УДК 004.056.53

Informatics and Mathematical Methods in Simulation
Vol. 2 (2012), No. 1, pp. 87-93

ЗАХИСТ ВЕБ-СЦЕНАРІЇВ ВІД НЕСАНКЦІОНОВАНОГО КОПІЮВАННЯ І МОДИФІКАЦІЇ

Л.М. Тимошенко[1], **Ю.М. Чайківська**[2]

[1] Одеський національний політехнічний університет,
просп. Шевченко, 1, Одесса, 65044, Украина; e-mail: lmt0902@gmail.com
[2] Тернопільський національний педагогічний університет ім. В. Гнатюка,
вул. М.Кривоноса, 2, Тернопіль, 46027, Україна

Розглядаються системи захисту веб-сценаріїв, що дозволяють виявити зловмисні коди і захистити веб-сайти від копіювання інформації шляхом фільтрації даних. Побудовані інструменти з використанням механізму BEEP в браузерах дозволяють спростити застосування політики безпеки у веб-сценаріях, причому зміна мережевих сервісів вимагає мінімальних зусиль.

Ключові слова: політика безпеки, веб-сценарій, механізм BEEP, функція відслідковування

Постановка проблеми

Система захисту повинна бути багаторівневою і будуватися відповідно до необхідного рівня стійкості системи в цілому. Для надійного захисту необхідно розподілити функції між цими рівнями так, щоб здійснювалося дублювання функцій захисту і відбувалася компенсація недоліків одного рівня іншим. Криптографічний захист потрібно забезпечувати на всіх верхніх рівнях, проте його застосування має відповідати передбачуваним загрозам. Оскільки загальні витрати на захист великі, то в першу чергу потрібно підсилити його слабкі елементи.

Аналіз слабких місць при скануванні на мережевому рівні допоможе виявити серйозні недоліки, зокрема неправильно сконфігуровані міжмережеві екрани або вразливості веб-сервера, які можуть спровокувати потенційну загрозу і дозволити зловмисникам проникнути в систему захисту. Сканування на мережевому рівні забезпечує швидкий і детальний аналіз мережевої інфраструктури організації як ззовні, так і зсередини.

Аналіз останніх досліджень і публікацій

Найбільш небезпечними є вразливості проектування, які виявляються та усуваються з великими труднощами. Вразливість притаманна алгоритму, тому навіть досконала його реалізація не дозволяє уникнути закладеної загрози. Імітація атак не завжди можна реалізувати. Такі випадки можна розділити на дві категорії: ситуації, в яких програма призводить до відмови в обслуговуванні вузла або мережі, і ситуації, в яких не може проводитись атака в мережі. В цьому випадку можна скористатися системою аналізу захищеності на рівні операційної системи, агенти якої встановлюються на кожен контрольований вузол і проводять всі перевірки локально. Для вирішення цієї проблеми деякі компанії-виробники пропонують користувачам декілька систем аналізу захищеності, що працюють на всіх вказаних вище рівнях − мережевому, системному і прикладному. Сукупність цих систем дозволяє з високим ступенем ефективності

визначити майже всі відомі вразливості.

Зокрема, компанія Internet Security Systems пропонує сімейство *SAFEsuite*, що складається з чотирьох сканерів: *Internet Scanner*, *System Scanner*, *Security Manager* і *Database Scanner* [1]. На сьогоднішній день це єдина компанія, яка пропонує системи аналізу захищеності, що функціонують на всіх трьох рівнях інформаційних інфраструктур. Інші компанії пропонують два або один сканер (*Network Associates*, *NetSonar* та ін.) [2].

Компанія Cisco пропонує тільки систему аналізу захищеності на рівні мережі і поділяє всі вразливості на два класи: потенційні, що випливають з перевірок заголовків і так званих активних «підштовхувань» (nudge) сервісу або аналізованого вузла. Потенційна вразливість може існувати в системі, проте активні перевірки зондування не підтвердять її; підтверджені − виявлені та існуючі на аналізованому хості.

