الجمهورية العربية السورية
جامعة البعث
كلية الهندسة المعلوماتية
قسم هندسة البرمجيات ونظم المعلومات

أمن تطبيقات الويب من جهة الزبون

Client-Side Web Application Security

رسالة أعدت لنيل درجة الماجستير في هندسة البرمجيات ونظم المعلومات

إعداد

المهندسة تهامة غسان قليشة

بإشراف

أ. د. ناصر أبو صالح

أستاذ في قسم هندسة البرمجيات ونظم المعلومات

كلية الهندسة المعلوماتية جامعة البعث

1441هـ-2020 م

جامعة البعث

كلية الهندسة المعلوماتية

الرقم: ٤٠٢/د

التاريخ: ٢٦/٧/٢٠٢٠

بيان بإجراء التعديلات العلمية واللغوية

قامت الطالبة تهامة قليشة بتصحيح الملاحظات التي وردت في متن رسالة الماجستير والتي أشار إليها الدكاترة أعضاء لجنة الحكم خلال مناقشة الرسالة.

رئيساً للجنة الحكم	مشرفاً وعضواً	عضواً
د. كمال السلوم	د. ناصر أبو صالح	د. عصام حماده

رئيس قسم هندسة البرمجيات ونظم المعلومات

د. كمال السلوم

الجمهورية العربية السورية

جامعة البعث

كلية الهندسة المعلوماتية

قرار لجنة الحكم و المناقشة العلنية على رسالة الماجستير

بناء على قرار مجلس البحث العلمي والدراسات العليا في جامعة البعث رقم/(**539**) **د/** المتخذ بالجلسة رقم /(**16**) **د/** للعام الدراسي **2020/2019** المنعقدة بتاريخ **6/ رمضان 1441هـ الموافق لـ 4/29/ 2020م** المتضمن الموافقة على تشكيل لجنة الحكم والمناقشة على رسالة الماجستير للطالبة **تهامة غسان قليشة** باختصاص **هندسة البرمجيات ونظم المعلومات** في قسم **هندسة البرمجيات ونظم المعلومات** في كلية **الهندسة المعلوماتية بجامعة البعث** المؤلفة من السادة الدكاترة:

	الاسم و الشهرة	المرتبة	الاختصاص	الكلية	الجامعة	الصفة
1	**د. ناصر أبو صالح**	أستاذ	ذكاء صنعي	الهندسة المعلوماتية	البعث	عضواً ومشرفاً
2	**د. كمال السلوم**	أستاذ	بحوث عمليات	الهندسة المعلوماتية	البعث	عضواً
3	**د. غسان حماده**	أستاذ مساعد	معلوماتية-أنفورماتيك	العلوم	حماه	عضواً

- اجتمعت اللجنة في تمام الساعة الحادية عشرة والنصف من يوم **الاثنين** تاريخ **2020/06/22** وناقشت علناً الرسالة التي قدمتها الطالبة **تهامة غسان قليشة** بعنوان:

/ أمن تطبيقات الويب من جهة الزبون /

- استعرضت الطالبة أطروحتها خلال المدة المحددة لها ثم بدأت المناقشة العلنية من قبل اللجنة والسادة الحضور وتم تسجيل الملاحظات المتعلقة بالرسالة والإجابة عن استفسارات اللجنة و الحضور.
- واستناداً إلى قرار مجلس جامعة البعث **رقم** /855/ **تاريخ 2017/3/22** المتضمن تسجيل رسالة الماجستير للطالبة المذكورة وبعد أن خلت لجنة الحكم المذكورة أعلاه بنفسها قررت الآتي :

1- اعتبار رسالة الماجستير المقدمة من الطالبة المذكوة عملاً علمياً يؤهلها لنيل درجة الماجستير وتستحق علامة الرسالة وقدرها / 91 إحدى وتسعون درجة / وبتقدير امتياز.

2- علماً أن الطالبة قد نجحت في جميع مقرراتها في السنة الأولى للماجستير.

3- وبالتالي تقترح اللجنة استحقاق طالبة الدراسات العليا **تهامة غسان قليشة** منحها درجة الماجستير في قسم **هندسة البرمجيات ونظم المعلومات** اختصاص **هندسة البرمجيات ونظم المعلومات** في كلية **الهندسة المعلوماتية** بجامعة البعث بتقدير / **جيد جداً** / وعلامة قدرها **79.66** (تسع وسبعون درجة وست وستون جزء من الدرجة فقط لا غير).

4- توصي اللجنة بتبادل الرسالة مع الجامعات العربية و الأجنبية و طباعتها و نشرها على نفقة جامعة البعث بعد إجراء التعديلات التي أوصت بها اللجنة.

5- يرفع هذا القرار إلى مجالس الجامعة المختصة لاستكمال الإجراءات القانونية لمنحه الدرجة المذكورة لتمتعه بكافة حقوق هذه الدرجة.

حمص في 2020/06/22م .

أعضاء اللجنة

الدكتور	الدكتور	الدكتور
غسان حماده	**ناصر أبو صالح**	**كمال السلوم**

Al Baath University
Journal of Al Baath University

جامعة البعث
مجلة جامعة البعث

قبول نشر بحث
طالب دراسات عليا

الرقم: ٧٠٧
التاريخ ١٥ / ٢ / 2020

طالب الدراسات العليا: **تهامة غسان قليشة** الدكتور المشرف : **ناصر أبو صالح**

كلية: **الهندسة المعلوماتية** – جامعة: **البعث**

نود إعلامكم بقبول بحثكم الموسوم:

أمن تطبيقات الويب من جهة الزبون

للنشر في مجلة جامعة البعث بالمجلد **42** لعام **2020**
بعد أن تم تحكيمه من قبل مختصين.

نشكر لكم هذه المساهمة الطيبة ونتطلع إلى استمرار تواصلكم مع مجلتنا ومدها بما لديكم من جديد والاطلاع على الأبحاث المنشورة في المجلة على موقع المجلة والرابط المدونين في أسفل الصفحة.

موقع الجامعة www.albaath-univ.edu.sy والرابط magazine @ albaath-univ.edu.sy

رئيس هيئة تحرير مجلة جامعة البعث

أ.د ناصر سعد الدين

الملخص:

مع احتلال الانترنت لمكان دائم في حياتنا اكتسبت النواحي الأمنية أهمية أكبر وهذا ما تؤكده التغطية الإعلامية الكبيرة للحوادث الأمنية الخطيرة.

عبر السنوات الأخيرة تحول مركز ثقل الشبكة من المخدم إلى المستخدم، حيث أصبح المستعرض بيئة تنفيذ متكاملة للتطبيقات الديناميكية والمعقدة. لسوء الحظ كذلك انتقل تركيز المهاجمين نحو الهجمات المعتمدة على المستعرض وتهديد منصة عمل الزبون. ومن الطبيعي مع هذا الانتقال أن تتطور الاجراءات المضادة والسياسات الأمنية بهذا الاتجاه.

لتقييم مدى أمان موقع ما، غالبا ما يلجأ مدراء هذه المواقع إلى الشركات الاستشارية للقيام بعمليات فحص للاختراق **penetration testing** وعمليات مراجعة للكود البرمجي. إلا أنه من الصعب لطرف خارجي كالحكومة أو المنظمات الرقابية القيام بتقييم مدى أمان موقع ما خارجياً، خاصة إذا كان هذا التقييم يجري على مجال واسع، كأن يتضمن عدداً كبيراً من المواقع التابعة لبلد ما أو قطاع صناعي معين. هذا النوع من التقييم ضروري بما أن المواطنين يعتمدون بشكل متزايد يوما بعد يوم على تطبيقات محددة. وهذا مشابه مثلاً لضرورة التقييم الالزامي للسلامة الانشائية للمباني من أجل حماية المواطنين من كوارث مستقبلية كان من الممكن تجنبها.

في هذه الأطروحة سنقوم أولا بإلقاء الضوء على التطور من تطبيقات معتمدة على المخدم إلى التطبيقات المعاصرة المعتمدة على الزبون والتي تقدم تجربة مختلفة للمستخدم. سنستكشف المفاهيم المنطوية تحت هذه التطبيقات وسنوضح أهمية الأمن من جهة الزبون وسنقوم بشرح المخاطر وآخر التقنيات والأبحاث المطروحة لمواجهتها.

ثم سنقوم بدراسة مدى تبني المواقع السورية للآليات الأمنية من جهة الزبون وكذلك تقديم نموذج مرجعي للحالة الحالية لأمن الشبكة، كما سنطرح نظاماً لتسجيل النقاط الأمنية **security scoring system** لتقدير مستوى الأمان الذي يقدمه كل موقع.

الكلمات المفتاحية:

أمن تطبيقات، تطبيقات ويب، أمن من جهة الزبون، المواقع السورية، دراسة تجريبية، تقييم أمني

فهرس المحتويات

فهرس المحتويات II

الأشكال والمخططات V

الجداول VII

1 المقدمة 3

1.1 هدف البحث: 5

1.2 دراسات وأعمال سابقة: 5

1.3 لمحة عامة عن محتوى الرسالة: 6

2 تطور الشبكة وأهمية الأمن من جهة الزبون 11

2.1 لمحة عن الشبكة: 11

2.2 اللّبنات الأساسية لتطبيق الويب: 13

2.2.1 سياق التنفيذ الخاص بالزبون: 14

2.2.2 إدارة الجلسات: 16

2.2.3 التفاعلات بين الأصول المختلفة: 18

2.2.4 الطلبات والردود في الشبكة: 19

2.3 أهمية الأمن من جهة الزبون: 20

2.4 التوجهات الحديثة للسياسات الأمنية: 22

3 الهجمات على الشبكة NETWORK ATTACKS 27

3.1 هجمات التنصت EAVESDROPPING ATTACKS 27

3.1.1 طرق المواجهة: 27

3.1.2 الواقع الحالي: 28

3.1.3 أفضل الممارسات: 29

3.2 هجوم الرجل في المنتصف MITM 29

3.2.1 طرق المواجهة: 31

3.2.2 الواقع الحالي: 33

3.2.3 أفضل الممارسات: 34

3.3 الهجمات على مستوى بروتوكول الـHTTPS 34

4 الهجمات على طلبات المستعرض 39

4.1 تزوير الطلب العابر للموقع (CSRF) 39

4.1.1 طرق المواجهة: 42

4.1.2 الواقع الحالي: 44

4.1.3 أفضل الممارسات: 44

4.2 تعديل واجهة المستخدم UI REDRESSING 45

4.2.1 طرق المواجهة: 46
4.2.2 الواقع الحالي: 47
4.2.3 أفضل الممارسات: 47

5 الهجمات على الجلسة 51

5.1 **هجوم اختطاف الجلسة Session Hijacking 51**
5.1.1 طرق المواجهة: 52
5.1.2 الحالة الحالية: 53
5.1.3 أفضل الممارسات: 54
5.2 **هجوم تثبيت الجلسة Session Fixation 54**
5.2.1 طرق المواجهة: 55
5.2.2 الحالة الحالية: 56
5.2.3 أفضل الممارسات: 56
5.3 **توثيق المستخدم باستخدام بيانات مسروقة 56**
5.3.1 طرق المواجهة: 57
5.3.2 الحالة الحالية: 58

6 الهجوم على السياق من جهة الزبون 63

6.1 **البرمجة العابرة للموقع Cross-Site Scripting 63**
6.1.1 طرق المواجهة: 64
6.1.2 أفضل الممارسات: 68
6.2 **الهجمات من دون كود Scriptless Injection Attacks 68**
6.2.1 طرق المواجهة: 69
6.3 **تضمين كود غير موثوق Compromised Script Inclusions 69**
6.3.1 طرق المواجهة: 70

7 الهجمات على جهاز الزبون 73

7.1 **التنزيل العرَضي Drive-By Downloads 73**
7.1.1 طرق المواجهة: 74
7.1.2 الحالة الحالية: 75
7.1.3 أفضل الممارسات: 75
7.2 **ملحقات المستعرض الخبيثة 76**
7.2.1 طرق المواجهة: 77
7.2.2 الحالة الحالية: 77
7.2.3 أفضل الممارسات: 79

8 دراسة الواقع الأمني للمواقع السورية 83

8.1 **جمع البيانات: 84**
8.2 **أدوات الدراسة: 84**
8.3 **آلية الدراسة: 84**
8.4 **الميزات الأمنية تحت الدراسة: 85**
8.4.1 الميزات التي تساهم في تأمين الاتصال 85

8.4.2 *الميزات التي تواجه هجوم البرمجة العابرة للموقع* 85
8.4.3 *الميزات التي تمكّن من السماح بالتأطير بشكل آمن:* 86
8.5 **نقاط الضعف تحت الدراسة:**.....86
8.5.1 *تفاصيل نظام التقييم للميزات الأمنية:* 88
8.5.2 *تفاصيل نظام التقييم لنقاط الضعف:* 89
8.6 **النتائج:** 91
8.6.1 *الميزات التي تساهم في تأمين الاتصال:* 91
8.6.2 *الميزات التي تواجه هجوم البرمجة العابرة للموقع:* 92
8.6.3 *الميزات التي تمكن من السماح بالتأطير بشكل آمن:* 93
8.6.4 *تضمين ملفات جافا سكريبت خارجية غير موثوقة المصدر:* 93
8.6.5 *تضمين محتوى مختلط mixed-content:* 94
8.6.6 *إيقاف خاصية X-XSS-Protection:* 94
8.6.7 *استخدام برامج تخديم مهملة outdated server software:* 94
8.7 **الدرجات الأمنية للمواقع:**.....95
8.7.1 *الدرجات السلبية:* 95
8.7.2 *الدرجات الإيجابية:* 96
8.7.3 *متفرقات:* 96
8.7.3.1 الدرجات بحسب فئة الموقع: 96
8.7.3.2 الدرجات وفقاً للغة التطوير المعلنة:.....98
8.8 **التحديات:**.....100
8.8.1 *دقة الدراسة الساكنة للخصائص الأمنية* 100
8.8.2 *نظام التقييم* 100
8.9 **الاستنتاجات والتوصيات:** 101
ملحق A خاصية SAMESITE لملفات تعريف الارتباط.....103
ملحق B سياسة أمان المحتوى CSP 113
قائمة الاختصارات.....121
المراجع 123

الأشكال والمخططات

شكل 2-1: **(a)** لقطة شاشة **(b)**منظور هيكلي للتطبيق المصرفي غير الآمن. نلاحظ بالفعل أنه في الشكل الأيمن، يتواصل السياق ذو الأصل **B** مع الأصل **A** باستخدام جلسة عمل تم إنشاؤها بالفعل، وهو مصدر للعديد من مشكلات أمان الويب. 14

شكل 2-2 أجزاء عنوان صفحة الويب 15

شكل 2-3 إدارة الجلسة باستخدام ملفات تعريف الارتباط 17

شكل 3-1 مخطط يوضح التزايد في نسبة المواقع المستخدمة لـ**https** 29

شكل 3-2 مثال لترويسة تفعيل **HSTS** مع تفعيل خيار **preload** للتضمين المسبق بقائمة المواقع ... 32

