in cookies:
        cookie_value = cookies.get(cookie)
        parsed_cookie = SimpleCookie(f"{cookie}={cookie_value}")
        morsel = parsed_cookie[cookie]

        missing_attributes = check_missing_attributes(morsel)

        cookies_data.append([cookie, morsel.value, missing_attributes])

    return cookies_data

# Функція для налаштування аргументів командного рядка
def setup_arg_parser():
    parser = argparse.ArgumentParser(description="Cookie Scanner with Identified Issues")
    parser.add_argument("--url", type=str, required=True, help="URL of the website to scan for cookies")
    return parser

def main(): # Головна функція
    parser = setup_arg_parser()
    args = parser.parse_args()
    url = args.url

    cookies_data = analyze_cookies(url)
    if cookies_data:
        print(f"Cookies for {url}:")
        print(tabulate(cookies_data, headers=["Cookie Name", "Value", "Identified Issues"], tablefmt="grid"))
    else:
        print(f"No cookies found for {url}")

if __name__ == "__main__":
    main()
```

Рисунок 2 – Приклад роботи скрипту

На рисунку 2 показано механізм роботи скрипта. Він перевіряє наявність атрибутів у сесійних файлах.

Вразливості та атаки, пов'язані з сесійними файлами cookies без атрибутів Secure, HttpOnly та SameSite, включають:

– *Відсутність атрибуту Secure*: Викрадення сесії через перехоплення: Без атрибуту Secure, cookies можуть передаватися в незахищеному вигляді через HTTP, що робить їх вразливими до перехоплення через атаки "man-in-the-middle" (MitM), особливо в незахищених Wi-Fi мережах.

– *Відсутність атрибуту HttpOnly*: Cross-Site Scripting (XSS): Якщо cookie не встановлено як HttpOnly, зловмисні скрипти, виконані через XSS-атаки, можуть отримати доступ до цих cookies, що дозволяє зловмисникам викрасти сесійні cookies та отримати несанкціонований доступ до облікового запису користувача.

– *Відсутність атрибуту SameSite*: Cross-Site Request Forgery (CSRF): Без атрибуту SameSite, cookies можуть бути автоматично відправлені з запитами, ініційованими з інших сайтів. Це може призвести до атак CSRF, де зловмисник може виконати небажані дії від імені жертви на вразливому сайті.

– *Cross-Site Script Inclusion (XSSI)*: Атаки, при яких зловмисники можуть включати сесійні cookies користувача в запити до інших сайтів, потенційно розкриваючи інформацію про сесію або інші чутливі дані.

Для захисту від цих та інших вразливостей, рекомендується:

1. Встановити атрибут Secure для cookies, щоб гарантувати, що вони передаються тільки через захищені HTTPS-з'єднання.
2. Використовувати атрибут HttpOnly, щоб запобігти доступу до cookies через скрипти на стороні клієнта, зменшуючи ризик викрадення сесій через XSS.
3. Налаштувати атрибут SameSite для cookies, щоб обмежити їх відправку з міжсайтовими запитами, захищаючи від атак CSRF та інших атак, які використовують cookies.

Обговорення результатів дослідження. У роботі проведено аналіз сучасних підходів до захисту сесій та cookie, зокрема, вивчення атрибутів Secure, SameSite та HttpOnly та механізмів захисту від CSRF. Мета полягала у вдосконаленні існуючих методів безпеки. Особлива увага була приділена розробленню автоматизованої системи, яка дозволяє перевіряти наявність необхідних атрибутів безпеки в cookies: Secure, SameSite та HttpOnly. Зазначена система пропонує сканувати HTTP-відповіді сервера з метою перевірки наявності та правильності встановлення згаданих атрибутів. Ці атрибути є критично важливими для забезпечення конфіденційності та цілісності сесій користувачів, і їх належне використання має велике значення для загальної безпеки веб-додатків.

