تصميم اداة لاكتشاف ثغرة التهجين المخزن

انصاف جمعه صالح

فخر الدين عباس محمد

جامعة النيلين

مجلة كلية الدراسات العليا

الرقم الدولي الموحد: 1858-6228

المجلد: 15 ، 2020م
العدد: 10

كلية الدراسات العليا
جامعة النيلين

Graduate College Journal – NU
Vol.15 -2020, No. (10)
ISSN: 1858-6228, http://www.neelain .edu.sd

مجلة الدراسات العليا - جامعة النيلين
المجلد (15) –2020، العدد (10)
الرقم الدولي الموحد للدوريات: 1858-6228

تصميم اداة لاكتشاف ثغرة التهجين المخزن

انصاف جمعه صالح، فخر الدين عباس محمد
كلية علوم الحاسوب وتقانة المعلومات – جامعة النيلين- السودان.
insaafict@hotmail.com , fakhry00@gmail.com

المستخلص

مع ازياد استخدام تطبيقات الويب فى مختلف المجالات ازدادت الحاجة لمعرفة المهددات التى تواجهها من اجل العمل على حمايتها منها وذلك لتصاعد الهجمات التى تستهدفها. من هذه الهجمات هجوم التهجين المخزن (Stored XSS) والذى يعتمد فى نجاحه على سماح الموقع للمستخدم بادخال بيانات يقوم الموقع بحفظها فى قواعد البيانات، حيث يقوم المهاجم بادخال اكواد خبيثة الى الموقع وتظهر الآثار الضارة لهذه الاكواد عندما يتم لاحقا عرض هذه البيانات ضمن محتوى الموقع، وتظل موجودة ما لم تتم معرفتها وازالتها. تقدم هذه الورقة اداة Stored XSS Bruter وهى اداة مفتوحة المصدر تم تطويرها لاكتشاف وجود هذه الثغرة من عدمه فى الموقع الالكترونى.

الكلمات المفتاحية : هجوم التهجين، هجوم التهجين المخزن، فحص تطبيقات الويب

1. المقدمة

مع تقدم التكنولوجيا والانترنت اصبح تصفح مواقع الانترنت جزءا لا يتجزأ من تفاصيل الحياة اليومية للعديد من البشر. ومع ازياد فوائد هذه المواقع وتعدد استخدامها تتزايد المخاطر التى تتهددها. من هذه المخاطر هجوم التهجين المخزن (Stored XSS) وهو هجوم يتم عن طريق ادخال بيانات تحتوى على اكواد ضارة بواسطة المستخدمين ليتم حفظها فى قاعدة البيانات؛ ويتم تنفيذ هذه الاكواد فى كل مرة يقوم فيها الزائر بتصفح الصفحة التى تحتوى عليها.

تهدف هذه الورقة الى اقتراح آلية لاكتشاف وجود ثغرة التهجين المخزن (Stored XSS) من عدمه فى الموقع الالكترونى وتطوير اداة فحص بناءا على الآلية المقترحة لفحص احتواء الموقع على هذه الثغرة.

يتناول الجزء الثانى من الورقة هجوم التهجين المخزن (Stored XSS) بتوضيح خطورته واسباب صعوبة اكتشافه وكيفية الوقاية منه. ويتناول الجزء الثالث من الورقة المنهجية المتبعة فى هذه الورقة من خلال توضيح الخوارزمية المقترحة لاكتشاف الثغرة موضوع الدراسة. يوضح الجزء الرابع من الورقة الاداة التى تم تطويرها بالاعتماد على المنهجية المقترحة واخيرا يتناول الجزء الاخير من الورقة النتائج التى تم الحصول عليها بعد استخدام هذه الاداة ومناقشة هذه النتائج.

2. هجوم التهجين المخزن (Stored XSS)

يعتبر هجوم التهجين المخزن (Stored XSS) من اخطر انواع هجوم التهجين(XSS) . تعتبر التطبيقات التى تسمح للمستخدمين بتخزين بيانات فيها هى الاكثر عرضة لهذا الهجوم. يحدث هجوم التهجين المخزن (Stored XSS) عندما يقوم الموقع الالكترونى بجمع مدخلات من المستخدمين -قد تكون ضارة -ومن ثم تخزين هذه المدخلات ليتم استخدامها لاحقا. واذا لم يتم تنقية (filtered) هذه المدخلات بصورة صحيحة؛ فإن هذه البيانات الخبيثة ستظهر كجزء من محتوى الموقع الالكترونى فى متصفح المستخدم وستنفذ بنفس صلاحيات الموقع الالكترونى. تعرف هذه الثغرة ايضا بهجوم التهجين من الدرجة الثانية (second-order XSS) وذلك لانه يتطلب على الاقل ارسال طلبين (two requests) للموقع الالكترونى (Owasp Testing Guide v4, 2020).