شكل 3-3 مثال لترويسة تفعيل **PKP**..... 33

شكل 4-1 استخدام صورة مخفية لتفعيل طلب مزور 40

شكل 4-2 إرسال نموذج إدخال إلى عنوان ذو أصل مختلف باستخدام إطار مخفي يحتوي نموذج إدخال يتم إرساله لاحقاً للتطبيق المستهدف 40

شكل 4-3 آلية إرسال ملفات تعريف الارتباط مع الطلبات 41

شكل 4-4 استخدام ترويسة **SameSite** 43

شكل 4-5 هجوم خطف النقرات **clickjacking** باستخدام طبقة شفافة فوق الواجهة الحقيقية 45

شكل 4-6 القيم الممكنة لترويسة **X-Frame-Options** 46

شكل 4-7 أمثلة على استخدام خاصية **frame-ancestors**..... 46

شكل 4-8 مثال لكود برمجي يقيد الإطارات **frame busting code** 47

شكل 5-1 آلية تنفيذ هجوم اختطاف الجلسة 51

شكل 5-2 الإعدادات الأفضل لحماية معرفات الجلسات..... 54

شكل 5-3 آلية هجوم تثبيت الجلسة **Session Fixation** 54

شكل 5-4 في هجوم **TabNabbing** ، يقوم المهاجم بتبديل علامة تبويب غير ضارة (يسار) إلى صفحة تصيد (يمين) ، مما يجنبه من الاكتشاف عند قيام المستخدم بالتحقق من **URI** لعلامة تبويب تم تحميلها حديثًا 57

شكل 6-1 مثال على **reflected XSS** أو **XSS** المنعكس 64
شكل 6-2 مثال على هجوم **XSS** المخزن أو**stored xss** 64
شكل 6-3 استخدام خاصية **sandbox** مع السماح للإطار بإرسال بيانات نماذج الإدخال 66
شكل 6-4 تتيح هذه السياسة تحميل أي مادة عرض من الموقع المضيف والسماح فقط بتحميل البرامج النصية والأنماط والصور من **cdn.example.com** باستخدام **HTTPS**.....66
شكل 6-5 التضمين السليم لكود جافا سكريبت الخارجية باستخدام **nonce** 66
شكل 6-6 التضمين السليم لملفات الجافا سكريبت الخارجية باستخدام رموز التقطيع 67
شكل 6-7 مثال لاستخدام خاصية **SRI**.....67
شكل 6-8 استخدام **CSP** لفرض استخدام **SRI**.....67
شكل 6-9 **استخدام ترويسة X-Content-Type-Options**.....68
شكل 6-10 كل كود الـ**HTML** الموجود بعد فاصلة الاقتباس الخاصة بعنوان الصورة وحتى الوصول إلى فاصلة اقتباس لإغلاق العنوان ...سيتم اعتباره جزءاً من عنوان الصورة وسيتم إرسال الطلب الى العنوان الموافق مع قيمة رمز الحماية 69
شكل 7-1 تعطيل استخدام واجهات برمجة التطبيقات لملء الشاشة وتحديد الموقع الجغرافي 79
شكل 7-2 تستضيف شركة ما تطبيقًا على **"https://example.com"** وتريد تعطيل إدخال الكاميرا والميكروفون لأصلها مع تمكينه للموقع (**https://other.com**)الذي يقوم بتضمين موقعها.....79
شكل 8-1 استخدام النقل الآمن في المواقع السورية 91
شكل 8-2 إحصائيات استخدام إصدارات بروتوكولات النقل الآمن.....92
شكل 8-3 احصائيات استخدام برامج تخديم مهملة.....94
شكل 8-4 توزع الدرجات السلبية للمواقع.....95
شكل 8-5 توزع الدرجات الايجابية للمواقع.....96
شكل 8-6 نسب أمان المواقع بحسب الفئة.....97
شكل 8-7 متوسط الدرجات السلبية وفقاً لفئة الموقع.....98
شكل 8-8 لغات التطوير المستخدمة 98
شكل 8-9 توزع درجات الضعف بحسب لغة التطوير المستخدمة 99
شكل 8-10 متوسط نسبة أمان المواقع وفقاً للغة التطوير.....99

الجداول

جدول 2-1 قائمة **OWASP** للثغرات الأمنية العشرة الأكثر خطورة لعام 2017 20
جدول 2-2 قائمة **CWE/SANS** للأخطاء البرمجية الـ25 الأكثر خطورة 21
جدول 8-1 القيم المسندة لنقاط الضعف 90

الفصل الأول

المقدمة

1 المقدمة

Google [1] و **LinkedIn** [2] و **Adobe** [3] و **Yahoo** [4] و **eBay** [5] و **Reuters** [6]، قد لا يكون واضحاً العامل المشترك بين هذه الشركات لكن جميعها كانت ضحيةً لهجمات الشبكة مما جعل حسابات المستخدمين عرضةً للخطر أو سرقة معلوماتهم أو تشويه محرج للمواقع الالكترونية لهذه الشركات. القائمة تحتوي ست شركات كبيرة واعية تماماً لمخاطر الشبكة. أحد التقارير[7] يبين أن حوالي800 مليون شخص كانوا عرضة لسرقة معلومات شخصية وثلثي هؤلاء فقدوا نقوداً نتيجة لذلك.

غالباً ما يتم الاستهانة بالآثار البعيدة لهذه الهجمات على الشركات والأفراد. الشركات التي وقعت ضحية لها لا تعاني فقط من تقطعات مزعجة للعمل إنما قد تواجه دعاوى قضائية وتحقيقات حكومية. بالإضافة لتشويه سمعة يؤدي إلى ترك الزبائن للشركة وفقدان المساهمين للثقة بهذه الشركة. والأسوأ من ذلك أن الخروقات المتكررة ستؤدي إلى فقدان ثقة عام بالخدمات المقدمة عبر الشبكة مما يؤذي الشركات التجارية التي تقدم خدماتها على الشبكة كما يؤذي خدمات الحكومة الالكترونية.

أصبحت الجريمة الالكترونية تجارة بأرباح خيالية تصل إلى بلايين الدولارات مما يجعل الشبكة في وضع حرج.

بالإضافة لذلك مع تطور المستعرض إلى بيئة متكاملة تحول تركيز المسؤوليات ليتم دفعه باتجاه الزبون. لم تعد مهمة الزبون فقط عرض نتيجة عمل التطبيق الذي يعمل في الخلف لكن أصبح هو التطبيق الذي يتفاعل مع تطبيق خلفي خفيف مهمته الرئيسية عملية التخزين وذلك عبر واجهة برمجية غنية.

التقنيات التي تقف خلف هذا التطور هي مجموعة تقنيات خفيفة من جهة المخدم مثل (**NodeJS** و**Go**) ومكتبات من جهة الزبون مثل (**JQuery** و**Bootstrap**) وأطر عمل من جهة الزبون مثل (**AngularJS** و **Ember.js**). بالإضافة إلى أن المستعرضات تقدم ميزات غنية ومتعددة للتطبيقات مثل النفاذ لآليات التخزين في جهة الزبون والنفاذ لمعلومات عن الجهاز المستخدم وحساساته والعديد من آليات الاتصال.

مما شجع هذا التحول أيضاً ظهور الأجهزة النقالة مع أنظمة تشغيلها المقيدة ومخازن تطبيقاتها المتحكم بها من قبل الموزع. حيث ليس فقط أغلب تطبيقات اليوم الموجودة في مخازن التطبيقات مبنية على أساس تقنية الويب ولكن أيضاً هنالك جهود مستمرة لفرض معايير [8] تؤمن الواجهات البرمجية الضرورية لبناء تطبيقات

يمكن أن تتفاعل مع الآلة المضيفة مما يجعلها غير قابلة للتمييز عن التطبيقات المحلية (الموجودة مع نظام التشغيل).

نرى أن الشبكة شهدت العديد من الخطوات التطورية حولتها من محتوى ساكن من جهة المخدم إلى تطبيقات ديناميكية من جهة الزبون، وحولت المستعرضات من تطبيقات مملة لعرض الصفحات إلى بيئات تنفيذ تقوم بتشغيل تطبيقات قوية وعالية الديناميكية، مع هذا التطور انتقل تركيز المهاجمين نحو الهجمات المعتمدة على المستعرض وتهديد منصة عمل الزبون.

لمواجهة هذا التحول في الهجمات تم تطوير العديد من الآليات الأمنية مثل سياسة أمن المحتوى **Content Security Policy (CSP)** وترويسة **HTTP Strict-Transport-Security (HSTS)** وغيرها. هذه الآليات مقادة من قبل المخدم لكن تتطلب من المستعرض أن يضعها موضع التنفيذ (**Server-driven Client-enforced**). وجود هذه الآليات والعديد غيرها ضمن تطبيق ما يمكن أن يستخدم كمؤشر لمدى الوعي الأمني ومدى الممارسات الأمنية المستخدمة لهذا الموقع.

لتقييم مدى أمان موقع ما، غالبا ما يلجأ مدراء هذه المواقع إلى الشركات الاستشارية للقيام بعمليات فحص للاختراق **penetration testing** وعمليات مراجعة للكود البرمجي. إلا أنه من الصعب لطرف خارجي كالحكومة أو المنظمات الرقابية القيام بتقييم مدى أمان موقع ما خارجياً، خاصة إذا كان هذا التقييم يتم على مجال واسع، كأن يتضمن عدداً كبيراً من المواقع التابعة لبلد ما أو قطاع معين. هذا النوع من التقييم ضروري بما أن المواطنين يعتمدون بشكل متزايد يوماً بعد يوم على تطبيقات محددة. وهذا مشابه مثلاً لضرورة التقييم الإلزامي للسلامة الإنشائية للمباني من أجل حماية المواطنين من كوارث مستقبلية كان من الممكن تجنبها.

وفي عصرنا الحالي الذي أصبح فيه كل العالم يعتمد على تطبيقات الويب في كافة مجالات الحياة، لم يعد ممكناً تجاهل أهمية أمن التطبيقات لأنه العقبة الأساسية التي تقف دون استخدام هذه التطبيقات في المجالات الحياتية في بلدنا. فمثلاً لا يمكن تطبيق الحوكمة الالكترونية بشكل جدي في ظل غياب الوعي الأمني. هذا عدا عن التعاملات البنكية والدفع الالكتروني. وحتى في أبسط الأشكال كالتراسل الرسمي عبر المواقع الالكترونية.

انطلاقاً من مدى أهمية أمن التطبيقات من جهة الزبون وقلة الدراسات والأبحاث فيه وكذلك من أهمية التقييم الدوري لأمن المواقع، وذلك لحماية المستخدم وتعزيزا لثقته باستخدام هذه المواقع، قمنا بدراسة عن مدى تبني المواقع السورية للآليات الأمنية من جهة الزبون وكذلك تقديم نموذج مرجعي للحالة الحالية لأمن الشبكة.

1.1 هدف البحث:

يهدف هذا البحث إلى:

- التعرّف على التقنيات والأدوات لمواجهة هذه المخاطر والمطروحة من قبل آخر الدراسات في المجال. ثم اقتراح الممارسات الأفضل بينها.
- برمجة أداة تقوم بإجراء اختبارات لكشف مدى تطبيق المواقع السورية لهذه التقنيات.
- اقتراح نظام تقييم تستخدم نتائجه كمؤشر على مدى الوعي الأمني لهذه المواقع.
- الوصول الى نتائج وملاحظات تخدم في زيادة أمن المواقع السورية لمواجهة الهجمات التي تستغل الثغرات من جهة الزبون.

1.2 دراسات وأعمال سابقة:

تم إجراء العديد من التقييمات حول وجود ثغرات أمنية محددة في تطبيقات الويب. على سبيل المثال، يقوم **WhiteHat Security**، بتقييم سنوي للبيانات المتعلقة بعدة أنواع من نقاط الضعف التي يجمعونها من زبائنهم [9]. على النقيض من بحثنا، فإن التقييم يجري بموافقة عملائهم وبالتالي فهو أكثر عدوانية وفاعلية، وهو ما يمكّنهم من العثور على أنواع إضافية من الثغرات الأمنية، مثل أخطاء **SQL Injections** و**XSS**.

البحث الذي قدمته جامعة **KU Leuven** [10] والذي استقينا منه الكثير من الارشادات، يحلل ما مجموعه 22,000 موقع أوروبي من 28 دولة أوروبية. ويدرس مدى أمان هذه المواقع.

البحث الذي يقدمه العريفي وزميله [11] يقيم أمن المواقع العربية الأكثر شعبيةً. يستكشف تحليلهم وجود صفحات مخادعة وبرامج ضارة في 7000 نطاق. لاكتشاف البرامج النصية الضارة المستضافة على صفحات الويب، قاما باستخدام الواجهات البرمجية التي تقدمها الشركات الأمنية المعروفة لفحص المواقع.

نيكيفوراكيس وآخرون قدموا تحليلًا واسع النطاق لتضمين ملفات **JavaScript** خارجية [12] . إضافة إلى ذلك، اقترحوا في مقالتهم أيضاً مقياسًا يسمى جودة الصيانة (**QoM**) لوصف الوعي الأمني لموقع الويب. تعتمد **QoM** الخاصة بهم العديد من الميزات مثل خاصية **HttpOnly** وترويسة **X-Frame-Options**، والتي يتم تضمينها أيضاً في تقييمنا. كما نوقش سابقًا، فإن وجود هذه الآليات الدفاعية يدل على أمان موقع الويب.

وجد **Vasek** و**Moore** [13] أن بعض ميزات موقع الويب، مثل برنامج المخدم ونظام إدارة المحتوى **CMS** المستخدم، يمكن أن تكون بمثابة عوامل خطر إيجابية لتهديد المخدمات. توضح دراستهم أن بعض أنواع المخدمات وأنواع **CMS** أكثر خطورة من غيرها (على سبيل المثال، من المرجح أن تكون الخوادم التي تشغل **Apache** و**Ngnix** معرضة للخطر أكثر من الخوادم التي تشغّل **Microsoft IIS**).

أما في المجال المحلي فإن الهيئة الوطنية لإدارة الشبكة [14] تقوم بين سنة وأخرى بإصدار تقارير مسح مجانية للمواقع الحكومية السورية. لكنها تركز في عملها على أمن المخدم وليس على أمن المستخدم. حتى أنه وللأسف الموقع التابع للهيئة لا يستخدم طبقة النقل الآمنة **https** حتى لصفحات تسجيل المستخدم وتسجيل الدخول. والذي يعتبر المعيار الأكثر ثقلا في تقييم أمان موقع ما.

1.3 لمحة عامة عن محتوى الرسالة:

في هذه الأطروحة، نقارب أمان الويب من جهة الزبون، وهو مجال مهم من الأبحاث التي ظهرت منذ منتصف العقد الأول من القرن العشرين. محتوى هذه الرسالة يؤكد على التطور باتجاه الآليات الأمنية من جهة الزبون.