Відмінність між системами *CyberCop Scanner* і *Internet Scanner* полягає в тому, що розробники з NAI не додають в свою систему перевірку, якщо не впевнені у її ефективності. В той же час розробники ISS це роблять навіть тоді, коли перевірка знаходить вразливість з певною точністю. Після випуску системи відбувається повернення до розроблених перевірок, їх покращення, додавання нових механізмів здійснення перевірок для підвищення достовірності. Не всі перевірки, розроблені в лабораторіях, функціонують належним чином. На це можуть впливати такі фактори:

- особливості конфігурації системи користувача;
- метод, яким було скомпільовано домен або сервіс;
- помилки віддаленої системи та інше.

В таких випадках автоматична перевірка може пропустити вразливість, яка легко визначається вручну і може бути широко розповсюджена у багатьох системах.

Мета дослідження

Мережевий сканер повинен бути першим інструментом, що використовується у процесі аналізу захищеності сценаріїв і повинен забезпечувати швидкий огляд загроз, які потребують уваги. Для забезпечення захисту від модифікацій веб-сценаріїв необхідно запропонувати механізм вбудованої політики безпеки в браузері.

Основна частина

Мережа складається із каналів зв’язку, вузлів, серверів, робочих станцій, прикладного і системного програмного забезпечення, баз даних і т.д. Всі ці компоненти потребують оцінки ефективності їхнього захисту. Засоби аналізу захищеності аналізують мережу і здійснюють пошук «слабких» місць, обробляють отримані результати і створюють різного роду звіти. До проблем, що ідентифікуються системами аналізу захищеності, належать:

- «люки» в програмах (*back-door*) і програми типу «троянський кінь»;
- «слабкі» паролі;
- чутливість до проникнення з незахищених систем;
- неправильне налаштування мережевих екранів, веб-серверів і баз даних та ін.

Технологія аналізу захищеності є дієвим методом реалізації політики мережевої безпеки проти порушення ззовні або зсередини мережі.

Веб-вузли, які дозволяють переглядати сторінки, зазвичай фільтрують інформацію для попередження проникнення коду сценарію в запущені браузери з різними алгоритмами, що ускладнює процес фільтрування і захисту від атак. Можна запропонувати альтернативний механізм для уникнення внесення сценарію в браузер − це вбудована політика безпеки в браузері. Ідея полягає в тому, що у веб-сайт вбудовується система

захисту, що дозволяє визначити, які сценарії запущені і коли. В такому випадку браузер зможе керувати сценарієм.

Багато веб-сайтів перевидають контент, що змінюється групами користувачів або іншим чином, наприклад, рекламною мережею і пошуковими системами. Якщо перевиданий контент містить вбудовані сценарії, відвідувачі сайту можуть бути не захищені від атак типу «перехресний сценарій сайту» (cross-site scripting (XSS)) [3]. Це дозволить сайту виступити в ролі учасника атаки на інші сайти. Стандартним захистом для веб-сайтів є фільтр або модифікація контента, видалення вбудованих сценаріїв або інших потенційно-небезпечних елементів [4].

На практиці реалізація фільтрування є складною задачею. Представлення інформації на сайтах потребує збільшення контенту із зображеннями, гіперпосиланнями, стилями типографічного оформлення і т.д. У сценарій може бути вкладений контент різними шляхами і можливо блокувати скрипти без виведення з ладу контентів. Однією із причин несанкціонованого копіювання є те, що різні структурний аналіз браузерів і візуалізація контенту: для одного фільтрування може бути ефективним, а для іншого ні. Окрім того, аналіз і спотворення відображення контенту використовують сучасні атаки, що обходять фільтри сервера (наприклад, черв'як Сема і Яманера) [5]. Запропонований метод для запобігання ін'єкції зловмисного скрипта базується на двох аспектах:

1) браузери формують механізм виявлення сценарію; якщо браузер не аналізує контент як скрипт в той час, як відображає веб-сторінку, то він не буде виконуватися;

2) розробник веб-програм повинен знати, які сценарії потрібно виконувати для того, щоб програма була функціональною.