يمكن استخدام هذه الثغرة لتنفيذ عدد من الهجمات التى تعتمد على المتصفح ومنها(Owasp Testing Guide v4, 2020) :

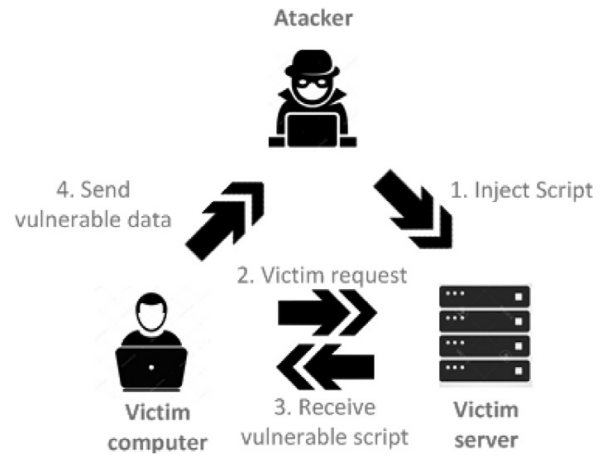
- الحصول على المعلومات الحساسة التى يتم عرضها بواسطة مستخدمى الموقع.
- طمس الموقع الالكترونى.

- فحص المنافذ المستخدمة بواسطة الاجهزة الداخلية (اجهزة داخلية بالنسبة لمستخدم الموقع الالكترونى).
- الاستيلاء على جلسة متصفح مستخدم آخر.
- تنفيذ الاكواد الضارة التى تعتمد على المتصفح.
- بالاضافة الى العديد من الانشطة الضارة.

لا يحتاج هجوم التهجين المخزن (Stored XSS) الى وجود وصلة تشعبية ليقوم بإستغلالها. ويعتبر الهجوم ناجحا عندما يقوم المستخدم بزيارة صفحة تحتوى على هجوم تهجين مخزن . يوضح السيناريو التالى المراحل الاساسية فى هجوم التهجين المخزن (Owasp Testing Guide (Stored XSS) v4, 2020):

- تخزين اكواد ضارة فى الصفحة التى تحتوى على ثغرة وذلك بواسطة المهاجم.
- تسجيل الدخول للموقع بواسطة المستخدم المخول.
- زيارة الصفحة التى تحتوى على ثغرة بواسطة المستخدم المخول.
- تنفيذ الاكواد الضارة بواسطة متصفح المستخدم المخول.

يعتبر هجوم التهجين المخزن (Stored XSS) اكثر خطورة عندما يتم حقن الاكواد الضارة فى الصفحات الخاصة بالمستخدمين اصحاب الصلاحيات العالية. فمثلا عندما يقوم المدير بزيارة صفحة تحتوى على ثغرة وتم حقن اكواد ضارة بها ؛ يتم تنفيذ الهجوم بصورة تلقائية بواسطة المتصفح الذى يستخدمه. وقد ينتج عن ذلك كشف معلومات حساسة مثل متسلسلة



تخويل الجلسة (session authorization tokens) (Owasp Testing Guide v4, 2020).

شكل 1 سيناريو تنفيذ هجوم التهجين المخزن

المصدر : Cross-site scripting (XSS) attacks and mitigation (Rodríguez et al, 2020)

يتضح من خلال الشكل 1 ان المهاجم يقوم بحقن الاكواد الضارة فى الخادم وبالتالى تكون هذه الاكواد دائمة فى الموقع ويتم تنفيذها فى كل مرة يتم فيها تصفح الصفحة التى تحتوى على الاكواد الضارة (Rodríguez et al, 2020).