تندرج الأطروحة في ثمانية فصول كما يلي:

الفصل الأول:

مقدمة عامة للبحث وأهميته

الفصل الثاني:

نظرة عامة لمجال أمن الشبكة العنكبوتية من جهة الزبون، الذي يحتوي على العديد من الثغرات الأمنية وتقنيات التخفيف من مخاطرها والمبنية فوق السياسات الأمنية الأساسية للشبكة.

كما نقدم السياق اللازم لفهم مجال أمان الويب من جهة الزبون تمامًا. نناقش اللّبنات الأساسية ذات الصلة لتطبيقات الويب الحديثة ونحدد إمكانات التهديد ذات الصلة.

في الفصول المتتالية سنركز بشكل صريح على الثغرات الأمنية من جهة الزبون، والتي يتم استغلالها من داخل المستعرض أو استهداف المستعرض بشكل صريح لأنها تتلقى عموماً اهتماماً أقل مقارنةً بنظرائهم من جانب المخدم. إجمالاً، نحن نغطي 13 هجوماً، نقدم لها وصفاً تفصيلياً، ونظرة عامة على تقنيات التخفيف التقليدية، وأحدث الأبحاث الحالية. لكل هجوم، نصف أيضاً الحالة الحالية للممارسات في تطبيقات الويب، ونحدد أفضل الممارسات للدفاع ضد هذه الهجمات. قمنا بتصنيف الهجمات بحسب قربها من المستخدم وذلك وفقاً لـ [15]:

الفصل الثالث:

يغطي الهجمات على مستوى الشبكة

الفصل الرابع:

يغطي الهجمات التي تستهدف الطلبات الصادرة من المستعرض

الفصل الخامس:

يغطي الهجمات التي تستهدف جلسة عمل المستخدم ضمن المستعرض

الفصل السادس:

يغطي الهجمات التي تستهدف سياق عمل التطبيق ضمن المستعرض

الفصل السابع:

يغطي الهجمات التي تستهدف جهاز المستخدم أو تهدد المستعرض بأكمله

الفصل الثامن والأخير:

يغطي تفاصيل الدراسة العملية التي أجريناها لتقييم الواقع الأمني للمواقع السورية وتفاصيل نظام التقييم المقترح ونبين النتائج التي توصلنا لها، وأخيرا نذكر تحديات البحث والدراسات السابقة في المجال.

الفصل الثاني

لمحة عن تطور الشبكة وأهمية الأمن من جهة الزبون

2 تطور الشبكة وأهمية الأمن من جهة الزبون

قبل البدء بمناقشة الهجمات والإجراءات المضادة لها سنلقي نظرة على التحولات التي طرأت على الشبكة ولماذا أصبح أمن الشبكة من جهة الزبون (وهو الموضوع الأساسي لهذه الأطروحة) مهماً.

سنقدم في هذا الفصل المعلومات الضرورية لتأسيس فهم عام لتطبيقات الويب. هذا المعلومات موجهة بشكل خاص باتجاه أمن التطبيقات من جهة الزبون مع التركيز على أمن الجلسات.

2.1 لمحة عن الشبكة:

الشبكة العالمية بدأت كنظام موزع للنصوص المتشعبة، حيث تتم استضافة مستندات على حاسبات موصولة إلى شبكة واحدة هذه المستندات تحوي مؤشرات مرجعية لمستندات أخرى موضوعة على حاسبات أخرى. هذه المستندات يمكن الوصول إليها باستخدام مستعرض وهو برمجية مخصصة لاستعراض مستندات النصوص التشعبية ومتابعة الارتباطات التشعبية المضمنة في نصوص هذه المستندات.

حتى يعمل هذا النظام كان لا بد من وجود ثلاثة معايير أساسية:

1. معرفات المصادر **Uniform Resource Identifier (URI)** وتؤمن معرفات مميزة عالميا للمصادر على الشبكة
2. بروتوكول النقل **HTTP (Hypertext Transfer Protocol)** بروتوكول نقل مدعوم عالمياً، يقدم آلية بسيطة لاستعادة المصادر عبر الشبكة وإرسال البيانات من المستعرض إلى المخدم.
3. صيغة المحتوى **HTML** صيغة مطبقة بشكل واسع بدأت كلغة بسيطة يمكن كتابتها باستخدام محرر نصوص بسيط وهذا ما زال قائما حتى الآن بعد أكثر من عشرين عام.

في بداياتها كانت الشبكة غير تفاعلية ولا حالة لها من جهة الزبون. ومع بساطة هذه الخصائص إلا أنها كانت كافية لتحقيق الغاية الأساسية للشبكة العالمية إلا أنها لم تعد كافية لتلبية الحاجة لبيئة تطبيقات غنية.

من دون شك جافا سكريبت هي التقنية الأكبر أثراً على تطور الشبكة. كان القصد منها التلاعب بصفحات الويب ضمن المستعرض باستخدام الـ**DOM (Document Object Model)** ، إلا أنها وبسرعة أثبتت أنها أكثر قوة من ذلك، مما جعلها اللغة الافتراضية للبرمجة من جهة الزبون.

إحدى عوامل نجاح الجافا سكريبت هو تقنية **AJAX** المبنية على أساس جافا سكريبت و**XML**، حيث تمكننا من إجراء عمليات غير متزامنة. باستخدام **AJAX** تستطيع صفحات الويب تخزين واستعادة البيانات بشكل مخفي وإجراء التعديلات على الصفحة تلقائياً. العديد من التطبيقات الحديثة ما زالت تعتمد على هذه التقنية مع أنه تم استبدال الـ**XML** بـ**JSON** وهي صيغة لتمثيل الأغراض في جافا سكريبت.

أحد التطورات التقنية أيضاً هو أنماط المحتويات الغنية المتواجدة على الشبكة حالياً. فالمحتوى لم يعد مقتصراً على صفحات **HTML** وصور، حيث تدعم المستعرضات الحديثة عدة صيغ للصوت والصورة ولغات تعريف الصور المبنية على تقنية **XML** مثل **SVG** والبيانات العلمية **MathML**.

أخيراً، العامل الأخير للتطبيقات الغنية هو حالة الزبون ضمن المستعرض. كمقاربة أولى تم استخدام الوسطاء المشفرة **encoded parameters** ضمن عنوان الـ**URI** وهو حل غير عملي تم استبداله سريعاً بملفات تعريف الارتباط **cookies** وهي زوج من (مفتاح–قيمة) يتم إصدارها من قبل المخدم وتخزينها من قبل المستعرض وربطها مع كل طلب صادر. حالياً ملفات تعريف الارتباط مستخدمة من قبل كل التطبيقات تقريباً، لكن سيئتها أنه يتم إرسالها مع كل طلب مما يجعلها غير مناسبة لتخزين بيانات ذات حجم كبير. لذلك العديد من الحزم البرمجية تقدم تخزين من جهة الزبون بصيغة أزواج من (مفتاح–قيمة) [16] أو قاعدة بيانات للأغراض[17] أو نظام ملفات افتراضي[18].

هذه التغييرات الثلاثة الأساسية أطلقت شرارة التحول من تبادل للبيانات باتجاه وحيد إلى شبكة كتابة/قراءة ثنائية الاتجاه أصبحت تعرف بـ **Web 2.0**. هذه المرحلة الجديدة من تطور الشبكة تجمع التطورات التقنية مع النواحي الاجتماعية للمستخدمين المشاركين بشكل فعال في التطبيق مما أنتج تطبيقات ديناميكية تتطور بشكل فاعل مع الزيادة في عدد مستخدميها. أمثلة على هذه التطبيقات هي ويكيبيديا وفايسبوك وخدمات غوغل المتعددة. تكثّف أثر العامل الاجتماعي مع بدء الناس بحمل الأجهزة الذكية المتصلة بشكل دائم إلى كل مكان يذهبون إليه. هذه الأجهزة تؤمن الوصول الدائم والفوري للمعلومات مما حفز ظهور الخدمات المتعلقة بالموقع والمرتبطة بالحالة.

مع تطور المستعرض إلى بيئة متكاملة تحول تركيز المسؤوليات ليتم دفعه باتجاه الزبون. لم تعد مهمة الزبون فقط عرض نتيجة عمل التطبيق الذي يعمل في الخلف لكن أصبح هو التطبيق الذي يتفاعل مع تطبيق خلفي خفيف مهمته الرئيسة عملية التخزين وذلك عبر واجهة برمجية غنية.

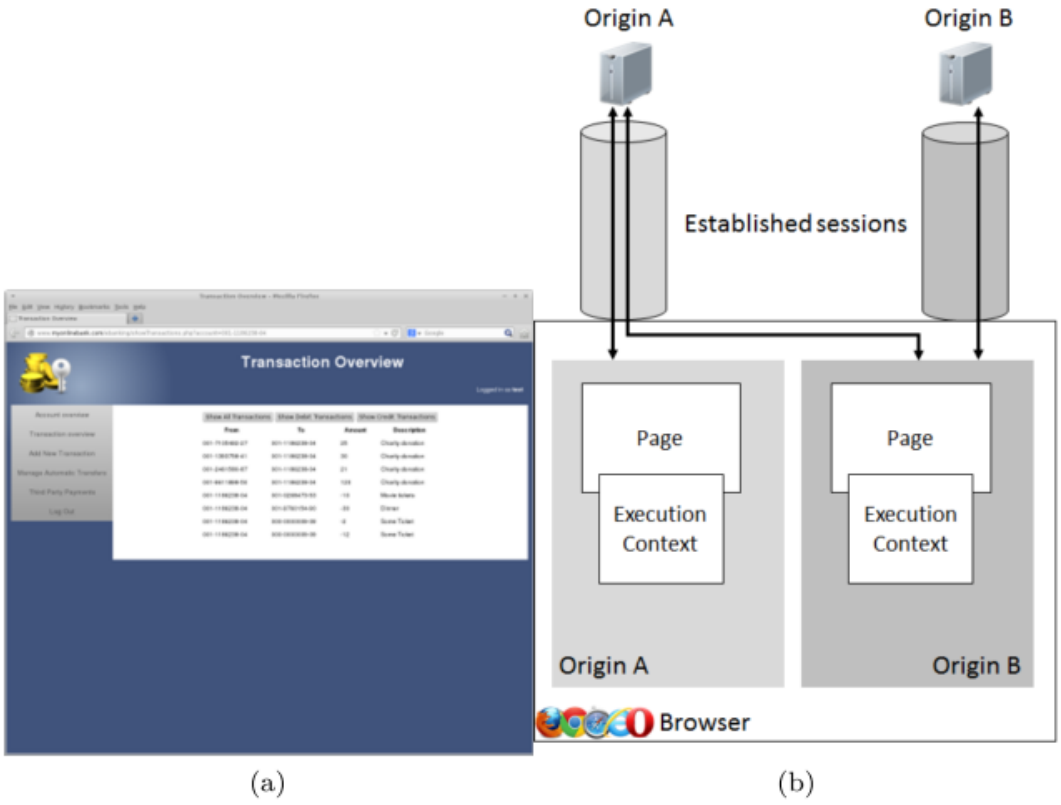
التقنيات التي تقف خلف هذا التطور هي مجموعة تقنيات خفيفة من جهة المخدم مثل (**NodeJS** و**Go**) ومكتبات من جهة الزبون مثل (**JQuery** و**Bootstrap**) وأطر عمل من جهة الزبون مثل (**AngularJS** و **Ember.js**). بالإضافة إلى أن المستعرضات تقدم ميزات غنية ومتعددة للتطبيقات مثل النفاذ لآليات التخزين في جهة الزبون والنفاذ لمعلومات عن الجهاز المستخدم وحساساته والعديد من آليات الاتصال.

مما شجع هذا التحول أيضاً ظهور الأجهزة النقالة مع أنظمة تشغيلها المقيدة ومخازن تطبيقاتها المتحكم بها من قبل الموزع. حيث ليس فقط أغلب تطبيقات اليوم الموجودة في مخازن التطبيقات مبنية على أساس تقنية الويب ولكن أيضاً هنالك جهود لفرض معايير[8] تؤمن الواجهات البرمجية الضرورية لبناء تطبيقات يمكن أن تتفاعل مع الآلة المضيفة مما يجعلها غير قابلة للتمييز عن التطبيقات المحلية (الموجودة مع نظام التشغيل).

لقد شهدت الشبكة العديد من الخطوات التطورية حولتها من محتوى ساكن من جهة المخدم إلى تطبيقات ديناميكية من جهة الزبون، وحولت المستعرضات من تطبيقات مملة لعرض الصفحات إلى بيئات تنفيذ تقوم بتشغيل تطبيقات قوية وعالية الديناميكية، مما أدى إلى تحول أمن الشبكة من جهة المخدم إلى جهة الزبون.

2.2 اللّبنات الأساسية لتطبيق الويب:

التطبيقات الحديثة مبنية ضمن بيئات ديناميكية معقدة، متألفة من العديد من المكونات والسياسات الأمنية المتفاعلة مع بعضها ولكن أيضاً مع وجود مصاعب ومآزق. هذا القسم يغطي اللّبنات الأساسية لتطبيق الويب والمتعلقة بأمن التطبيق من جهة الزبون. في الشكل الآتي لقطة لنظام بنكي غير آمن الصورة تظهر تطبيقاً عادياً لكن في الواقع هنالك العديد من خصائص المستعرض والمكونات التي تمكن من الحصول على تطبيق تفاعلي ومعقد. المكونات الموضحة في الشكل هي فقط جزء من هذه المكونات. كمثال على المكونات الأخرى إضافات المستعرض والـ **plugins** وخيوط تنفيذ الجافا سكريبت...