Перший аспект ґрунтується на тому, що браузер − це найкращий варіант для фільтрації сценаріїв. Насправді більшість веб-сервісів (наприклад, GPokr, S3AjaxWiki) логічно виконуються в браузері з активними веб-сайтами як накопичення даних. Фільтрування з боку браузера може бути єдиним варіантом для цих сервісів.

Другий аспект ґрунтується на тому, що веб-сайт повинен забезпечувати політику фільтрації для браузера. Тобто він може конкретизувати, який сценарій прийняти до виконання, а який відмінити. Таким чином, веб-сайт встановлює політику безпеки і браузер надає інструкції для її виконання. Такий метод називається вбудованою політикою безпеки в браузері – *Browser-Enforced Embedded Policies* (BEEP).

Розглянутий метод BEEP дозволяє управляти генеральною політикою безпеки. Захист політики реалізований за допомогою довіреної функції JavaScript, що дозволяє вбудувати перегляд веб-сторінки. Цю функцію називають «гаком» (відслідковуванням). Відповідно модифікований браузер передає кожен сценарій, що виявляється за допомогою «гака» при структурному аналізі (разом з іншою суттєвою інформацією), і сценарій виконується у випадку, якщо функція «гака» надасть дозвіл.

В результаті дослідження була додана підтримка BEEP до кількох браузерів і побудовані інструменти, що дозволяють спростити застосування політики безпеки у веб-сценарії. Було виявлено, що підтримка BEEP в браузерах вимагає невеликих і локалізованих модифікацій, а зміна мережевих сервісів вимагає мінімальних зусиль.

Розглянемо переваги методу BEEP. По-перше, це гнучка політика безпеки. «Гаком» безпеки може бути будь-яка функція, що реалізується за допомогою JavaScript. Можна розглянути два види політик безпеки. Перший − рекомендований список, в якому функція «гака» включає в собі односторонню хеш-функцію кожного легітимного сценарію, розташованого на сторінці. При виявленні небезпечного сценарію в браузері він передається функції «гака», яка здійснює хешування сценарію і порівнює його з рекомендованим списком. У випадку, коли хешований сценарій не знайдено у списку, він відкидається. Другий тип політики безпеки полягає у використанні механізму захисту об'єктної моделі документів (DOM), що забезпечує стандартний інтерфейс для доступу і керування HTML-об'єктами. В даному випадку веб-сервіси структурують сторінки для ідентифікації контенту, що може містити зловмисні сценарії. Можливий

зловмисний контент користувача розміщений в або елементах, що виконують роль механізму захисту аплетів (*sandbox*). Наприклад: . . .можливий зловмисний контент. . . .

В межах *sandbox* дозволено використовувати великий контент, що може містити типографічні стилі, графічні зображення та ін., проте всі ці сценарії блоковані до виклику функції «гака». При виклику функції «гака» здійснюється тестування документа в проаналізованому відображенні дерева DOM. Починаючи з вузла сценарію DOM-функція «гака» ретельно досліджує всі вузли аж до кореня дерева, шукаючи «не виконувані» (noexecute) вузли. Якщо такий вузол знайдено, сценарій не виконуватиметься.

Поряд з механізмами безпеки, що включать в собі рекомендований список і DOM-аплет, BEEP дозоляє виявити і відфільтрувати всі ін'єктовані сценарії. По-перше, перевірені сценарії ідентифікуються безпосередньо веб-сторінкою, по-друге, браузер запускає функцію «гака» перед виконанням будь-якого сценарію. В цьому можна легко переконатися: визначення функції гака як першого сценарію в заголовку документа гарантує те, що він буде аналізуватися першим. Разом ці умови означають, що будь-який не схвалений сценарій буде відкинуто перед його запуском.