I. مواضع حقن الاكواد الضارة فى التطبيق

من البديهى عدم الوثوق بالمتصفح. على الرغم من ان اغلب مصادر الهجوم هى اما حقول نماذج الادخال او الوصلات التشعبية. الا انه يجب التعامل مع كل البيانات القادمة من المتصفح على انها ملوثة. اذا كان التطبيق يعتمد على قيم قادمة من المتصفح فمن المحتمل ان تكون هذه المعلومات ضارة بغض النظر من مصدر المعلومات سواء كان المستخدم قام بإدخالها يدويا ام تم توليدها تلقائيا بواسطة المتصفح. (Shema and Ely, 2010)

- **مكونات معرف المصدر الموحد (*Uniform Resource Identifier Components*):** يمكن التلاعب بجزء من مكونات معرف المصدر بواسطة هجوم التهجين (Stored XSS). يقوم الخادم بالتعامل مع اسماء المفات،اسماء الادلة، والمعاملات (اسم المعامل وقيمته) بطريقة ما مما يشكل مصدرا للخطر حتى وان كانت خاطئة مثلا قد تشير الى صفحة غير موجودة. اذا كان الموقع يقوم بطباعة الوصلة التشعبية على الصفحة فمن المحتمل ان يتم استغلالها. مثلا اذا كان الموقع يطبع معرف المصدر عند عدم وجود الصفحة التى تشير اليها الوصلة التشعبية كالتالى (Shema and Ely, 2010):

```
<html>
Oops! We couldn't find http://some.site/nopage"<script></script>.
Please return to our <a href=/index.html>home page</a>
</html>
```

- **حقول نماذج الادخال** : تستخدم النماذج لجمع معلومات من المستخدمين مما يلمح الى ان هذه البيانات قد تكون ملوثة. وهذا ينطبق بصورة واضحة على الحقول التى يفترض ان يقوم المستخدم بملئها. وبصورة غير واضحة فى الحقول التى لا يتوقع ان يتم التعديل فى قيمها بواسطة المستخدم، مثل الحقول

المخفية وحقول العرض. يمكن تعديل قيمة اى حقل قبل ارساله للخادم ولذا لا يجب التعامل مع اى بيانات يتم الحصول عليها من جانب العميل على انها آمنة (Shema and Ely, 2010).

- **المحتوى المضاف بواسطة المستخدم** : يمكن ان تحتوى الملفات الثنائية مثل ملفات الصور والفيديو وملفات بى دى اف (PDF) على اكواد جافا سكريبت مضمنة بداخلها او اى اكواد اخرى يتم تنفيذها بواسطة المتصفح. تعتمد مواقع مشاركة المحتوى بصورة اساسية على العناصر التى يتم اضافتها بواسطة المستخدمين. الهجمات التى تتم بواسطة هذه الطريقة تعتبر اقل شيوعا ولكن ذلك لا يقتضى انها اقل خطرا (Shema and Ely, 2010).
- **ترميز كائن جافا سكريبت (**JavaScript Object Notation (JSON)**)** : هى طريقة لتمثيل انواع البيانات كسلسلة نصية بواسطة لغة جافا سكريبت ليتم نقلها بواسطة بروتكول نقل النصوص المتشعبة. فمثلا يمكن ان تستخدم مواقع البريد الالكترونى رموز جوسون لاسترجاع بيانات الرسائل الالكترونية او معلومات جهات الاتصال. بيانات جوسون تاتى من المصادر السابقة (نماذج الادخال ، معرف المصدر،...) . تمرير المحتوى معالج جوسون ودوال ()eval يضيف العديد من المخاطر الامنية. وذلك لسهولة اعتراض اكواد لغة جافا سكريبت وامكانية تعديل الدوال. اسهل وسيلة للحماية من هذا النوع من الثغرات هو باستخدام اطارت لغة جافا سكريبت (مثل Dojo, jQuery, YUI) والتى تحتوى على منهجيات للتعامل مع البيانات غير الموثوق بها (Shema and Ely, 2010).
- **خصائص نموذج كائن المستند (DOM)**: من طرق تنفيذ هجوم تهجين الموقع (XSS) استخدام كائن المستند وذلك لتعديل المستند بصورة غير متوقعة. حيث يقوم المهاجم باسناد اكواده الضارة الى احدى خصائص كائن المستند لتتم قراءتها وطباعتها بواسطة نفس الصفحة (Shema and Ely, 2010). مثلا :

```
document.write("<p>URL: " + document.location + "</p>")
```

اذا كان بالامكان تعديل خاصية document.location لتحتوى على اكواد ضارة بلغة الترميز التشعبية (HTML) فبإمكان المهاجم استغلال المتصفح. ويمكن ان تحتوى هذه الخاصية على معرف المصدر المستخدم فى الطلب وبالتالى يمكن للمهاجم تعديله بسهولة . والفارق المهم والدقيق هنا هو ان الخادم لن يعلم بقيمة هذه الخاصية وبالتالى لن يقوم بطباعتها على الصفحة. ويحدث الهجوم فقط فى المتصفح من خلال صياغة معرف مصدر ضار بواسطة المهاجم ، قد يكون ذلك باضافة وسوم سكريبت الى جملة الاستعلام كالتالى (Shema and Ely, 2010) :

http://bugzilla/enter_bug.cgi?<script>...</script>

II. اسباب صعوبة اكتشاف هجوم التهجين المخزن (Stored XSS):

هناك عدد من الاسباب تجعل هجوم التهجين المخزن (Stored XSS) غير سهل الاكتشاف وهى:

1. اغلب ادوات فحص التطبيقات تعمل من خلال ارسال طلب يحتوى على الاكواد الضارة للصفحة المستهدفة ورصد استجابة الموقع الالكترونى لهذا الطلب. فى حالة الهجوم المخزن نجد ان اكتشاف هذه الثغرة يتطلب ارسال طلبين او اكثر لنفس الصفحة حتى يتم رصدها (Owasp Testing Guide v4, 2020).
2. نتيجة الهجوم المخزن (Stored XSS) لا تظهر مباشرة على الصفحة التى تم حقنها بالاكواد الضارة وذلك على عكس الهجوم المنعكس (Reflected XSS) والذى تظهر نتيجته مباشرة على الصفحة(Rodríguez et al, 2020).

III. الحماية من هجوم التهجين المخزن (Stored XSS)

يمكن اتباع الاجراءات الوقائية التالية (Lepofsky, 2014):

- اجراء فحص الثغرات لاكواد لغة الترميز التشعبية (HTML) ولغة جافا سكريبت (Java Script).
- اجراء فحص لقنوات الاتصال الخارجية. بالاضافة الى ذلك فحص الثغرات واجراء تحليل دقيق لكيفية استقبال وتخزين بيانات المستخدم من القنوات الخارجية. تشير قنوات الاتصال الخارجيه فى هذا الصياغ الى الآليات الاخرى المستخدمة الى جانب حقول الادخال.
- اجراء الفحص الادارى وتحديد مدخلات المستخدم. يجب اجراء عملية فحص لكل اجزاء التطبيق التى يتم التعامل معها بواسطة

المدراء لتحديد بيانات المستخدم الموجودة فيها ومن ناحية اخرى لكل الاجزاء المقيدة من التطبيق.

- تحديد الترميز المسموح به للغة ترميز النصوص المتشعبة (HTML). تحديد آلية الترميز المستخدمة لكل صفحات اللغة التى سيتم عرضها من خلال المتصفح، مثل UTF-8.
- الحرص على فلترة وتصحيح المدخلات. ينطبق هذا على البيانات المرسلة من التطبيق للمتصفح والعكس.
- توفير تدريب التوعية الامنية. يجب ان يعزز التدريب مفهوم عدم الاستجابة لاى جهة (بريد الكترونى، محادثة آنية، او مكالمة هاتفية) تطلب من المستخدم معلوماته الشخصية او التعريفية.

3. المنهجية المتبعة فى الورقة العلمية

تعتمد المنهجية المقترحة على فحص كل الصفحات الموجودة فى الموقع الذى نرغب بفحصه وتتم عملية الفحص فى مرحلتين . تهدف المرحلة الاولى الى البحث عن نماذج الادخال الموجودة فى كل صفحة من صفحات الموقع ومن ثم ملء كل نموذج من نماذج الادخال ببيانات ضارة تتم صياغتها بصورة مخصصة لذلك النموذج. بعد اكتمال المرحلة الاولى تبدأ المرحلة الثانية والتى يتم فيها البحث عن البيانات التى تم حقنها من خلال نماذج الادخال حيث يتم البحث فى كل صفحات الموقع عن تلك البيانات الضارة وذلك لانه لا يمكن التنبؤ بالموضع الذى سيتم عرضها فيها ولذلك يتم تصفح كل صفحات الموقع بحثا عنها واذا تم العثور عليها -كلها او اى واحدة منها- فهذا يعنى ان الموقع معرض لهجوم التهجين المخزن (Stored XSS). والخطوات المقترحة لذلك هى:

أ) ادخال عنوان الموقع الذى نرغب بفحصه

فى هذه المرحلة نحصل على العنوان الكامل للموقع الذى نرغب فى فحصه.