شكل 2-1:

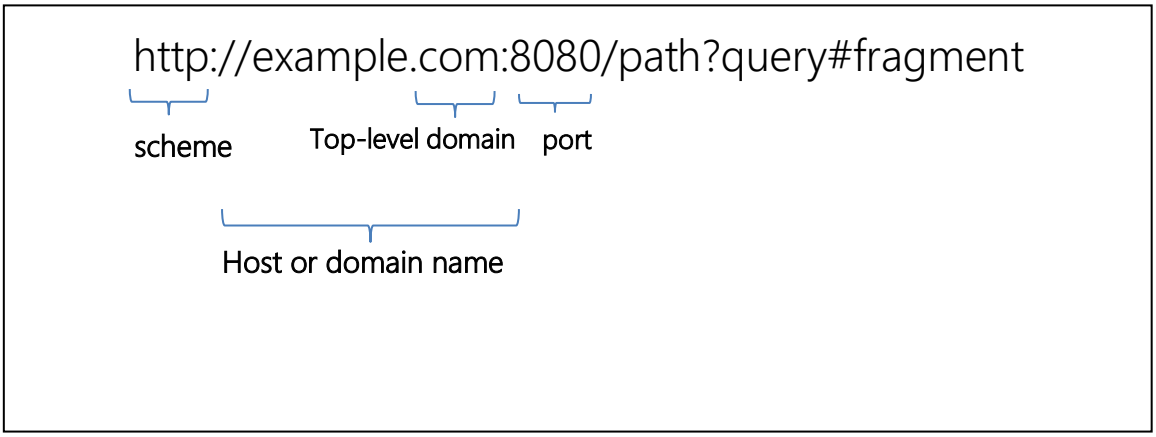
*(a) لقطة شاشة (b)منظور هيكلي للتطبيق المصرفي غير الأمن. نلاحظ بالفعل أنه في الشكل الأيمن، يتواصل السياق ذو الأصل **B** مع الأصل **A** باستخدام جلسة عمل تم إنشاؤها بالفعل، وهو مصدر للعديد من مشكلات أمان الويب.*

ضمن المستعرض الـ "نافذة **window**" هي حاوية تحتوي كل بيانات التطبيق كالصفحة المعروضة حالياً وسياق تنفيذ للجافا سكريبت. بالإضافة لذلك تؤمن النفاذ للميزات المقدمة من المستعرض مثل الـ **cookie jar** والواجهات البرمجية المتعددة. بينما نافذة واحدة تحمل كمية كافية من التعقيد إلا أن المستعرضات لا تكتفي بعرض العديد من النوافذ في نفس الوقت بجوار بعضها كألسنة أو نوافذ مستعرض مستقلة فقط إلا أنها تدعم التعشيش الديناميكي للنوافذ باستخدام **frame** أو**iframe**. سنناقش الخصائص التفصيلية لهذه المكونات وتفاعلاتها بتفصيل أكبر.

2.2.1 سياق التنفيذ الخاص بالزبون:

يتحكم بالسلوك الديناميكي للتطبيق ضمن المستعرض. سياق التنفيذ يقوم بتشغيل كود الجافا سكريبت للتطبيق والذي له صلاحية النفاذ للبنية الشجرية للصفحة ومحتوياتها عبر الـ **Document Object Model (DOM)** وكذلك الميزات الخاصة بالمستعرض وواجهاته البرمجية التي تمكن من الإبحار والاتصال البعيد

والتفاعل مع جهاز الزبون وتخزين البيانات عند الزبون. سياق التنفيذ له أصل محدد **(origin)** يعرف بالثلاثية: (مخطط **scheme** + مضيف **host** + منفذ **port**) وهذه المعلومات مستخلصة من عنوان الصفحة المرتبطة بالسياق **URI**.



شكل 2-2 أجزاء عنوان صفحة الويب

مع نضوج الويب التقنيات مثل الإطارات والألسنة والنوافذ المنبثقة سمحت بالتنفيذ المتزامن لعدة تطبيقات كل ضمن سياقه الخاص. يمكن لهذه السياقات التواصل فيما بينها إلى حد ما ضمن المستعرض، وكذلك مع المخدّمات البعيدة.

السياسة الأمنية المحورية في المستعرضات هي سياسة الأصل الواحد **Same-Origin Policy (SOP)** هذه السياسة تفرض قيوداً على هذا التواصل لمنع سياقات التنفيذ ذات الأصول المختلفة من التأثير على بعضها البعض. كنتيجة لتلك السياسة فإن التطبيقات المختلفة يمكن لها التواجد معا في نفس المستعرض بوجود ضمان العزل والسرية بينها.

مع تقديم ميزات إضافية من قبل المستعرضات زادت أهمية هذه السياسة، على سبيل المثال فإن الواجهات البرمجية الحديثة التي تؤمن القيام بعمليات التخزين تستخدم حاويات تخزين مرتبطة بالأصل **origin-specific**. وبشكل مشابه العديد من الواجهات البرمجية تطلب إذن صريح من المستخدم قبل الإفصاح عن بيانات أو ميزات حساسة ومنحها للسياق، هذه الأذونات مرتبطة بأصل السياق الذي طلبها. مثل مشاركة موقع الجهاز أو التقاط الصوت والصورة.

بينما تقوم سياسة الأصل الواحد بفصل سياقات التنفيذ عن بعضها البعض لا يوجد سياسة فصل تدعم عزل جزء من الكود ضمن سياق التنفيذ. الاستخدام الأساسي لمثل هذه السياسة هو عند تضمين ملفات جافا سكريبت خارجية وهذا شائع جدا في التطبيقات الحديثة حيث تقوم بتضمين مكتبات واعلانات وبريمجات تحليلية [12]. مع تضمين هذا الكود الخارجي ضمن سياق الصفحة من دون أي حدود فإنه يصبح قادرا على

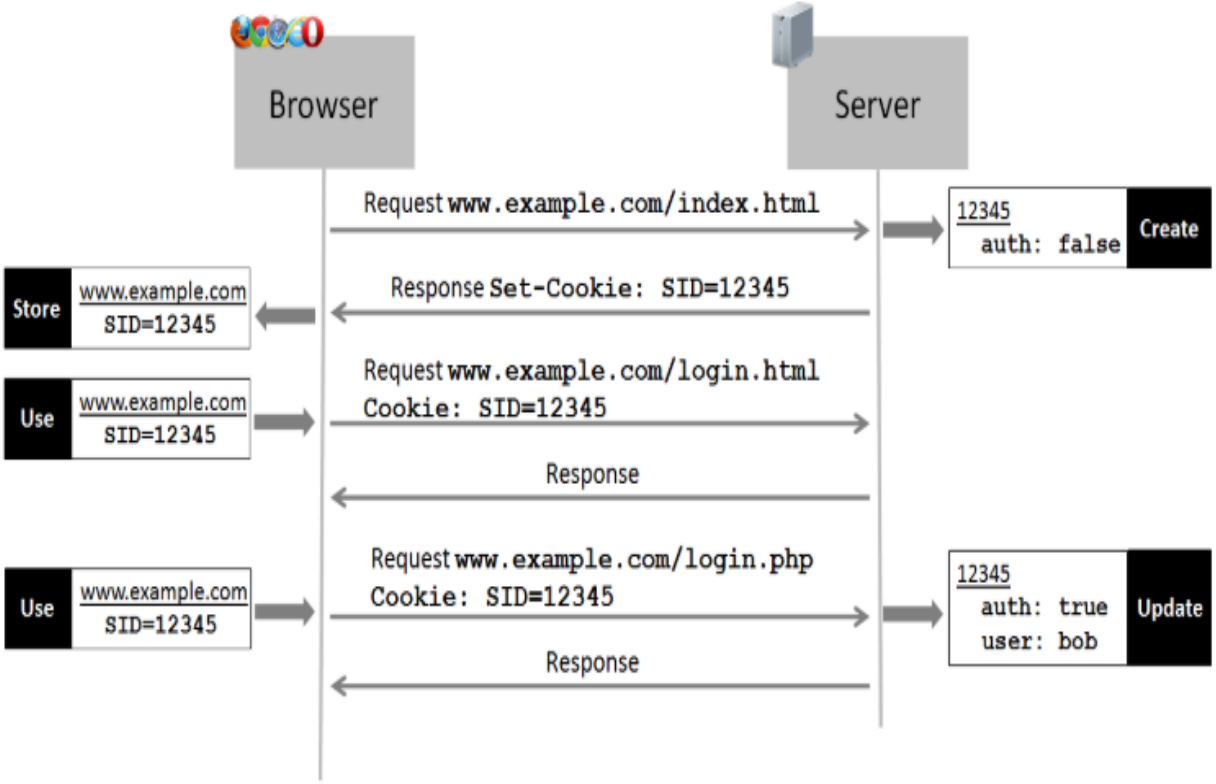
الوصول إلى كل الميزات المرتبطة بأصل الصفحة. هذا يفرض وجود علاقة ثقة قوية بين الأصل والطرف المزود لهذا الكود وهذه الثقة يسهل خرقها مثلاً إذا قام مهاجم ما باختراق الطرف المزود [6].

2.2.2 إدارة الجلسات:

في بدايات الشبكة، عندما كانت المواقع تتألف فقط من محتوى ساكن، قدم بروتوكول **HTTP** ترويسة مخصصة لتوثيق المستخدم "**Authorization Header**" [19]، وما زالت هذه الترويسة مستخدمة حتى هذا اليوم. الاستخدام الأكثر شيوعاً لهذه الترويسة هو للمصادقة الأساسية، حيث يتم تشفير بيانات اعتماد المستخدم، وبشكل أكثر تحديدًا اسم المستخدم وكلمة المرور، ويتم تضمينها كقيمة للترويسة. يقوم التطبيق من جانب المخدم باستخراج بيانات الاعتماد هذه والتحقق منها واتخاذ قرار بشأن السماح بالطلب أم لا. بعد مصادقة ناجحة، سيقوم المتصفح بإرفاق بيانات اعتماد المستخدم لكل طلب لاحق لهذا الأصل. نظرًا لأن المستعرض يتذكر بيانات الاعتماد هذه خلال فترة تشغيله، فإن تسجيل الخروج من حساب ممكن فقط عن طريق إغلاق المستعرض.

نظرًا لأن تطبيقات الويب أصبحت أكثر تعقيدًا، فقد أراد المطورون دمج المصادقة مع التطبيق، مما أدى إلى تبسيط تجربة المستخدم مع استمرار نفس الشكل والمظهر. لمصادقة المستخدمين، قاموا بتضمين نموذج إدخال **HTML**، حيث كان على المستخدم إدخال اسم مستخدم وكلمة مرور. عن طريق إرسال بيانات النموذج، تم إرسال اسم المستخدم وكلمة المرور إلى المخدم، حيث يمكن التحقق من صحتها. ومع ذلك، نظرًا لأن بيانات الاعتماد لا يتم إرسالها إلا في طلب واحد عند إرسال النموذج، بدلاً من كل طلب لاحق كما هو الحال مع ترويسة المصادقة، فإن تطبيقات الويب تحتاج إلى وسيلة لتتبع حالة مصادقة المستخدم عبر الطلبات، وهو تحدّ مع بروتوكول **HTTP** عديم الحالة.

لتطبيق آلية لإدارة الجلسة قادرة على ربط طلبات متعددة من نفس المستخدم مع حالة جلسة من جانب المخدم، مما يسمح للتطبيق بتخزين معلومات الجلسة، مثل حالة المصادقة أو عربة التسوق (كما في الشكل الآتي).



شكل 2-3 إدارة الجلسة باستخدام ملفات تعريف الارتباط

لتتبع جلسة العمل هذه من جانب المخدم عبر طلبات متعددة يتم تعيين معرّف لجلسة العمل، والذي يحتاج المستعرض لتضمينه في كل طلب. في البداية، تم تضمين هذا المعرف في عنوان الصفحة **URI** كوسيط، على سبيل المثال:

http://example.com/index.html?SID=1234

لسوء الحظ، يتطلب هذا من تطبيق الويب إلحاق معرف جلسة المستخدم بكل عنوان في الصفحة، وتكرار هذا الإجراء لكل مستخدم. ظهور ملفات تعريف الارتباط **Cookies** [20] عالج هذه المشكلة. ملفات تعريف الارتباط هي أزواج ذات قيمة موفرة من قبل المخدم، يتم تخزينها من قبل المستعرض في جرة ملف تعريف الارتباط "**cookie jar**" ويتم إرفاقها بكل طلب موجه لذات النطاق. بشكل أساسي، تسمح ملفات تعريف الارتباط للمخدم بتخزين أجزاء صغيرة من البيانات من جانب المستعرض، والتي سيتم إرفاقها بالطلبات المستقبلية. يمكن استخدام ملفات تعريف الارتباط لتخزين الإعدادات البسيطة، مثل تفضيلات اللغة، أو لإنشاء أنظمة أكثر تعقيدًا، مثل آليات إدارة الجلسة. اليوم، تعد إدارة الجلسة التي تعتمد على ملفات تعريف الارتباط هي المعيار الواقعي، لكن العديد من الأنظمة لا تزال تسمح بإدارة الجلسة المستندة إلى الوسطاء

كآلية احتياطية، على سبيل المثال عندما لا يتم دعم ملفات تعريف الارتباط من قبل المستعرض، أو تعطيلها من قبل المستخدم.

2.2.3 التفاعلات بين الأصول المختلفة:

مع أن سياسة الـ **SOP** تمنع بنجاح تفاعل السياقات مختلفة الأصول من التأثير ببعضها إلا أنه يوجد عدة أنماط للتفاعل موجودة في الويب، أو يتم تفعيلها بشكل مقصود.

- مثال أول:

التفاعل المحلي بين السياقات المختلفة: وهي خاصية مطلوبة بشدة من قبل مطوري تطبيقات الـ **mashup**..حاليا معيار تبادل الرسائل **Web Messaging** [21] يقدم آلية للتفاعل المدروس بين السياقات المختلفة ومدعوم من قبل جميع المستعرضات الحديثة.

- مثال ثاني:

تفاعل ضمني... غير صريح: عندما الصفحة ذات الأصل **A** ترسل طلب لمخدم في الأصل **B** مثلاً لتضمين صورة او سكريبت من **B**. سيقوم المستعرض بإرسال جميع ملفات الـ **cookies** المخزنة لديه والتي تخص **B** مما يسمح للطلب الخارج عن السياق أن يصبح جزءا من الجلسة مع **B**.

هذا النوع من التفاعل أساسي جدا للعديد من التفاعلات على الويب ... زر الإعجاب لفيسبوك مثال على ذلك حيث يمكن لأي صفحة تضمين هذا الزر وعندما يقوم المستخدم بضغطه سيتم إرسال بياناته إلى فيسبوك. لسوء الحظ، مع أن هذا النمط من التفاعل ضروري وشائع إلا أنه وفي الوقت ذاته يشكل الأساس لهجمات تزوير الطلبات العابرة للموقع والتي سنتحدث عنها لاحقا.

- مثال أخير:

التعقيدات التي تفرضها التفاعلات الصريحة التي يقوم بها سكريبت ما ضمن سياق الصفحة: التواصل التقليدي في السكريبت يكون باستخدام غرض **XMLHttpRequest** لإرسال الطلبات وتم حصره بالتواصل ضمن نفس الأصل. ولكن كنتيجة لهذا القيد، ظهرت حلول جديدة لتجاوز قيود الأصل الواحد باستخدام وسم **<script>** بالإضافة لـ**json** وحشو إضافي. هذه التقنية والتي تدعى **JSON-P** تقوم بتضمين ملفات جافا سكريبت ديناميكيا والتي بدورها تقوم باستدعاء تابع عودي **callback function** ضمن سياق التنفيذ المحلي. يعمل هذا الإجراء بشكل فعال على تمكين التواصل بين الأصول المختلفة، حيث يمكن للمتصفح إرسال الوسطاء في طلب السكريبت،

ويستجيب المخدم بالبيانات المطلوبة. لسوء الحظ، يؤدي هذا أيضاً إلى ظهور ثغرة أمنية شديدة، نظرًا لأن المخدم يمكنه عن طريق الحقن أو عن غير قصد ضخ محتوى في سياق تنفيذ الصفحة. تفادياً لهذه الطريقة الخطيرة، تم تعديل معيار الـ **XMLHttpRequest** ليجعل الاتصال عبر الأصل ممكنًا بشكل صريح من خلال تطبيق مشاركة المصادر المشتركة **Cross-Origin Resource Sharing (CORS)** [22]. يسمح **CORS** للمخدمات بتزويد المستعرض بسياسة أمان تسمح صراحةً بإرسال تفاعلات معينة إلى أصل مختلف من سياق تنفيذ محلي. نظرًا لأن **CORS** هي سياسة اختيارية، فلن تصبح المخدمات القديمة عرضة لهجمات جديدة.