Описаний метод потребує модифікацій браузера. Наприклад, в місцях, де відбувається модифікація браузера, потрібно встановити в початковому коді виклик інтерпретатора JavaScript. В цих точках виклику браузер ідентифікує сценарій веб-сторінки. Для того, щоб застосувати функцію «гака», потрібно спочатку вставити коди в програму і викликати інтерпретатор JavaScript.

В залежності від результату виконання першої функції програми потрібно запустити сценарій із сторінки або пропустити його. Для проведення дослідження були модифіковані Konqueror і Safari браузери для підтримки функції «гака» і реалізовано часткову підтримку закритого коду веб-браузера Opera. Ці зміни вимагають близько 650 стрічок коду в першому випадку і близько 100 стрічок для Opera.

Атаки, що базуються на використанні зловмисного сценарію, написаного на JavaScript, вбудовують його код в контент довіреного веб-вузла. При перегляді користувачем сторінки сайту вбудований сценарій завантажується і виконується в браузері користувача з привілеями довіреного сайту. Ін'єктований сценарій може видавати привілейовану інформацію (cookies, історія браузера, приватна інформація з сайту) [6]. Сценарій може використовувати браузер користувача для реалізації атаки відмови служб на веб-вузлі і т.п.

Проникнення сценарію може здійснюватися кількома шляхами. В перехресному сценарії сайту (XSS) атакуючий використовує веб-сайти, вносячи підготовлений користувачем текст в сторінки без фільтрування тексту. Наприклад, члени онлайн-товариства типу MySpace, Blogger, і Flickr можуть увійти у власний контент і додавати коментарі в інші контенти [6]. Цей контент зберігається на сайті і будь-хто може його переглянути. Якщо зловмиснику вдалося включити свій скрипт в цей контент, то будь-які читачі цього контенту будуть запускати скрипт з привілеями сайту. Якщо читач є членом сайту, скрипт може мати доступ до модифікації читацького контенту, включаючи приватну інформацію, що зберігається на сайті або в браузері (рис. 1).

Інший метод проникнення сценарію − «віддзеркалення». Наприклад, при запиті неіснуючої сторінки багато сайтів намагаються представити корисну відповідь, що включає URL неіснуючої сторінки. Тому, якщо сайт не захищений, програма <script>...</script> в URL виконуватиметься в браузері користувача, коли він отримає повідомлення «сторінку не знайдено». Таким чином зловмисник намагається заманити користувача на URL з вбудованими сценаріями. Наприклад: http://trusted.site/<script>document.location='http://evil.site/?'+document.cookie</script>.

Зловмисник може розмістити URL як спам в електронній пошті, а також в коментарях або рекламі на trusted.site чи іншому сайті. Якщо жертва завантажує посилання, сценарій виконуватиметься на незнайденій сторінці, що обслуговується

`trusted.site`, вилучаючи cookie користувача і надсилаючи його на `evil.site`.

Інший можливий сценарій атаки використовує динамічну природу JavaScript санкціонованого доступу до веб-сторінок, де HTML-контент обслуговується веб-сервером і змінюється в браузері через виконання сценаріїв. Наприклад, сайт можна побудувати таким чином, що URL типу `http://vulnerable.site/welcome.html?name=Joe` буде видавати персоналізований контент, використовуючи статичну HTML сторінку в поєднанні з вбудованим сценарієм. Зокрема, сценарій може використовувати особливості типу innerHTML і document.write для модифікації контенту сторінки, що відображається в браузері, таким чином персоналізуючи значення «name». Це дає можливість зловмисному сценарію проникнути в браузер за допомогою комбінації «name».