ب) اكتشاف الموارد الموجودة فى الموقع (الصفحات)

ويتم فى هذه المرحلة تصفح الموقع بحثا عن الصفحات الموجودة فيه ويتم ذلك من خلال زاحف يقوم باتباع الوصلات التشعبية من صفحة الى اخرى وبعد ذلك يتم البحث عن الصفحات التى لم يتم التعرف عليها بواسطة الزاحف (وهى الصفحات غير المتصلة مع بقية صفحات الموقع من خلال الوصلات التشعبية) باستخدام مفهوم القوى العمياء (Brute Force) والذى يعتمد على استخدام قاموس Fuzzdb وهو قاموس مفتوح المصدر يحتوى على اشهر الاسماء المستخدمة لتسمية صفحات مواقع الانترنت (fuzzdb-project/fuzzdb, 2020). ويتم حفظ اسماء كل الصفحات التى تم اكتشافها بواسطة الزاحف والقاموس وذلك لاستخدامها فى المراحل التالية.

ج) البحث عن نماذج الادخال فى كل صفحة وحقن اكواد ضارة فى كل نموذج

بعد الانتهاء من مرحلة اكتشاف الموارد يتم استعادة كل الصفحات التى تم اكتشافها واحدة تلو الاخرى ومن ثم البحث عن نماذج ادخال البيانات (input Form) فى كل صفحة وصياغة مدخلات خاصة بكل نموذج وارسال هذه المدخلات ليتم حفظها فى قاعدة البيانات.

د) البحث عن الاكواد الضارة التى تم حقنها

يتم فى هذه المرحلة المرور على كل الصفحات واحدة تلو الاخرى بحثا عن البيانات التى تم ادخالها سابقا وذلك لانه لا يمكن معرفة الصفحة التى سيتم عرض البيانات فيها وعليه من خلال هذه المرحلة سنتمكن من رصد البيانات التى تم ادخالها ومنها نستنتج الصفحة التى تم ادخال البيانات منها والنموذج المستخدم فى الادخال.

هـ) كتابة التقرير النهائي للفحص

بعد اكتمال كل المراحل السابقة يتم عرض التقرير النهائي للفحص واذا كان الموقع يحتوى على هذه الثغرة يحتوى التقرير على عدد الثغرات التى تم العثور عليها ومواقع هذه الثغرات ودرجة خطورتها؛ اما اذا كان الموقع لا يحتوى على هذه الثغرة فيتم توضيح ذلك فى التقرير النهائي للفحص.

4. برنامج Stored XSS Bruter 1.0

هذا البرنامج هو برنامج مفتوح المصدر وقد تم تطويره باستخدام لغة البرمجة بايثون (python).

يوضح الشكل 2 تنفيذ برنامج Stored XSS Bruter حيث يظهر شعار البرنامج وتظهر رسالة تطلب من المستخدم ادخال عنوان الموقع الى يرغب بفحصه.

```
C:\Users\LENOVO\Anaconda3\python.exe C:/Users/LENOVO/Desktop/scrappy/xss.py
 ____  _                     _  __  ______ ____    ____             _
/ ___|| |_ ___  _ __ ___  __| | \ \/ / ___/ ___|  | __ ) _ __ _   _| |_ ___ _ __
\___ \| __/ _ \| '__/ _ \/ _` |  \  /\___ \___ \  |  _ \| '__| | | | __/ _ \ '__|
 ___) | || (_) | | |  __/ (_| |  /  \ ___) |__) | | |_) | |  | |_| | ||  __/ |
|____/ \__\___/|_|  \___|\__,_| /_/\_\____/____/  |____/|_|   \__,_|\__\___|_|


Enter Scan Target URL (exampl: http://target/FUZZ
Enter URL :
```

شكل 2 تنفيذ برنامج Stored XSS Bruter

الشكل 3 يوضح مرحلة اكتشاف الموارد بواسطة البرنامج وتتكون نتيجة الاستجابة لكل مورد من البيانات التالية:

- Time Elapsed وهى الفترة الزمنية المستغرقة لارسال الطلب والحصول على استجابه له محسوبة بالثوانى.
- Code وهو رمز الاستجابة الخاص ببرتكول نقل النصوص المتشعبة (HTTP).
- Reason وهو تفسير لرمز الاستجابة.
- URL وهو العنوان الذى تم ارسال طلب للحصول عليه.
- Response Size وهو حجم الاستجابة بالبايت.

```
-----------------------------------------------------------
                          Resouce Discovery
-----------------------------------------------------------
---------------------------------------------------------------------------
Time Elapsed        Code  Reason      URL                     Response Size
---------------------------------------------------------------------------
0.07434487342834473 200 OK  http://localhost/dvwa/about.php 4840 Bytes
0.03009963035583496 200 OK  http://localhost/dvwa/instructions.php  14266 Bytes
0.02005290985107422 200 OK  http://localhost/dvwa/login.php 1523 Bytes
0.011029958724975586     200 OK  http://localhost/dvwa/setup.php 4110 Bytes
0.005015134811401367     200 OK  http://localhost/dvwa/config/config.inc.php 0 Bytes
0.006051540374755859     200 OK  http://localhost/dvwa/config/config.inc.php.bak 1849 Bytes
0.00501704216003418 200 OK  http://localhost/dvwa/config/config.inc.php.dist     1857 Bytes
0.007014751434326172     200 OK  http://localhost/dvwa/dvwa/includes/dvwapage.inc.php     45 Bytes
0.0070514678955078125     200 OK  http://localhost/dvwa/dvwa/includes/dvwaphpids.inc.php  159 Bytes
0.012031078338623047     200 OK  http://localhost/dvwa/dvwa/includes/dbms/mysql.php  394 Bytes
0.00605010986328125 200 OK  http://localhost/dvwa/dvwa/includes/dbms/pgsql.php  256 Bytes
0.015051126480102539     200 OK  http://localhost/dvwa/external/phpids/0.6/docs/examples/example.php 107 Bytes
0.006016731262207031     200 OK  http://localhost/dvwa/external/phpids/0.6/docs/examples/cakephp/ids.php 158 Bytes
0.007018566131591797     200 OK  http://localhost/dvwa/external/phpids/0.6/docs/examples/cakephp/intrusion.php   166 Bytes
0.0060160160064697266    200 OK  http://localhost/dvwa/external/phpids/0.6/docs/phpdocumentor/phpids/_caching---database.php.html     5512 Bytes
0.0060350894927978516    200 OK  http://localhost/dvwa/external/phpids/0.6/docs/phpdocumentor/phpids/_caching---factory.php.html 3240 Bytes
```

شكل 3 مرحلة اكتشاف الموارد بواسطة برنامج Stored XSS Bruter

الشكل 4 يوضح مرحلة البحث عن نماذج الادخال فى كل صفحة واذا تم العثور على نموذج ادخال يتم صياغة بيانات خاصة به وارسال هذه البيانات ليتم حفظها فى قاعدة البيانات.

```
[+] Detected 0 forms on http://localhost/dvwa/external/phpids/0.6/lib/ids/vendors/htmlpurifier/htmlpurifier/urifilter/makeabsolute.php.
[+] Detected 1 forms on http://localhost/dvwa/vulnerabilities/xss_s/index.php.
***  Payload Injected  ***
[+] Detected 0 forms on http://localhost/dvwa/vulnerabilities/csp/source/impossible.php.
[+] Detected 0 forms on http://localhost/dvwa/external/phpids/0.6/lib/ids/vendors/htmlpurifier/htmlpurifier/attrtransform/safeobject.php.
[+] Detected 0 forms on http://localhost/dvwa/external/phpids/0.6/docs/phpdocumentor/phpids/_init.php.html.
[+] Detected 0 forms on http://localhost/dvwa/external/phpids/0.6/lib/ids/vendors/htmlpurifier/htmlpurifier/strategy/composite.php.
[+] Detected 0 forms on http://localhost/dvwa/external/phpids/0.6/lib/ids/vendors/htmlpurifier/htmlpurifier/language/messages/en.php.
[+] Detected 0 forms on http://localhost/dvwa/external/phpids/0.6/lib/ids/vendors/htmlpurifier/htmlpurifier/definitioncachefactory.php.
[+] Detected 0 forms on http://localhost/dvwa/vulnerabilities/upload/source/medium.php.
[+] Detected 0 forms on http://localhost/dvwa/instructions.php.
[+] Detected 0 forms on http://localhost/dvwa/external/phpids/0.6/lib/ids/vendors/htmlpurifier/htmlpurifier/stringhashparser.php.
[+] Detected 0 forms on http://localhost/dvwa/external/phpids/0.6/lib/ids/vendors/htmlpurifier/htmlpurifier/attrdef/css/filter.php.
[+] Detected 0 forms on http://localhost/dvwa/external/phpids/0.6/lib/ids/vendors/htmlpurifier/htmlpurifier/configschema.php.
[+] Detected 0 forms on http://localhost/dvwa/vulnerabilities/xss_d/source/low.php.
```