2.2.4 الطلبات والردود في الشبكة:

نظرًا لأن الويب نظام موزع، فإنه يعتمد على البنية الأساسية للشبكة وبروتوكولات الاتصال المرتبطة بها. على الرغم من أن تفاصيل **HTTP**، بروتوكول الاتصال الفعلي على الويب، مغلفة بشكل جيد، إلا انه يحوي العديد من الخصائص والتعقيدات التي تؤثر على جهود نشر أمان الويب.

إن أول تأثير لاستخدام بروتوكول **HTTP**، يأتي من طبيعة نقل البيانات بشكل نصي صريح. بدون حماية إضافية، فإن جميع البيانات المرسلة إلى الأصل باستخدام مخطط **http** تكون غير محمية بمجرد أن تغادر المستعرض، مما يتيح العديد من الهجمات على الشبكة، مثل التنصت أو الرجل في المنتصف. يمكن أن يكون لهذه الهجمات عواقب وخيمة، حيث يتم عادةً نقل بيانات تعريف المستخدم ومعرفات الجلسة والمعلومات الشخصية من المستعرض إلى المخدم. تفادياً لهذه الممارسة غير الآمنة، تم تقديم **HTTPS**، وهو ما يعني تشغيل بروتوكول **HTTP** عبر قناة مؤمنة بواسطة **Transport Layer Security (TLS)**، والمعروفة مسبقًا باسم **Secure Sockets Layer (SSL)**. هذه القناة تقدم المصداقية والسرية وسلامة البيانات عبر القناة، وبالتالي منع الهجمات على مستوى الشبكة. على الرغم من أن اتصال **TLS** يكون في طبقة الشبكة، إلا أنه يؤثر على العديد من المفاهيم الهامة مثل خاصية **secure** لملفات تعريف الارتباط.

هنالك نمط شائع لاستخدام **https** يتمثل بتقديم التطبيق عبر **http** والانتقال إلى **https** فقط لنقل الأجزاء الحساسة من التطبيق مثل نقل بيانات بطاقة الدفع الالكتروني أو كلمات سر المستخدمين ثم العودة إلى **http** بعد انتهاء النقل. هذا السيناريو يعتبر خطراً وغير مبرر حيث أنه يترك التطبيق عرضة لهجمات مثل اختطاف الجلسة.

النمط الثاني الشائع يتمثل باستخدام المحتوى المختلط، حيث يتم تحميل الصفحة عبر **https** مما يمنع العبث بها، لكن في نفس الوقت يتم تحميل موارد إضافية مثل ملفات جافا سكريبت عبر اتصال عادي غير آمن مما يجعلها عرضة للعبث ويهدد أمن الصفحة كاملة.

2.3 أهمية الأمن من جهة الزبون:

في بدايات الشبكة كان تركيز المهاجمين على الخدمات من جهة المخدم محاولين استغلالها أو السيطرة على جهاز المخدم. أمثلة على هذه الهجمات حقن الـ**SQL** (**SQL injection**) وحقن الأوامر (**command injection**) واستغلال ثغرات ملء الذاكرة المؤقتة (**buffer overflow**).

مع ازدياد قدرات ومهام المستعرضات بدأ المهاجمون باستهداف أجهزة الزبائن محاولين استغلال نقاط الضعف لتحميل برمجيات خبيثة، مثلاً للنفاذ للحساب البنكي للضحية. وبسبب ندرة نقاط الضعف في نظام التخزين للمستعرضات وإضافاتها تحول تركيز المهاجمين إلى النقاط الأضعف. الهجمات مثل البرمجة العابرة للموقع **cross-site scripting** وتزوير الطلبات العابر للموقع **cross-site request forgery** تستخدم المستعرض كوسيلة للقيام بطلبات إلى المخدم باسم الضحية.

الجدول يوضح المخاطر الأمنية العشرة الأكثر خطورة بالنسبة لإحصائيات **OWASP** لعام 2017:

1	INJECTION
2	BROKEN AUTHENTICATION
3	SENSITIVE DATA EXPOSURE
4	XML EXTERNAL ENTITIES (XXE)
5	BROKEN ACCESS CONTROL
6	SECURITY MISCONFIGURATION
7	CROSS-SITE SCRIPTING (XSS)
8	INSECURE DESERIALIZATION
9	USING COMPONENTS WITH KNOWN VULNERABILITIES
10	INSUFFICIENT LOGGING & MONITORING

جدول 2-1

*قائمة **OWASP** للثغرات الأمنية العشرة الأكثر خطورة لعام 2017*

الإحصائيات التي تقوم بها الشركات الأمنية الكبرى مثل **OWASP** [23] و**CWE/SANS** [24] توضح هذا التحول نحو الهجمات من جهة الزبون حيث تخصص حوالي ثلث الهجمات الأخطر للهجمات المنفذة من جهة الزبون.

1	Improper Restriction of Operations within the Bounds of a Memory Buffer
2	IMPROPER NEUTRALIZATION OF INPUT DURING WEB PAGE GENERATION ('CROSS-SITE SCRIPTING')
3	Improper Input Validation
4	Information Exposure
5	Out-of-bounds Read
6	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
7	Use After Free
8	Integer Overflow or Wraparound
9	CROSS-SITE REQUEST FORGERY (CSRF)
10	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
11	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
12	Out-of-bounds Write
13	Improper Authentication
14	NULL Pointer Dereference
15	Incorrect Permission Assignment for Critical Resource
16	Unrestricted Upload of File with Dangerous Type
17	Improper Restriction of XML External Entity Reference
18	Improper Control of Generation of Code ('Code Injection')
19	USE OF HARD-CODED CREDENTIALS
20	UNCONTROLLED RESOURCE CONSUMPTION
21	Missing Release of Resource after Effective Lifetime
22	Untrusted Search Path
23	Deserialization of Untrusted Data
24	Improper Privilege Management
25	IMPROPER CERTIFICATE VALIDATION

جدول 2-2

*قائمة **CWE/SANS** للأخطاء البرمجية الـ25 الأكثر خطورة*

من الطبيعي عندما يتحول تركيز المهاجمين باتجاه الزبون أن تتطور السياسات والإجراءات الأمنية كذلك. وهذا التطور أيضاً يوافق التطورات التي تشهدها الشبكة.

السياسات الأمنية الأولى كانت ساكنة ومضمنة كسلوك افتراضي في المستعرض وتطبق نفس القواعد على كل التطبيقات. كمثال على هذه السياسات الساكنة سياستي الأصل الواحد **Same-Origin Policy(SOP)** والسلوك الموافق لسياسة الأصل الواحد لغرض الـ **XMLHttpRequest** كما رأينا سابقاَ. بعدها أتت السياسات الديناميكية المنفذة بشكل أساسي من جهة المخدم. من الأمثلة التقليدية هي الحماية التقليدية المبنية على أساس استخدام **tokens** أو المبنية على ترويسة الطلب **request header** للحماية من تزوير الطلب العابر للموقع، أو الحمايات المعتمدة على التحقق **validation based** للحماية من البرمجة العابرة للموقع. هذه السياسات تنسجم مع ظهور التطبيقات الديناميكية والتي لها مكونات معالجة من جهة المخدم. الموجة التالية من الآليات الأمنية تستغلّ تحوّل المستعرض إلى بيئة تطبيقات متكاملة. السياسات الأمنية الحديثة منفذة من جهة الزبون ولكن مقادة من قبل التطبيق في جهة المخدم "**server-driven client-enforced**". أحد الأمثلة الأساسية هي السياسة الأمنية للمحتوى **Content Security Policy(CSP)** [25] وهي سياسة مرتبطة بالتطبيق ويقوم المستعرض بفرضها وتطبيقها. السياسات المقادة من المخدم والمنفذة من الزبون غالبا ما تستخدم ضمن استراتيجيات حماية متعددة الطبقات حيث ينفذ كل من المخدم والزبون سياسة أمنية في محاولة جاهدة لإيقاف المهاجمين حتى عندما يتمكنون من تجاوز إحدى الإجراءات المطبقة.

2.4 التوجهات الحديثة للسياسات الأمنية:

أحد التوجهات المهمة في أمن التطبيقات من جهة الزبون هي بروز السياسات الأمنية المقادة من قبل المخدم والمنفذة من قبل الزبون. هذه السياسات تطورت ضمن خط البحث العلمي في تقنيات المواجهة المستقلة من

جهة الزبون. هذه السياسات تقدم من قبل المخدم حيث تكون الخصائص الأمنية المطلوبة محددة وبنية التطبيق أيضاً معروفة، ويتم تطبيق هذه السياسة من قبل المستعرض والذي يشكل المكان الأمثل لتطبيق السياسات الأمنية عند الزبون.

هذه السياسات ظهرت مع التطور المستمر للعملية الأمنية التي بدأت بالسياسات الأمنية العامة للمستعرضات مثل سياسة الأصل الواحد **SOP** المطبقة على جميع التطبيقات من دون استثناء أو تخصيص. مع تطور قدرات المخدمات تم تطبيق الاجراءات الأمنية عند المخدم. كمثال على ذلك مواجهة هجمات تزوير الطلبات باستخدام الرموز **token-based** أو تعقيم الدخل والخرج لمواجهة هجمات الحقن. هذه الإجراءات في بدايتها كانت تتطلب جهداً كبيراً من مطوري التطبيقات إلا أن مجتمع المطورين سرعان ما قدم الدعم عبر توفير مكتبات جاهزة للاستخدام لهذا الغرض. لسوء الحظ توافر هذا الدعم ليس كافياً لتأمين الشبكة حيث يبقى معظم المطورين جاهلين بالتقنيات الأمنية التي يتوجب عليهم استخدامها، كما أن التطبيقات القديمة غالبا لم تُحدَّث لتتضمَّن الحلول الأمنيّة الجديدة لعدَّة أسباب كالصُّعوبة التّقنية للتعديل أو عدم اهتمام المالكين بهذا الموضوع.

كنتيجة لبقاء هذه التطبيقات ضعيفة بالرغم من توافر إجراءات المواجهة، بدأ التوجه نحو تطوير أدوات أمنية مستقلة من جهة الزبون تقدم للمستخدم الحماية التي يرغب بها. أحد الأمثلة الشائعة هي إضافة المستعرض التي تسمى [26]**NoScript**، والتي أساساً تقوم بمنع تنفيذ الجافا سكريبت. الأدوات الأمنيَّة المستقلَّة تعمل من دون أي دعم من التطبيق الذي تقوم بحمايته ويجب أن تكون قادرة على حماية جميع التطبيقات التي يزورها المستخدم. من أجل تحقيق هذه التوافقية مع جميع التطبيقات فإن هذه الأدوات توسع حدود ما هو مسموح لتحافظ على التوازن بين قابلية استخدام التطبيق وأمنه. هذا يشكل تحدّياً حقيقيّاً لأنه ليس من الممكن دائماً اتّخاذ القرار الأمنيّ الصّحيح من دون معلومات إضافيّة من المخدم.

مع ازدياد أهميّة منصّة عمل الزبون، ظهرت العديد من التقنيات الجديدة من جهة الزبون لتطبيق سياسات أمنية مقدمة من قبل التطبيق على المخدم والذي يقوم بتقديم معلومات عن السلوك المتوقع للتطبيق والمصادر المطلوبة والرموز المهمّة. كمثال أوّل على السياسات الأمنية التي يقودها التّطبيق هما علامتي **HttpOnly** و**Secure** المستخدمتان لملفات تعريف الارتباط، واللتان تحددان للمستعرض ما هي ملفات تعريف الارتباط التي يمكن الوصول إليها عبر الجافا سكريبت والتي يمكن إرسالها عبر قناة اتصال غير مشفرة. وكمثال آخر أكثر تعقيدا السياسة الأمنيّة للمحتوى **Content Security Policy CSP** والغرض من هذه السياسة هو محاربة هجمات الحقن.

الانتقاد الدائم للسياسات المقادة من قبل المخدم والمطبقة من قبل الزبون كخاصية **sandbox** والـ**CSP** هو أنها تواجه جزءاً من الهجمات وليس جميعها ولا توفر حلاّ محكماً لهجوم محدّد. لكن إذا أخذنا بالحسبان أن أغلب الهجمات تتولد من ثغرات من جهة المخدم كالفشل في تعقيم الدخل والخرج فإنّ هذا التقييم يبدو غير منصف. الغرض من هذه الإجراءات هو إعطاء المطوّر الواعي أمنيّاً أداةً إضافيّة لحصار المهاجمين الذين تمكنوا من تجاوز أحد الدفاعات من جهة المخدم، وكذلك لإضافة طبقة حماية للتطبيقات القديمة التي يصعب تعديلها وإضافة الحماية لها من جهة المخدم.

الفصل الثالث

الهجمات على الشبكة **Network Attacks**

3 الهجمات على الشبكة Network Attacks

قد تكون الهجمات على مستوى الشبكة بعيدة عن المستخدم، ولكن هذا لا يقلل من قوتها. إن تنفيذ هجوم الشبكة بنجاح لا يسمح للمهاجم بالتنصت على حركة مرور الشبكة أو التلاعب بها فحسب، ولكن أيضاً يسمح باستخدام هذه القدرات كنقطة انطلاق نحو العديد من الهجمات الأخرى، كما سنرى لاحقاً.

المحتوى في هذا الفصل موجه نحو التقنيات في طبقة التطبيق وأثر البروتوكولات المستخدمة الذي له علاقة بالتقنيات المستخدمة مثل خاصية **Secure** للكوكيز المرسلة عبر اتصال مؤمن ببروتوكول **TLS**. أمن البروتوكول يعتبر خارج نطاق هذه الرسالة، مثلاً هجمات فك التشفير على بروتوكول **TLS**. كذلك تقنيات الأمن والبنية التحتية من جهة المخدم كذلك تعتبر خارج نطاق البحث إلا إذا كان لها علاقة بالإجراءات من جهة الزبون. وكذلك البنية التحتية للشبكة.