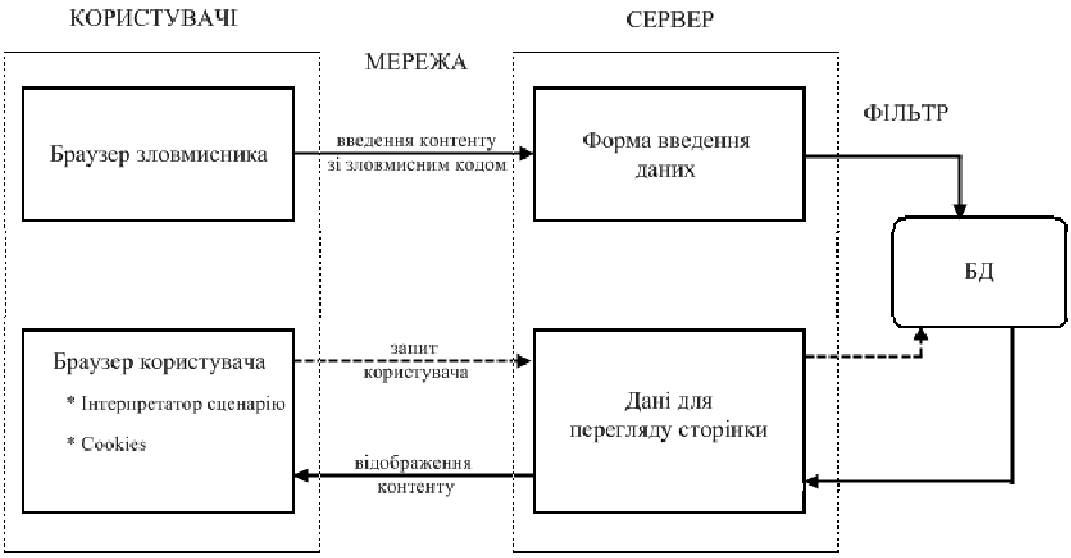

Рис. 1. Схема виявлення ін'єктованих сценаріїв

Стандартним вирішенням проблеми проникнення сценарію є встановлення фільтру на веб-вузлі або перетворення всіх можливих зловмисних контентів в сценарії для видалення або блокування.

Для виявлення вбудованих сценаріїв можна запропонувати альтернативний підхід, ідея якого полягає в конкретизації кожної сторінки веб-вузла, політика безпеки якого дозволяє або забороняє виконання сценарію. В даному випадку веб-вузол конкретизує політику через «гак» безпеки, що використовується для перевірки сценарію перед його виконанням у браузері. Функція гака реалізує ефективний захист - будь-який сценарій підлягає детальному аналізу перед його виконанням. Це здійснюється наступним чином: функція інсталюється перед аналізом зловмисного коду і його виконанням. Стандарт HTML не конкретизує порядок аналізу і виконання. На практиці перевірено, що в основному браузери аналізують і виконують перший елемент `<script>` в заголовку. Поведінка браузера залежить від визначення функції «гака» в першому сценарії елемента `<head>` в документі. Таким чином, жодний динамічний контент не включається доти, поки не виконається функція «гака» в `<head>` кожної веб-сторінки. Варто зазначити, що введення функції захисту відразу захищає від невмілого користування, бо будь-які зловмисні коди, спрямовані на зміну функції, будуть проаналізовані після встановлення. Таким чином здійснюється фільтрування «гаком» і попередження запуску програм.

Для перевірки політики безпеки від модифікації сценарію був використаний тест, що містить комплект 61 векторних атак XSS [7]. Тест перевіряє частини коду HTML і JavaScript, що можуть бути вбудовані в контент користувача. Нами було обрано 26 сайтів, які були локально збережені, щоб не вносились зміни з боку мережі. В подальшому використовувався сценарій ідентифікації для обчислення функції SHA-1 кожного сценарію в межах сторінки. Кожна сторінка 26 сайтів була завантажена в браузері на протязі 20 годин і вимірювався загальний час завантаження кожної сторінки: для не модифікованих 520 сторінок час складав 698.2 секунди, а для сторінок, що містять вбудовану політику безпеки, − 798.6 секунд. Отже, в середньому час завантаження збільшився на 14.4%.