شكل 4 مرحلة البحث عن نماذج الادخال فى كل صفحة بواسطة برنامج Stored XSS Bruter

يوضح الشكل 5 مرحلة البحث عن البيانات التى تم ادخالها وفى حالة العثور عليها يتم كتابة عنوان الصفحة التى تحتوى على البيانات الضارة (Payload) ونص البيانات الضارة والصفحة التى ادخال هذه البيانات الضارة بواسطتها كما يتم ايضا كتابة درجة الخطر اما اذا لم يتم العثور على بيانات ضارة بتلك الصفحة فتتم كتابة رسالة تفيد بذلك.

```
scanning: http://localhost/dvwa/vulnerabilities/xss_r/help/help.php
***  No Payload Detected on this URL  ***
scanning: http://localhost/dvwa/vulnerabilities/captcha/source/low.php
***  No Payload Detected on this URL  ***
scanning: http://localhost/dvwa/external/phpids/0.6/lib/ids/vendors/htmlpurifier/htmlpurifier/urifilter/makeabsolute.php
***  No Payload Detected on this URL  ***
scanning: http://localhost/dvwa/vulnerabilities/xss_s/index.php
[+] Stored XSS Detected on http://localhost/dvwa/vulnerabilities/xss_s/index.php
[*] Payload details:
("<Script>alert('action :, method :post, inputs : txtName btnSign btnClear "
 "user_token')</scripT>")
[-] Originated From: http://localhost/dvwa/vulnerabilities/xss_s/index.php
[-]Risk: High
```

شكل 5 مرحلة البحث عن البيانات الضارة (Payload) فى كل صفحة بواسطة برنامج Stored XSS Bruter

```
------------------------------------------------------------
                        Scan Results
------------------------------------------------------------
Scan Target: http://localhost/dvwa/
Scan Date: 13 March 2020
Time Elapsed: 12.371018409729004 Seconds
Total URL Scanned: 371
Total Forms Detected: 5
Total Payloads Injected: 5
Total Stored XSS Vulnerabilites Detected: 4
------------------------------------------------------------
Vulnerability(ies) Details:
[+] Stored XSS Detected on http://localhost/dvwa/vulnerabilities/xss_s/index.php
[*] Payload details:
("<Script>alert('action :, method :post, inputs : txtName btnSign btnClear "
 "user_token')</scripT>")
[-] Originated From: http://localhost/dvwa/vulnerabilities/xss_s/index.php
[-]Risk: High


[+] Stored XSS Detected on http://localhost/dvwa/vulnerabilities/xss_s/index.php
[*] Payload details:
("<Script>alert('action :, method :post, inputs : txtName btnSign btnClear "
 "user_token')</scripT>")
[-] Originated From: http://localhost/dvwa/vulnerabilities/xss_s/
[-]Risk: High
```

شكل 6 النتيجة النهائية للفحص بواسطة برنامج Stored XSS Bruter

يوضح الشكل 6 النتيجة النهائية للفحص ويتم فيها عرض الآتى :

- Scan Target وهو عنوان الموقع الذى تم فحصه.
- Scan Date وهو تاريخ اليوم الذى تم فيه البحث.
- Time Elapsed وهو عبارة عن الفترة الزمنية التى ساتغرقتها عملية الفحص بالثوانى.
- Total URL Scanned ويمثل العدد الكلى للعناوين التى تم فحصها فى الموقع.
- Total Forms Detected ويمثل العدد الكلى لنماذج الادخال التى تم العثور عليها فى الموقع.
- Total Payloads Injected ويمثل العدد الكلى للاكواد الضارة التى تم حقنها فى الموقع.
- Total Stored XSS Vulnerabilities Detected ويمثل العدد الكلى للاكواد الضارة التى تم اكتشافها فى الموقع (من الاكواد الضارة التى تم حقنها سابقا).
- Vulnerability(ies) Details ويمثل تفاصيل كل الثغرات التى تم اكتشافها ويتم عرض الآتى:
  - عنوان الصفحة التى تم اكتشاف الاكواد الضارة فيها.
  - نص الاكواد الضارة التى تم اكتشافها.
  - عنوان الصفحة التى تم حقن الاكواد الضارة بواسطتها.
  - درجة الخطر.

```
CWE ID: 79
WASC ID: 8

Description:
The software does not neutralize or incorrectly neutralizes
user-controllable input before it is stored in the database
and placed in output that is used as a web page that is served to other users.

Solution:
validating and escaping user inputs.
```

شكل 7 الجزء الاخير من النتيجة النهائية للفحص بواسطة برنامج Stored XSS Bruter

يتم عرض المعلومات التالية فى الشكل رقم 7:

- CWE ID وهو رقم هجوم التهجين المخزن فى قائمة CWE وهى قائمة تحتوى على الثغرات الامنية الشائعة فى البرمجيات والعتاد (CWE - Common Weakness Enumeration, 2020).
- WASC ID وهو رقم هجوم التهجين المخزن فى قائمة اتحاد امن تطبيقات الويب (Web application security consortium ("The Web Application Security Consortium / Cross Site Scripting", 2020)
- وصف لاسباب هجوم التهجين المنعكس.
- كيفية سد هذه الثغرة.