سنغطي ثلاثة أنواع مختلفة من هجمات الشبكة. أولاً، نناقش هجمات التنصت، حيث يستمع المهاجم إلى حركة مرور الشبكة المرسلة عبر السلك أو عبر الهواء. بعد ذلك، نركز على هجمات **Man-in-the-Middle (MitM)**، حيث يعترض المهاجم حركة المرور ويتلاعب بها. أخيرًا، نركز على الهجمات التي تستهدف بروتوكول النقل **Hypertext Transfer Protocol Secure (HTTPS)**.

3.1 هجمات التنصت Eavesdropping Attacks

يتنصت المهاجم على الحركة في الشبكة مما يجعله قادراً على الوصول إلى معلومات حساسة مثل معلومات بطاقة الاعتماد وأسماء المستخدمين وكلمات السر ومحتويات الرسائل الالكترونية، ويمكنه أيضاً الوصول إلى معلومات هامة مثل معرّف الجلسة وملفات تعريف الارتباط والتي تمكن المهاجم من تصعيد الهجوم عبر خطف الجلسة **session hijacking**.

3.1.1 طرق المواجهة:

يسهم استخدام بروتوكولات النقل الآمنة، مثل الوصول المحمي بتقنية حماية الشبكات اللاسلكية**(WPA)**[27] وبروتوكول[28]**(EAP)** بشكل فعال في التخفيف من هجمات التنصت المحلي ولكنه لا يحمي حركة المرور

من التنصت ما بعد الشبكة المحلية. الطريقة الفعالة لتأمين البيانات على كامل خط النقل هي باستخدام بروتوكول **HTTP** فوق **TLS** [29] . حيث يهدف استخدام **TLS** إلى توفير السرية وسلامة البيانات والمصادقة. خاصية السرية تزيل بفعالية فائدة البيانات المنقولة بالنسبة لمهاجم التنصت، حيث ستكون البيانات مشفرة بالكامل. كما أن بروتوكول **TLS** يتم بشكل مستمر مراجعته وتقييمه أمنياً من خلال دراسات بحثية متعددة تقوم حتى بدراسة خوارزميات التشفير المستخدمة ضمنه. مثلاً تم اكتشاف هجمات تقوم باستخراج ملفات تعريف الارتباط من مجرى بيانات مشفر[30]. وتم اكتشاف نقاط ضعف في خوارزمية التشفير **RC4** [31] مما يعني أنها أصبحت تعتبر غير آمنة. كنتيجة لهذه الدراسات ومثيلاتها تقوم مجموعة عمل الـ**TLS** ضمن**IETF** بدراسة تركيبات تشفير **cipher suites** جديدة[32] .

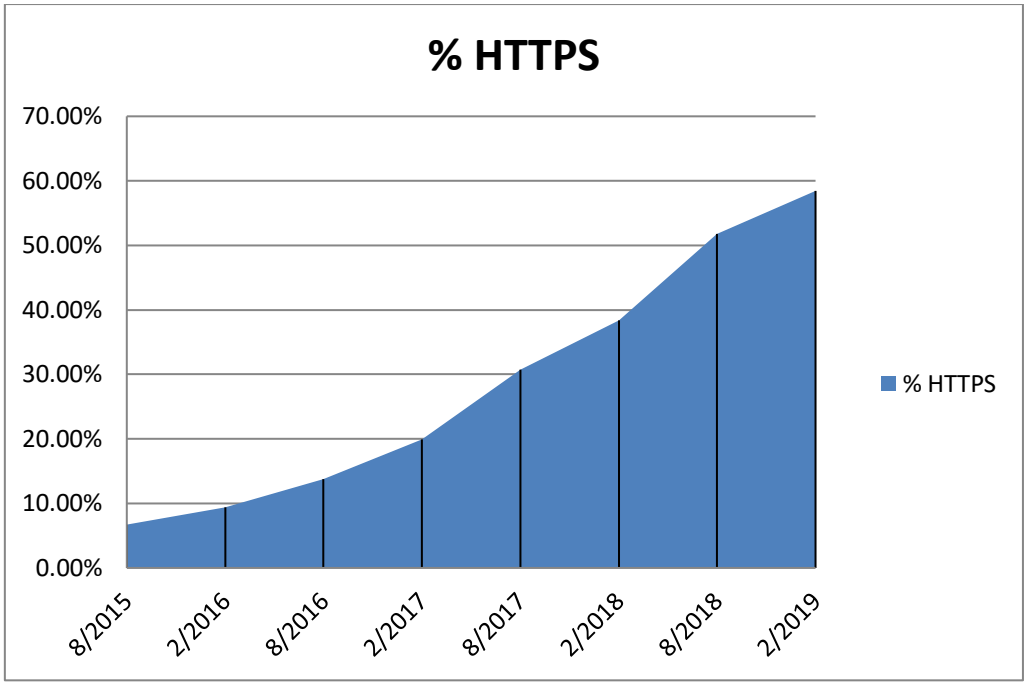
بالإضافة للشيفرات الجديدة فإن طريقة تطبيق الـ**TLS** هو عامل مهم يجب أخذه بالحسبان. الإصدارات القديمة من **TLS** تبقى قيد الاستخدام لفترة ملحوظة حتى بعد ظهور إصدارات جديدة ذات ميزات أمنية أفضل. على سبيل المثال فإنه عند حدوث الهجمات التي ذكرناها سابقاً، كانت هذه الثغرات قد تمت معالجتها باستخدام شيفرات جديدة معرفة ضمن الإصدار **TLS 1.2**.

3.1.2 الواقع الحالي:

وفقاً لإحصائيات تمت في شباط من عام 2019 فإن 58% من المواقع الـ10 مليون الأكثر شهرة وفقاً لـ **ALEXA** تستخدم **TLS** مع شهادة رقمية مصدرة من قبل سلطة معروفة [33].

Google قامت بإطلاق مبادرة "**https** في كل مكان" وكجزء من مبادرتها أصبح استخدام الـ**https** أحد عوامل الترتيب في نتائج البحث [34]، وهذا أحد أهم عوامل التزايد الواضح في عدد المواقع المستخدمة لـ **https**.

الشكل الآتي يوضح هذا التزايد:



شكل 3-1 مخطط يوضح التزايد في نسبة المواقع المستخدمة لـhttps

3.1.3 أفضل الممارسات:

استخدام **TLS** لكامل التطبيق. وينصح باتباع النصائح الموجودة في مستند الـ **BCP** الذي تقدمه **IETF** [35] للحصول على الإعداد الأفضل لـ**TLS** . كما ينصح بفحص إعداداتها باستخدام أدوات تقدم هذه الخدمة مثل: **Qualys SSL Labs**[36].

3.2 هجوم الرجل في المنتصف MitM

في هذا الهجوم يتموضع المهاجم داخل الشبكة بين الضحية والتطبيق المستهدف. هذا الموقع يجعله قادر على الوصول إلى كل البيانات المنتقلة بين الضحية والتطبيق والأهم أنه يسمح له بتعديل هذ البيانات. أي أن المهاجم قادر على التحكم بأفعال الزبون بشكل كامل ما قد يؤدي إلى آثار كارثية.

هجوم الرجل في المنتصف أخطر من هجوم التنصت وبحاجة لقدرات أكبر مع ذلك هو الأكثر انتشاراً. قد لا يكون منتشرا بشكل كبير على شكل استهداف شخصي من قبل أفراد، لكنه شائع على نطاقات واسعة مثل الرقابة المفروضة من قبل الدول.

هدف هذا الهجوم هو امتلاك القدرة على الوصول إلى بيانات الضحية المنتقلة عبر الشبكة وتعديلها. هذا يجعل المهاجم قادراً على تعديل عمليات شرعية أو تنفيذ عمليات جديدة باسم المستخدم وكذلك تعديل الملفات المرسلة من وإلى الضحية. الاتصالات المحمية بـ**TLS** مصممة لتفادي هذه الهجمات لكن بعض العيوب في النظام الداعم لها قد يسمح بتنفيذ بعض الهجمات الدقيقة بكل الأحوال. هذه العيوب قد تكون الثقة بطرف غير موثوق أو ترك عبء اتخاذ القرار للمستخدم.

تنفيذ هجوم الرجل في المنتصف في شبكة ما يمكن أن يتحقق على مستويات وقد لا يكون من المهم الاطلاع على النواحي التقنية لطرق تنفيذ الهجوم لكن لا يمكن التغاضي عن الأثر الخطر له، فما إن يتموضع المهاجم في منتصف الشبكة يصبح التلاعب بالبيانات أمراً سهلاً ومباشراً، أحد الأمثلة الشائعة حالياً هو استخدام المحتوى المختلط **mixed content** حيث يقوم المهاجم بالتلاعب بالمحتوى المنقول عبر **http** والمضمن ضمن صفحة محمية بـ **https** مما يؤدي لتهديد سياق التنفيذ للصفحة كاملةً.

مع أن **TLS** صمم ليمنع هذه الهجمات إلا أنها تبقى قائمة لعدة أسباب:

في عام **2009** تحدث **Moxie Marlinspike** [37] عن أن المستخدمين الذين يقومون بزيارة تطبيق محمي غالباً لن يقوموا بكتابة جزء الـ "**https://**" من العنوان يدوياً، هذا يعني أن الطلب الأولي سيكون عبر **http**. بعدها عادة يقوم التطبيق بإعادة توجيه المستخدم إلى العنوان المحمي الصحيح مما يسبب الانتقال من **http** إلى **https**. هذا الانتقال يمكن أن يتم استغلاله من قبل المهاجم مما يسبب خفض مستوى الاتصال من **https** إلى **http** وهذا يدعى بهجوم تعرية **SSL Stripping**[38].

ثانياً، إن خاصية التصديق في **TLS** مبنية على استخدام زوج مفاتيح خاص وعام، حيث يكون المفتاح العام مصدقاً من قبل سلطة إصدار الشهادات **Certificate Authority(CA)** والتي هي جزء من البنية التحتية للمفتاح العام **Public Key Infrastructure (PKI)**. لسوء الحظ فإن نموذج الثقة هذا يتألف من اتحاد جميع المفاتيح العامة لجميع الـ**CA** المسجلة ضمن المستعرض. أي أن أيّة **CA** يمكن أن يصدر شهادة لأيّة موقع بغض النظر عن توافر التقنية اللازمة لتقييد الثقة بـ**CA** محددة [39].

جميع المستعرضات تضم قوائم بآلاف الـ**CAs** الموثوقة [40] ما يجعل أيّة موقع عرضة للهجوم باستخدام شهادة موقعة لكن مزورة. ولجعل الأمور أكثر تعقيداً فإن من عادة المستعرضات عدم التحقق بشكل مستمر من حالة الشهادات وما إذا كانت ملغاة **revoked** باستخدام **Online Certificate Status Protocol (OCSP)** [41]، ويستعيضون عن ذلك بتضمين قائمة ثابتة من الشهادات الملغاة والتي تحدث كل فترة

زمنية محددة. وحتى لو قام المستعرض بالتحقق من الحالة باستخدام **OCSP** يتم التغاضي في حال عدم الرد مما يجعل المهاجم قادراً على تجاوز هذا التحقق.

ثالثاً، عند الكشف عن شهادة غير صحيحة من قبل المستعرض فإن عبء اتخاذ القرار الأمني يترك للمستخدم. بغض النظر عما إذا كانت الشهادة غير صحيحة لأنها انتهت المدة الزمنية المخصصة لها أو لأنها غير متوافقة مع الموقع المطلوب، تقوم المستعرضات بإظهار تنبيهات مخيفة تطلب فيها من المستخدم أن يقرر الوثوق بهذا الموقع أو لا. وبما أن المستخدمين معتادون على رؤية هذه التنبيهات للمواقع الشرعية والموثوقة فمن المرجح أن يقوم المستخدم بتجاهل التنبيه ما يجعله عرضة للهجوم.

رابعاً، نقطة ضعف أخرى في هذا النظام تأتي من أجهزة الـ**MitM** المستخدمة طواعيةً والتي تقوم المنظمات والشركات بوضعها بهدف فلترة حركة البيانات على الشبكة. أسباب استخدام هذه التجهيزات تتراوح من تأمين الحماية مثلاً باستخدام حائط ناري للتطبيق **WAF**، إلى منع الموظفين من النفاذ إلى مواقع تعتبر غير مناسبة مثل مواقع التواصل الاجتماعي. المشكلة هي أنه حتى تقوم هذه التجهيزات بعملها يجب إما أن تقوم بتخزين شهادتها على جهاز المستخدم أو أن تحصل على شهادة لكل موقع محمي على الشبكة. الخيار الأول صعب الإعداد وقابل للتطبيق فقط إذا كنت تستطيع الوصول إلى كل الأجهزة والثاني يبدو مستحيلاً. لسوء الحظ النظام لا يمنع التعاون بين الـ**CAs** وموزعي أجهزة الـ**MitM** ما يؤذي الثقة الموضوعة بالنظام.

أخيرا، الثقة الموضوعة في الـ**CAs** من السهل إساءة استخدامها عند تعرضها للخرق. مثلاً إن خرق **DigiNotar** [42] أدى إلى إصدار شهادات مزورة مما سمح بهجمات على الاتصالات المحمية لـ **Yahoo** و**Mozilla** و **WordPress**ومشروع **Tor**. الدور الموثوق لسلطات منح الشهادات يمكن أن يستغل أيضاً بالإكراه من قبل الحكومات لإصدار شهادات مزورة. هذه الاستراتيجية يعتقد انها متبعة في البلدان غير الديمقراطية لكن تسريبات "سنودن" أظهرت أنها متبعة من قبل المخابرات السرية عبر العالم[1] .

3.2.1 طرق المواجهة:

لزمن طويل الطريقة الأساس لمعالجة هجوم تعرية الـ**SSL** كان ملقى على المستخدم الذي يجب أن يلاحظ وجود إشارة القفل والذي يؤشر إلى اتصال محمي. إحدى الحلول التقنية قدمت عبر إضافة للمستعرض تدعى **HTTPS Everywhere** [43] حيث تقوم بإجبار المستعرض على استخدام **https** للمواقع التي تدعمه مما يمنع هجوم تعرية **SSL**. المقترح البحثي [44]**HProxy** يطور تاريخ الاستعراض للمستعرض ليقوم

بتخزين ملف أمني لكل موقع يقوم بزيارته ومقارنة كل زيارة مستقبلية مع هذا الملف. أخيراً، فإن المقترح البحثي **ForceHTTPS** [45] أدى إلى ظهور **HTTP Strict Transport Security (HSTS)** [46] والتي تسمح للمخدم أن يطالب المستعرضات التي تدعم هذه التقنية بأن تتصل معه باستخدام **https** حصراً. يمكن للمخدم تفعيل الـ **HSTS** ببساطة عبر تضمين الترويسة **Strict-Transport-Security** في الردود مصرحاً بطول حياة حماية الـ**HSTS**. الفجوة الوحيدة هي في أول اتصال مع الموقع حيث لا يعرف المستعرض إن كان المخدم يطبق هذه السياسة أم لا. تم معالجة هذه المشكلة من قبل المستعرضات الحديثة بتضمين قائمة مسبقة للمواقع التي فعلت الـ**HSTS** [47] ويمكن للتطبيق طلب ضمه لهذه القائمة ببساطة عبر تضمين مؤشر بسيط في ترويسة تفعيل الـ**HSTS** ولكن يجب القيام بذلك بحذر.

```
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
```

شكل 3-2

*مثال لترويسة تفعيل **HSTS** مع تفعيل خيار **preload** للتضمين المسبق بقائمة المواقع*

إحدى طرق مواجهة هجوم **MitM** على اتصالات الـ**TLS** يركز على تحديد ما إذا كانت الشهادة المقدمة موثوقة أم لا، **Certificate Transparency (CT)** [48] تهدف إلى الاحتفاظ بسجل علني للشهادات المصدرة بحيث يتمكن أصحاب المواقع من التحقق مما إذا كان هنالك شهادات مزورة باسمهم وتمكّن المستعرض من التحقق من الشهادة أيضاً.