Висновки та перспективи подальших досліджень. У цьому дослідженні ми встановили нові межі застосування сесійних cookie для забезпечення безпеки веб-додатків. Наші висновки підкреслюють важливість розвитку та імплементації більш надійних методів автентифікації та авторизації користувачів. Однак, існують істотні обмеження, пов'язані з сумісністю та впровадженням стандартів безпеки.

Перспективи подальших досліджень включають аналіз впливу новітніх технологій, таких як машинне навчання та штучний інтелект [10], на методи захисту веб-додатків. Також потрібно більше роботи для визначення впливу змін веб-протоколів на безпеку cookie.

Враховуючи швидкий розвиток веб-технологій, необхідно створити адаптивні системи, які можуть прогнозувати та запобігати новим видам атак в реальному часі. Розширюючи концепцію автоматичної перевірки безпеки cookies, ми можемо створити сканер вразливостей для веб-додатків, який буде інтегрований у комплексну систему перевірки безпеки [7]. Цей сканер буде використовувати чекліст Open Web Application Security Project (OWASP) [8] як основу для оцінки безпеки веб-додатків. Це дозволить розробникам і аудиторам безпеки систематично виявляти вразливості, що спираються на найбільш актуальні і повні знання про можливі загрози.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] J.S. Park, and R. Sandhu, “Secure cookies on the Web”, *IEEE Internet Comput.*, vol. 4, no. 4, pp. 36-44, 2000. Accessed: Feb. 26, 2024, doi: https://doi.org/10.1109/4236.865085.

[2] H. Kwon, H. Nam, S. Lee, C. Hahn, and J. Hur, “(In-)Security of cookies in HTTPS: Cookie theft by removing cookie flags”, *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 1204-1215, 2019, doi: https://doi.org/10.1109/tifs.2019.2938416.

[3] V. Khu-smith, and C.J. Mitchell, “Enhancing the security of cookies”, in *Proc. 4th Int. Conf. Secur. Cryptol. – ICISC 2001*, Seoul, Korea, 2001, doi: https://doi.org/10.1007/3-540-45861-1_11.

[4] Ю. Толкачова, О. Гарасимчук, та І. Опірський, “Аналіз безпеки протоколу oauth”, *Вісник Львівського державного університету безпеки життєдіяльності*, т. 27, с. 67-76, 2023, doi: https://doi.org/10.32447/20784643.27.2023.08.

[5] “Cookies: Механізм управління станом HTTP”, *IETF Datatracker*. [Електронний ресурс]. Доступно: https://tools.ietf.org/html/draft-ietf-httpbis-rfc6265bis-03. Дата звернення: Лют. 26, 2024.

[6] Y. Lakh, E. Nyemkova, A. Piskozub, and V. Yanishevskyi, “Investigation of the broken authentication vulnerability in web applications”, in *Proc. 11th IEEE Int. Conf. Intell. Data Acquisition Adv. Comput. Syst: Technol. Appl.* (*IDAACS*), 2021, doi: https://doi.org/10.1109/IDAACS53288.2021.9660889.

[7] Ya. Stefinko, A. Piskozub, and R. Banakh, “Manual and automated penetration testing. Benefits and drawbacks. Modern tendency”, in *Proc. 13th International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science* (*TCSET*), 2016, doi: https://doi.org/10.1109/TCSET.2016.7452095.

[8] “OWASP Top Ten”, *OWASP Foundation*. [Online]. Available: https://owasp.org/www-project-top-ten. Accessed on: Feb. 26, 2024.

[9] “Ласкаво просимо на python.org”, *Python.org*. [Електронний ресурс]. Доступно: https://www.python.org. Дата звернення: Лют. 26, 2024.

[10] А. Піскозуб, Д. Журавчак, та А. Толкачова, “Дослідження вразливостей у чатботах з використанням великих мовних моделей”, *Безпека інформації*, т. 29, №. 3, с. 111-117, 2023, doi: https://doi.org/10.18372/2225-5036.29.18069.