يوضح الشكل 8 التقرير النهائى للفحص فى حالة عدم وجود ثغرة حيث تظهر رسالة تفيد بعدم وجود ثغرة التهجين المخزن بالموقع موضوع الفحص.

```
------------------------------------------------------------
                        Scan Results
------------------------------------------------------------
Scan Target: http://localhost/waco/
Scan Date: 14 March 2020
Time Elapsed: 4.096141815185547 Seconds

 Congratulations

Your website is FREE from Stored XSS Vulnerability
```

شكل 8 التقرير النهائى للفحص فى حالة عدم وجود ثغرة بواسطة برنامج Stored XSS Bruter

5. النتائج والمناقشة

تم استخدام عدد من التطبيقات المجانية ومفتوحة المصدر لاختبار مدى نجاح المنهجية المقترحة فى اكتشاف هجوم التهجين المنخزن (Stored XSS). تم تصميم التطبيقات المستخدمة بغرض توفير تطبيقات يمكن للمبرمجين والمهتمين بمجال امن تطبيقات الويب تطوير مهاراتهم باستخدامها او اختبار ادوات الفحص التى يقومون بتطويرها عليها وغالبا ما تحتوى هذه التطبيقات على مثال واحد لكل ثغرة من الثغرات الشائعة فى تطبيقات الويب. وهى:

- DVWA
- bWAPP
- XVWA

تم الحصول على النتائج الموضحة فى الجدول رقم 1 عند فحص التطبيقات المذكورة فى باستخدام ادوات الفحص المذكوره اعلاه.

جدول رقم 1 عدد الثغرات التى تم اكتشافها فى كل تطبيق بواسطة برنامج Stored XSS Bruter

التطبيق	عدد الثغرات التى تم اكتشافها بواسطة الاداة
DVWA	1
bWAPP	1
XVWA	1

نسبة لان اغلب هذه التطبيقات تم تطويرها بغرض التدريب فإنها تحتوى على مثال لثغرة واحدة غالبا لكل نوع من انواع الثغرات – كما ذكر سابقا- وعليه فإن الاداة المعتمدة على المنهجية قد تمكنت من اكتشاف ثغرة واحدة بكل تطبيق من التطبيقات الاختبارية.

نظرا لخطورة هجوم التهجين المخزن (Stored XSS) فإن إستخدام هذه الاداة يساعد فى اكتشاف وجود هذه الثغرة بنماذج الادخال فى الموقع الالكترونى كما تساعد فى تحديد النموذج الذى يحتوى على هذه الثغرة ودرجة خطورتها وتقدم معلومات عن كيفية التعامل مع هذه الثغرة والعمل على سدها. والجدير بالذكر هنا ان هذه المنهجية المقترحة تستهدف اكتشاف ثغرة هجوم التهجين المخزن (Stored XSS) فى نماذج الادخال.

المصادر والمراجع

Cwe.mitre.org. 2020. CWE - Common Weakness Enumeration. [online] Available at: <https://cwe.mitre.org> [Accessed 1 September 2020].

GitHub. (2020). fuzzdb-project/fuzzdb. [online] Available at: https://github.com/fuzzdb-project/fuzzdb [Accessed 31 Aug. 2020].

kennel209.gitbooks.io. (n.d.). Testing for Stored Cross Site Scripting (OTG-INPVAL-002) | Owasp Testing Guide v4. [online] Available at: https://kennel209.gitbooks.io/owasp-testing-guide-v4/en/web_application_security_testing/testing_for_stored_cross_site_scripting_otg-inpval-002.html [Accessed 31 Aug. 2020].

Lepofsky, R. (2014). The Manager's Guide to Web Application Security A Concise Guide to the Weaker Side of the Web. Berkeley, Ca Apress.

projects.webappsec.org. (n.d.). The Web Application Security Consortium / Cross Site Scripting. [online] Available at: http://projects.webappsec.org/w/page/13246920/Cross%20Site%20Scripting [Accessed 31 Aug. 2020].

Rodríguez, G.E., Torres, J.G., Flores, P. and Benavides, D.E. (2020). Cross-site scripting (XSS) attacks and mitigation: A survey. Computer Networks, 166, p.106960.

Shema, M. and Ely, A., 2010. Seven Deadliest Web Application Attacks. Amsterdam: Syngress.