Висновки

Запропонований механізм підтримки політики безпеки BEEP застосовано до кількох браузерів, що дозволило посилити захист у веб-сценаріях. Він забезпечує захист веб-сценарію від модифікації, використовуючи функцію «гака», що включає в собі односторонню хеш-функцію кожного легітимного сценарію, розташованого на сторінці.

Варто зазначити, що цих механізмів вистачає для виявлення зловмисного сценарію, проте можна додавати й інші механізми. Наприклад, функція «гака» може використовуватися для перевірки веб-сайту і видачі повідомлення про знаходження зловмисного сценарію. Проте механізми політики безпеки можуть змінюватися через певний час: нова політика безпеки дописується до сторінки сайту і буде призначатися браузером.

Список літератури

1. Shriram Krishnamurthi. The Continue server (or, How I Administered PADL 2002 and 2003) // Lecture Notes in Computer Science. – 2003. – Vol.2562. – PP. 2-16.
2. Antonatos S. Puppetnets: Misusing Web Browsers as a Distributed Attack Infrastructure / S. Antonatos, P. Akritidis, V.T. Lam, K.G. Anagnostakis // ACM Transactions on Information and System Security. – 2008. – Vol.12, Iss.2. – PP. 1-38.
3. Klein A. DOM Based Cross Site Scripting or XSS of the Third Kind *[Електронний ресурс]* // Web Application Security Consortium. – Режим доступу: http://www.webappsec.org/projects/articles/071105.shtml.
4. Bowman M.C. et al. The Harvest Information Discovery and Access System // Computer Networks and ISDN Systems. – 1995. – Vol.28, Iss.1-2. – PP. 119-125.
5. Castillo C. Cooperation Schemes between a Web Server and a Web Search Engine / C. Castillo // LA-WEB'03 Proceedings of the First Conference on Latin American Web Congress. – USA: IEEE Computer Society, 2003. – PP. 212-213.
6. Srinivasan P., Pant G., Menczer F. Target Seeking Crawlers and their Topical Performance *[Електронний ресурс]* / P. Srinivasan, G. Pant, F. Menczer. – Режим доступу: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.23.6092.
7. RSnake. XSS (Cross Site Scripting) Cheat Sheet Esp: for filter evasion. http://ha.ckers.org/xss.html.

ЗАЩИТА ВЕБ-СЦЕНАРИЕВ ОТ НЕСАНКЦИОНИРОВАННОГО КОПИРОВАНИЯ И МОДИФИКАЦИИ

Л.Н. Тимошенко[1], Ю.М. Чайковская[2]

[1] Одеський національний політехнічний університет,
просп. Шевченка, 1, Одеса, 65044, Україна; e-mail: lmt0902@gmail.com
[2] Тернопольский национальный педагогический университет имени В. Гнатюка,
ул. М. Кривоноса, 2, Тернополь, 46027, Украина

Рассмотрены системы защиты веб-сценариев, которые дают возможность обнаружить злоумышленные коды и защитить веб-сайты от копирования информации путем фильтрации данных. Разработанные инструменты с использованием механизма BEEP в браузерах позволяют упростить применение политики безопасности в веб-сценариях, причем смена сетевых сервисов требует минимальных усилий.
Ключевые слова: политика безопасности, веб-сценарий, механизм BEEP, функция отслеживания

PROTECTION OF WEB SCRIPTS FROM UNAUTHORIZED COPYING AND MODIFICATION

Lidiya M. Tymoshenko[1], Julia M. Chaikivska[2]

[1] Odessa National Polytechnic University,
1 Shevchenko Ave., Odessa, 65044, Ukraine; e-mail: lmt0902@gmail.com
[2] Ternopil V. Hnatyuk National Pedagogical University,
2 M. Kryvonosa str., Ternopil, 46027, Ukraine

We consider the protection systems of web scripts that allow detecting malicious code and making possible protection websites from unauthorized copying information by filtering the data. The developed tools (with the use of BEEP in browsers) allow to simplify the application of security policies in a web scripts. In addition, changes of network services for using proposed methods require minimal effort.
Keywords: security policy, web script, BEEP, tracking feature