[11] J.M. Gomes, “Secure, HttpOnly, SameSite HTTP Cookies Attributes and Set-Cookie Explained”, *Medium*. [Online]. Available: https://medium.com/swlh/secure-httponly-samesite-http-cookies-attributes-and-set-cookie-explained-fc3c753dfeb6. Accessed on: Feb. 26, 2024.

[12] “Using Burp to Hack Cookies and Manipulate Sessions”, *PortSwigger*. [Online]. Available: https://portswigger.net/support/using-burp-to-hack-cookies-and-manipulate-sessions. Accessed on: Mar. 12, 2024.

[13] “Cross-site scripting”, *PortSwigger*. [Online]. Available: https://portswigger.net/web-security/cross-site-scripting. Accessed on: Mar. 13, 2024.
[14] “Cookies Having Independent Partitioned State (CHIPS)”, *Google for developers*. [Online]. Available: https://developers.google.com/privacy-sandbox/3pcd/chips. Accessed on: Mar. 15, 2024.
[15] “JavaScript”, *MDN Web Docs*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript. Accessed on: Mar. 10, 2024.
[16] “Оберіть і завантажте Браузер Firefox вашою мовою”, *Mozilla*. [Електронний ресурс]. Доступно: https://www.mozilla.org/uk/firefox/all. Дата звернення: Лют. 26, 2024.

Стаття надійшла до редакції 26.04.2024.

REFERENCE

[1] J.S. Park, and R. Sandhu, “Secure cookies on the Web”, *IEEE Internet Comput.*, vol. 4, no. 4, pp. 36-44, 2000. Accessed: Feb. 26, 2024, doi: https://doi.org/10.1109/4236.865085.
[2] H. Kwon, H. Nam, S. Lee, C. Hahn, and J. Hur, “(In-)Security of cookies in HTTPS: Cookie theft by removing cookie flags”, *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 1204-1215, 2019, doi: https://doi.org/10.1109/tifs.2019.2938416.
[3] V. Khu-smith, and C.J. Mitchell, “Enhancing the security of cookies”, in *Proc. 4th Int. Conf. Secur. Cryptol. – ICISC 2001*, Seoul, Korea, 2001, doi: https://doi.org/10.1007/3-540-45861-1_11.
[4] Y. Tolkacheva, O. I. Garasymchuk, and I. R. Opirsky, “Security analysis of the oauth protocol”, *Bulletin of Lviv State University of Life Safety*, vol. 27, pp. 67-76, June. 2023. [Online]. Available: https://doi.org/10.32447/20784643.27.2023.08. Date of access: Feb. 26, 2024.
[5] “Cookies: An HTTP state management mechanism”, *IETF Datatracker*. [Online]. Available: https://tools.ietf.org/html/draft-ietf-httpbis-rfc6265bis-03. Accessed: Feb. 26, 2024.
[6] Y. Lakh, E. Nyemkova, A. Piskozub, and V. Yanishevskyi, “Investigation of the broken authentication vulnerability in web applications”, in *Proc. 11th IEEE Int. Conf. Intell. Data Acquisition Adv. Comput. Syst: Technol. Appl.* (*IDAACS*), 2021, doi: https://doi.org/10.1109/IDAACS53288.2021.9660889.
[7] Ya. Stefinko, A. Piskozub, and R. Banakh, “Manual and automated penetration testing. Benefits and drawbacks. Modern tendency”, in *Proc. 13th International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science* (*TCSET*), 2016, doi: https://doi.org/10.1109/TCSET.2016.7452095.
[8] “OWASP Top Ten”, *OWASP Foundation*. [Online]. Available: https://owasp.org/www-project-top-ten. Accessed on: Feb. 26, 2024.
[9] “Welcome to python.org”. Python.org. [Online]. Available: https://www.python.org/. Accessed: 26 Feb. 2024.
[10] A. Piskozub, D. Zhuravchak, and A. Tolkacheva, “Researching vulnerabilities in chatbots with LLM (Large Language Model)”, *Ukr. Scient. Jour. Inf. Secur.*, vol. 29, iss. 3, pp. 111-117, 2023, doi: https://doi.org/10.18372/2225-5036.29.18069.
[11] J.M. Gomes, “Secure, HttpOnly, SameSite HTTP Cookies Attributes and Set-Cookie Explained”, *Medium*. [Online]. Available: https://medium.com/swlh/secure-httponly-samesite-http-cookies-attributes-and-set-cookie-explained-fc3c753dfeb6. Accessed on: Feb. 26, 2024.
[12] “Using Burp to Hack Cookies and Manipulate Sessions”, *PortSwigger*. [Online]. Available: https://portswigger.net/support/using-burp-to-hack-cookies-and-manipulate-sessions. Accessed on: Mar. 12, 2024.