طريقة أخرى تقوم على كشف الفروقات بين الشهادة المقدمة حالياً والشهادات السابقة تقنية تدعى نشر المفتاح العام **certificate pinning** أو **HTTP Public Key Pinning (HPKP)** [49]. **HPKP** هي ميزة تطلب من المستعرض أن يربط مفتاح تشفير عام محدد مع مخدم ويب محدد لتقليل مخاطر هجمات **MITM** بشهادات مزورة.

HPKP هي تقنية **Trust on First Use (TOFU)**. في المرة الأولى التي يقوم التطبيق بإخبار المستعرض عبر ترويسة **HTTP** خاصة بالمفاتيح العامة التي تنتمي إليه، يخزن المستعرض هذه المعلومات لفترة زمنية محددة. عندما يزور المستعرض التطبيق مرة أخرى، فإنه يتوقع أن تحتوي شهادة واحدة على الأقل في سلسلة الشهادات على مفتاح عمومي له بصمة معرفة سابقاً عبر **HPKP**. إذا قام المخدم بتسليم مفتاح عام غير معروف، يجب على المستعرض تقديم تحذير للمستخدم.

هناك عدة تحذيرات من الاستخدام الخاطئ لهذه التقنية، لأنه قد يسبب حرمان المستخدمين من استخدام التطبيق لفترة طويلة. فمثلاً إذا تعرضت الشهادة لأي مشكلة مثل السرقة أو الاسترداد فإن المستخدمين لن يكونوا قادرين على الدخول للتطبيق قبل انتهاء مدة الـ**max-age** المحددة ضمن الـ**HPKP** . لذلك ينصح بإدراج شهادة احتياطية ضمن الـ **HPKP** غير موضوعة بالخدمة لاستخدامها في مثل هذه الحالات.

الشكل الآتي يوضح مثال لترويسة **HPKP**:

```
Public-Key-Pins:
  pin-sha256="cUPcTAZWKaASuYWhhneDttWpY3oBAkE3h2+soZS7sWs=";
  pin-sha256="M8HztCzM3elUxkcjR2S5P4hhyBNf6lHkmjAHKhpGPWE=";
  max-age=5184000; includeSubDomains;
```

*شكل 3-3 مثال لترويسة تفعيل **PKP***

3.2.2 الواقع الحالي:

رأينا سابقاً التزايد الكبير لنسب استخدام **HTTPS**. ووفقاً لنفس المصدر هنالك تزايد كبير لاستخدام ترويسة **HSTS** حيث يبين المسح أن ما يقارب 14% من المواقع المليون الأكثر شهرة تستخدم هذه الترويسة في شباط عام 2019 بينما كانت 12.75 % في آب عام 2018 أي أن هنالك تزايداً ملحوظاً في استخدام هذه الترويسة.

كشفت إحدى الدراسات [50] أن الشهادات المزورة منتشرة جداً. من بين 3،447،719 اتصال **TLS** في العالم، استخدم ما لا يقل عن 6845 (0.2٪) شهادة مزورة. يعزو الباحثون هذه الشهادات المزورة إلى البرامج الدعائية والبرامج الضارة وأدوات الأمان مثل برامج مكافحة الفيروسات وعناصر تحكم الوالدين وجدران الحماية.

3.2.3 أفضل الممارسات:

يصعب تقديم أفضل الممارسات لتجنب هجمات **MitM** على الاتصالات المؤمنة بواسطة **TLS**، لأن منع مثل هذه الهجمات هو هدف **TLS** تمامًا. يعد مستند الـ **BCP** الذي تقدمه **IETF** [35] مرجعاً جيداً للتطبيق الآمن لـ **TLS**، والذي يناقش أفضل الممارسات الحالية، مثل اختيار مفتاح قوي بما فيه الكفاية، وتشفير موقعك بالكامل، وتمكين السرية الأمامية (**forward secrecy**)، وما إلى ذلك. بالإضافة إلى ذلك، تمكين **HSTS**. في غضون ذلك، لا يزال الخبراء يناقشون المسار المستقبلي لطرق التطبيق الأكثر أمانًا، حيث من المحتمل أن يكون اختيار الـ**CA** الذي يدعم شفافية الشهادات **CT** وتمكين نشر المفتاح العام **PKP**، من الممارسات الجيدة. لحسن الحظ، يبدو أن تحسين تجربة **TLS** من جهة المستخدم هو جهد مستمر لمطوري المستعرضات.

3.3 الهجمات على مستوى بروتوكول الـhttps

بمجرد بدء الاتصال الآمن بدون وجود **MitM**، يجب أن نكون قادرين على اعتبار المحتوى المرسل آمناً. لكن للأسف بعض الهجمات المتطورة على بروتوكولات **HTTPS** و**TLS** قادرة على استخراج البيانات من اتصال آمن، أو إضافة بيانات له. لحسن الحظ، تتم معالجة معظم هذه الهجمات في إصدارات محسنة من بروتوكول **TLS**.

أمثلة عملية عن هذه الثغرات أو الهجمات تتضمن: **Lucky-13** [30] و[51]**BEAST** و **CRIME** [52]و**BREACH** [53] والهجمات التي تستهدف الـ**RC4**[31] .

لا توجد مناقشه مفصّله لتقنيات المعالجة، حيث لا توجد تقنيّات مخصصة للتخفيف من هذه الهجمات عدا عن مواكبة أحدث إصدار للبروتوكول.

كلما تم اكتشاف عيب في بروتوكول أو أداة ما، فإنه غالباً ما يكون قد عولج في الإصدار الأحدث، أو سيتم الإعلان عن حل على الفور تقريباً. فمثلاً ثغرة النزيف القلبي (**Heartbleed** [54]) كانت قد عولجت وتم نشر ملف التصحيح لها عندما تم الإعلان عن وجودها.

في سعيها الدائم لجعل الشبكة أكثر أماناً قامت **Google** بالإعلان [55]عن خططها لوقف الدعم عن بروتوكولات **SSL** و**TLS 1.0** و**TLS 1.1** بدءاً من الإصدار **Chrome 81**. وكخطوة تحذيرية كل

المواقع التي تستخدم إحدى هذه النسخ غير الآمنة سيظهر لمستخدميها تنبيه بعدم أمنها ابتداء من 2020\01\13 وذلك بدءاً من الإصدار **Chrome 79** [56]. وكذلك ستفعل **Apple** و**Mozilla** و**Microsoft**.

الفصل الرابع:

الهجمات على طلبات المستعرض

4 الهجمات على طلبات المستعرض

عبر مهاجمة طلبات المستعرض. المهاجم قادر على تزوير طلبات إلى التطبيق الهدف باسم المستخدم. المهاجم يخدع مستعرض المستخدم لإرسال طلب مزيف، غالباً من دون أن يلاحظ المستخدم ذلك.

المشكلة في تزوير الطلبات هو أن التطبيق الهدف غالباً لا يستطيع التفريق بين الطلبات الشرعية والمزوّرة. من المستحيل أن يقرر التطبيق ما إذا كان طلبٌ ما مزوراً أم لا دون اتخاذ إجراءات إضافية.

سنغطي طريقتين أساسيتين لتزوير الطلبات باسم المستخدم، الأولى تزوير الطلب العابر للموقع **Cross Site Request Forgery(CSRF)** حيث يخدع المهاجم مستعرض الزبون لإرسال الطلبات إلى التطبيق الهدف اوتوماتيكياً. هذا النوع من الهجمات يمكن أن يسبب العديد من المشاكل مثل تعديل معلومات الحساب الشخصي أو مثلاً القيام بمناقلات بنكية كتحويل النقود من حساب إلى آخر. الثانية هي تعديل واجهة المستخدم **UI Redressing** حيث يقوم المهاجم بخداع المستخدم للتفاعل مع صفحة بريئة في الظاهر. بينما في الحقيقة العمليات التي تتم يتم إرسالها إلى التطبيق الهدف. هذا النوع من الهجمات يمكن أن يسبب مثلاً تحديث الحالة للمستخدم على مواقع التواصل الاجتماعي من دون علمه أو مثلاً تفعيل كاميرا الويب من خلال إعدادات مشغل ملفات الفلاش.

4.1 تزوير الطلب العابر للموقع (CSRF)

يكون المهاجم قادراً على تزوير طلبات إلى التطبيق المستهدف انطلاقاً من مستعرض خاص بمستخدم شرعي. التطبيق المستهدف يقوم بمعالجة هذه الطلبات المزورة بنفس الطريقة كما لو أنها طلبات حقيقية من الضحية. في حال نجاح هذا الهجوم قد يسبب العديد من التبعات كتعديل إعدادات الحساب أو سرقة النقود من حساب بنكي.

هجوم **CSRF** ما زال شائعا وله الترتيب التاسع ضمن **CWE/SANS Top 25** للأخطاء البرمجية الأكثر خطورة للعام 2019 [24]. التطبيقات الكبيرة والصغيرة تتأثر على السواء و كمثال على انتشار هذه الثغرات هو الهجوم على الأنظمة البنكية[57] و[58]**Gmail** و[5]**eBay**. حتى ينجح هذا الهجوم يجب أن يكون المستخدم قد قام بتسجيل الدخول إلى التطبيق. من وجهة نظر التطبيق، الطلبات المزورة لها نفس بنية الطلبات الشرعية ولذلك لا يمكن تمييزها عن الطلبات الشرعية.

عملية خداع المستعرض ليقوم بإرسال هذه الطلبات هو عملية مباشرة. المستعرضات تقوم بشكل مستمر بإرسال الطلبات للعديد من المواقع غير المترابطة، على سبيل المثال عند تحميل المصادر الخارجية مثل الصور وملفات التنسيقات **stylesheets** أو مستند سيتم تحميله ضمن إطار ما، أيضاً عند إرسال بيانات استمارات الإدخال إلى عنوان ذو أصل مختلف "**Cross-origin URI**". يمكن للمهاجم بسهولة أن يضمّن كود في موقعه الخاص أو في موقع يستطيع التحكم به، يقوم هذا الكود بإرسال طلبات للتطبيق المستهدف، أو قد يقوم بحقن كود جافا سكريبت أو **html** في موقع غير مرتبط ولكنه شرعي مثلاً عبر نشر كود في قسم التعليقات لمنتدى ما. الشكل الآتي يوضح استخدام صورة مخفية لتفعيل طلب مزور:

```
1  <img width="0" height="0"
2            src="http://admin.example.com/deleteAccount.php" />
```

شكل 4-1 استخدام صورة مخفية لتفعيل طلب مزور

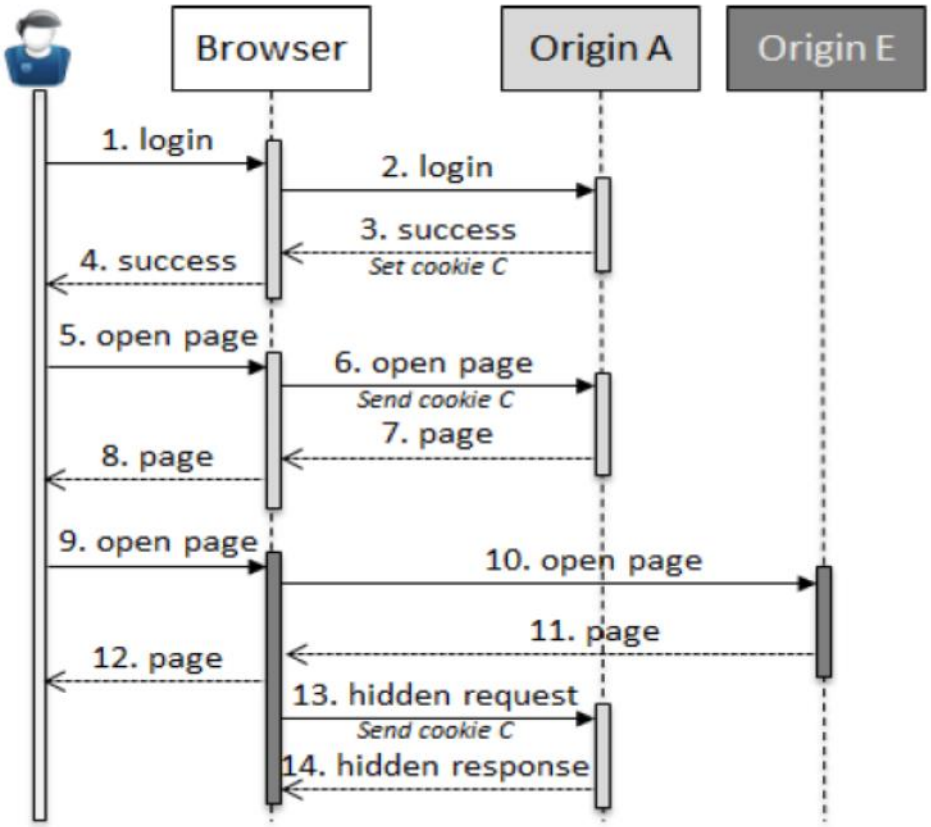
والشكل الآتي يظهر إرسال نموذج إدخال إلى عنوان ذو أصل مختلف، حيث يقوم هذا الكود بإنشاء إطار مخفي يحتوي نموذج إدخال يتم إرساله لاحقاً للتطبيق المستهدف.

```
1   document.getElementById("somediv").innerHTML += "<iframe
2             id='attackframe' style='height: 0px; width: 0px;'></iframe>";
3   var f = document.getElementById("attackframe");

4   var code =    "<form id='attackform' action='http://admin.example.com/
5             createAccount.php' method='POST'>";
6   code +=       "<input type='hidden' name='username' value='attacker'>";
7   code +=       "<input type='hidden' name='password' value='12345678'>";
8   code +=       "<input type='hidden' name='action' value='create'>";
9   code +=     "</form>";

10  f.contentDocument.body.innerHTML = code;
11  f.contentDocument.getElementById("attackform").submit();
```

شكل 4-2
إرسال نموذج إدخال إلى عنوان ذو أصل مختلف باستخدام إطار مخفي يحتوي نموذج إدخال يتم إرساله لاحقاً للتطبيق المستهدف

هجوم **CSRF** يكون ناجحاً فقط إذا تم ضمن جلسة موثقة مسبقاً بين مستعرض الضحية والتطبيق المستهدف. لسوء الحظ فإن تصميم آليات إدارة الجلسات الحالي في المستعرض يقوم بربط معلومات الجلسة مع كل طلب صادر، مما يسهّل هذه الهجمات. فالمستعرض يقوم بإرسال ملفات تعريف الارتباط المتعلقة

بتطبيق ما مع كل طلب إلى التطبيق إن كان صادراً عن التطبيق ذاته أو عن غيره. الشكل الآتي يوضح العملية:



شكل 3-4
آلية إرسال ملفات تعريف الارتباط مع الطلبات

بالإضافة لذلك العديد من التطبيقات تستخدم الجلسات طويلة المدى والتي تستمر حياتها طالما المستعرض مفتوح بغض النظر عمَا إذا كان التطبيق ما زال مفتوحا في أحد ألسنة المستعرض أم لا. فإذا كانت الجلسة موثقة قد يؤدي هذا إلى إمكانية استهداف التطبيق حتى إن كان غير مفتوح ضمن المستعرض.