[13] “Cross-site scripting”, *PortSwigger*. [Online]. Available: https://portswigger.net/web-security/cross-site-scripting. Accessed on: Mar. 13, 2024.

[14] “Cookies Having Independent Partitioned State (CHIPS)”, *Google for developers*. [Online]. Available: https://developers.google.com/privacy-sandbox/3pcd/chips. Accessed on: Mar. 15, 2024.
[15] “JavaScript”, *MDN Web Docs*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript. Accessed on: Mar. 10, 2024.
[16] “Choose and download Firefox Browser in your language”, *Mozilla*. [Online]. Available: https://www.mozilla.org/uk/firefox/all/. Accessed: 26 Feb. 2024.

ANASTASIIA TOLKACHOVA,
DANYIL ZHURAVCHAK

AUTOMATE THE VERIFICATION OF SESSION COOKIE ATTRIBUTES

In this research, we focus on a critical web security topic, namely the security of session cookies, which play a key role in the functioning of modern web applications. As a standard mechanism for storing data on the client side, cookies are crucial for authentication, authorization and maintaining the state of a user's session. However, despite their necessity and convenience, cookies can also pose serious security risks. Our research focuses on the analysis and automation of cookie attribute verification, which is critical to ensuring protection against various web attacks. Identifying and eliminating weaknesses in cookie attributes can significantly reduce the risk of malicious attacks such as session hijacking, cross-site scripting (XSS) and cross-site request forgery (CSRF) attacks. We take an in-depth look at modern methods and tools for securing cookies, including implementing strict policies on cookie attributes such as Secure, HttpOnly, and SameSite. These attributes help to restrict access to cookies from unauthorized use via client-side scripts and provide additional protection against cross-site attacks. In addition, we consider the importance of updating the cookie standard, RFC6265bis, which offers improved security mechanisms, including the SameSite attribute, which allows controlling the sending of cookies during cross-requests, thereby reducing the risk of CSRF attacks. Our research also includes an analysis of potential threats and vulnerabilities associated with the misuse or misconfiguration of cookies, as well as a discussion of strategies to minimize these risks. We demonstrate how detailed automated verification of cookie attributes can significantly improve the security of web applications. The results of the study point to the need to constantly monitor and evaluate the protection of session cookies, as well as the importance of implementing security best practices and standards to ensure the reliability and security of web applications.

Keywords: cookies, vulnerabilities, CSRF, web application, security, session.

Толкачова Анастасія Юріївна, магістр, кафедри захисту інформації Національного університету “Львівська політехніка”, Львів, Україна, ORCID 0000-0002-8196-7963, tolkachova.nastia@gmail.com.

Журавчак Даниїл Юрійович, аспірант, асистент кафедри захисту інформації Національного університету “Львівська політехніка”, Львів, Україна, ORCID 0000-0003-4989-0203, danyil.y.zhuravchak@lpnu.ua.

Tolkachova Anastasiia, master's degree student, information security academic department, Lviv Polytechnic National University, Lviv, Ukraine.

Danyil Zhuravchak, post graduate student, assistant of the information security academic department, Lviv Polytechnic National University, Lviv, Ukraine.