هجوم تزوير تسجيل الدخول "**Login CSRF**" هو أحد أنواع هجمات تزوير الطلب العابر للموقع حيث يقوم المهاجم بتزوير طلب تسجيل دخول للضحية باستخدام حساب يختاره المهاجم، وعندما يقوم الضحية بإرسال طلب للتطبيق فإن أي بيانات مدخلة سيتم ربطها بحساب المهاجم وقد تتسرب إلى المهاجم. مثلاً محرك بحث يحتفظ بتاريخ الاستعراض للمستخدمين الموثقين (الذين قاموا بتسجيل الدخول).

بالإضافة إلى هذا الشكل التقليدي لهجوم تسجيل الدخول المزور فقد يستهدف هذا الهجوم التطبيقات التي تستخدم مزودات التوثيق الخارجية مثل **OpenID** أو **authentication Facebook**. هذه المزودات تقوم بتزويد التطبيق بآلية توثيق **assertion** تحوي المعلومات الضرورية لتأكيد عملية التوثيق وتأكيد هوية المستخدم. باستخدام هذا الهجوم يتمكن المهاجم من إرسال الـ**assertion** الخاص به للتطبيق المستهدف من داخل مستعرض الضحية وتحقيق جلسة موثقة مرتبطة ببيانات المهاجم.

المشكلة في هجمات التزوير هي أن المخدم غير قادر على تمييز ما إذا كان الطلب شرعي أم مزور. كما أن قيام المستعرضات بمعالجة الطلبات الداخلية ضمن نفس التطبيق والخارجية بشكل متشابه واعتماد التطبيقات الحالية على هذه الآلية يعيق آليات مواجهة هذا الهجوم.

4.1.1 طرق المواجهة:

خلال السنين الأولى لظهور هجوم تزوير الطلبات تم اقتراح العديد من آليات المواجهة ولكن أثبتت عدم فعاليتها:

- إحدى آليات المعالجة تفرض التحقق من ترويسة Referer [59][1] من جهة المخدم. حيث لا تقبل الطلبات التي لها أثر فعلي على التطبيق إلا إذا كانت قيمة الـ Referer تشير إلى موقع موثوق وإلا يتم رفض الطلب. التحقق من الـReferer قادر على مواجهة هجوم التزوير لكن للأسف لا يمكن الاعتماد على وجودها. فغالباً ما تكون غير موجودة لأسباب تتعلق بالخصوصية حيث تشير هذه الترويسة إلى المصدر الذي أصدر الطلب وعنوانه. كما أن المستعرضات لا تقوم بتضمينها عندما يقوم مصدر ما تم تحميله على https بطلب مصدر على http. هنالك العديد من إعدادات المستعرضات وبرامج الخصوصية تمكن من الحذف الاوتوماتيكي للترويسة.
- كتحسين لترويسة Referer فإن ترويسة Origin [60] تزود المخدم بمعلومات عن أصل الطلب من دون معلومات ذات خصوصية كبرى كالتي تحويها ترويسة Referer. لسوء الحظ، إن التوصيف الرسمي لها لا يجبر المستعرضات على إرسالها وإنما ينص على أنه "يمكن" استخدامها. لكن عند استخدام مشاركة المصادر ذات الأصل المختلف CORS يصبح استخدامها الزامياً.
- كبديل لذلك فالمقاربات المستندة إلى استخدام الرموز token-based [61] هي أكثر فعالية. مثلا، إرسال حقل مخفي ضمن نموذج الإدخال. عندما يقوم المستعرض بإرسال طلب حساس، يتم تضمين الرمز بشكل اوتوماتيكي ويتم التحقق منه عند المخدم.

[1] لا يوجد خطأ املائي هنا. وإنما هذا الخطأ يعود لمستند التعريف الأصلي لهذه الترويسة وقد تم المحافظة عليه بهذا الشكل إلى يومنا هذا.

الأهم لضمان فعالية هذه الطرق هو إبقاء الرمز بعيداً عن متناول المهاجم ويتحقق ذلك بتضمين الرمز في الصفحة حيث يكون محمياً بسياسة الأصل الواحد SOP مما يمنع سرقتها من قبل سياق تنفيذ يتحكم به المهاجم ضمن نفس المستعرض. سرية الرمز لها أهمية كبرى، فيجب أن يكون قيمة يصعب التنبؤ بها ويجب أن تكون مميزة لكل مستخدم. هذا يوجب أن يتم تخزينها ضمن غرض الجلسة المخزن عند المخدم مما قد يؤثر على أدائه.

- أسفرت المزيد من الأبحاث حول الأساليب المستندة إلى استخدام الرموز، والتي غالباً ما تتعارض مع تطبيقات الويب 2.0، عن العديد من التحسينات على الرموز التقليدية، لتمكين البرمجة المعقدة من جانب المستعرض والطلبات المشتركة بين المواقع. jCSRF [62] ، وهو حل وسيط من جانب المخدم ، يضيف بشفافية رموز الأمان إلى موارد العميل ويتحقق من صحة الطلبات الواردة. بدلاً من ذلك، تقوم تقنيات الإرسال المزدوج [63] بتضمين nonce في موقعين مختلفين، على سبيل المثال، ضمن ملف تعريف الارتباط وكحقل نموذج مخفي، مما يسمح للمخدم بمقارنة كلتا القيمتين، دون متابعة الحالة. نظرًا لعدم تمكن المهاجم من التعامل مع كلا الرمزين، فلا يمكنه تزوير طلبات صالحة.
- خاصية **SameSite** [64] لملفات تعريف الارتباط: تسمح للمخدم بالتأكيد على أن ملفات تعريف الارتباط يجب ألا يتم إرسالها ضمن طلبات عابرة للمواقع. هذا التأكيد يسمح للمستعرضات بالتخفيف من خطر تسرب المعلومات لجهة غريبة، ويوفر بعض الحماية ضد هجمات **CSRF**، تأخذ الخاصية إحدى قيمتين:

  - **Strict**: لن يتم إرسال ملف تعريف الارتباط إلا اذا كان الطلب صادراً من نفس التطبيق. للتوضيح لنفترض أن المستخدم قام بتسجيل الدخول إلى موقع **github** وأثناء تصفحه لبعض المنتديات ضغط رابط لأحد المشاريع المنشورة على هذا الموقع. في حال تحديد القيمة **strict** لن يتم إرسال ملف تعريف الارتباط إلى **github** وبالتالي لن يتمكن المستخدم من رؤية المشروع. هذه الحالة مناسبة لتطبيق البنك مثلاً ولكنها غير عملية بالنسبة لمواقع التواصل الاجتماعي.

  - **Lax**: وهي القيمة الافتراضية وتسمح بإرسال ملف تعريف الارتباط حتى لو كان الطلب من رابط خارجي.

```
Set-Cookie: sess=abc123; path=/; SameSite=lax
```

شكل 4-4

*استخدام ترويسة **SameSite***

- حلول من جهة المستخدم تقوم على كشف الطلبات التي من المحتمل أن تكون خطرة وإما تقوم بحظرها أو تجردها من المعلومات المهمة كملفات تعريف الارتباط. مثل الأدوات: RequestRodeo[65] وCsFire[66] وRequestPolicy[67] وDeRef[68] و NoScript ABE[69]. العيب الأساسي في هذه الأدوات هو عدم التوافقية مع جميع المواقع ما يؤدي إلى ظهور تنبيهات غير ضرورية أو غير صحيحة وهذا يؤثر على قابلية الاستخدام.

4.1.2 الواقع الحالي:

تركز الممارسات الحالية للتخفيف من هجمات **CSRF** على المقاربات المستندة إلى استخدام الرموز، إما مصممة خصيصًا للتطبيق أو موجودة كجزء من إطار عمل ويب، مثل **Ruby on Rails** و**CodeIgniter** والعديد من البرامج الأخرى. كذلك، توفر المكتبات أو واجهات برمجة التطبيقات من جانب المخدم حماية **CSRF** أيضاً، مثل واجهة برمجة تطبيقات **OWASP ESAPI** و**CSRFGuard**. يمكن للمواقع التي يتم بناؤها باستخدام نظام إدارة المحتوى (**CMS**) الاستفادة من دعم **CSRF** المدمج أيضاً. على سبيل المثال، توفر **Drupal** و**Django** و**WordPress** حماية **CSRF** المستندة إلى الرمز المميز.

أما من ناحية استخدام خاصية **SameSite** وفقاً لورقة العمل الصادرة عن **IETF** [70] فإن غوغل أعلنت أنها اعتباراً من الإصدار **Chrome80** جميع ملفات تعريف الارتباط التي لم تذكر لها خاصية **SameSite** ستعامل وكأن لها القيمة الافتراضية **lax**، أما ملفات تعريف الارتباط التي تنص صراحة على القيمة **none** فلن يتم إرسالها ما لم تكن خاصية **Secure** مفعلة أيضا.

4.1.3 أفضل الممارسات:

استخدام خاصية **SameSite** بالإضافة للمقاربات المعتمدة على استخدام الـ**tokens** مع الانتباه لحماية هذه الرموز. وبالنسبة للتطبيقات القديمة فإن الحلول الجاهزة من جهة المخدم مثل **CSRF Guard** يمكن استخدامها.

كخط حماية ثاني ينصح بطلب التأكيد من المستخدم عند القيام بأي طلب حساس قد يؤدي لتغير حالة المستخدم، مثلاً عندما يحاول المستخدم تغيير كلمة السر يجب أن نطلب منه إعادة تسجيل الدخول بكلمة السر القديمة أولاً.

4.2 تعديل واجهة المستخدم UI redressing

هجوم تعديل واجهة المستخدم، والمعروف أيضاً باسم **Click Jacking** أو **Tap Jacking** يعيد تصميم الواجهة المرئية "**redecorates**" للتطبيق المستهدف، مما يربك المستخدم الذي يتفاعل مع التطبيق. على سبيل المثال، يوضح الشكل الآتي هجوم خطف النقرات **clickjacking** باستخدام طبقة شفافة فوق الواجهة الحقيقية:

شكل 4-5

*هجوم خطف النقرات **clickjacking** باستخدام طبقة شفافة فوق الواجهة الحقيقية*

يمكن استخدام هجمات تعديل واجهة المستخدم لتحفيز أي تفاعل من جانب المستخدم داخل التطبيق المستهدف، مثل النقر فوق زر، أو سحب العناصر وإسقاطها، إلخ...

يستخدم هجوم معالجة واجهة المستخدم العديد من الميزات البريئة، مع دمجها لخداع المستخدم للنقر فوق عنصر حساس. لا يمكن أن تكون هجمات تصحيح واجهة المستخدم مزعجة فحسب، بل يمكن أن تكون ضارة أيضاً. من الأمثلة عليها، **Tweetbombs** [71]، التي تنشر تحديثات حالة **Twitter** على حساب

الضحية ، و **LikeJacking** ، والتي تؤدي إلى الإعجابات غير المقصودة على صفحات **Facebook**. من الأمثلة على هذه الهجمات تلك التي تخدع المستخدم لتمكين الوصول إلى كاميرا الويب لمشغل **Flash** [72] ، والهجمات على أجهزة التوجيه اللاسلكية ، وسرقة مفاتيح **WPA** السرية [73].

4.2.1 طرق المواجهة:

- استخدام ترويسة **X-Frame-Options(XFO)** [74]: تم تقديمها في الأصل من قِبل **Microsoft** في **IE 8** وتم إضفاء طابع رسمي عليها لاحقاً، يتم استخدام هذه الترويسة ليحدد التطبيق ما إذا كان من المسموح لتطبيقات أخرى وضعه ضمن **<frame>** أو **<iframe>** أو **<embed>** أو **<object>**.

```
• X-Frame-Options: sameorigin
• X-Frame-Options: deny
```

شكل 4-6
*القيم الممكنة لترويسة **X-Frame-Options***

- استخدام خاصية **frame-ancestors** التي تقدمها **CSP**: تم تصميم **CSP** في البداية للحماية من هجمات الـ**XSS** وهجمات الحقن الأخرى. لكنها أيضاً تقدم خاصية **frame-ancestors** لتحديد المصادر التي يسمح لها بتضمين الصفحة ضمن **<frame>** أو **<iframe>** أو **<embed>** أو **<object>**.

```
• Content-Security-Policy: frame-ancestors 'none';
• Content-Security-Policy: frame-ancestors 'self' https://www.example.org;
```

شكل 4-7
*أمثلة على استخدام خاصية **frame-ancestors***

- حلول من جهة المستعرض:
  - **InContext** [72]: تتضمن عدة إجراءات لضمان أن النقرة غير مزورة. مثلاً عبر مقارنة لقطات للشاشة عند النقر، أو تسليط الضوء على منطقة المؤشر لمنع الهجمات التي تعتمد على وجود مؤشر مزيف.

- **NoScript** [26]: فيه مكون يقوم بمقارنة لقطات للشاشة للمنطقة التي تم النقر فيها وهذا كان أساساً للتقنية السابقة **InContext** وعند كشف أي اختلاف يتم تحذير المستخدم ويطلب منه تأكيد الحدث قبل إرسال الطلب.

4.2.2 الواقع الحالي:

بطبيعتها جميع تطبيقات الويب عرضة لهجمات النقر، لكن الهجوم لا يحظى إلا باهتمام ضئيل مقارنة بالهجمات ذات الخطورة العالية. وقع مستخدمو تطبيقات الويب الرئيسية المعروفة، مثل **Twitter** و**Facebook** وما إلى ذلك، ضحية هذا النوع من الهجمات. تنشر العديد من تطبيقات الويب شكلاً من أشكال التعليمات البرمجية للهروب من التأطير والتي قد يتمكن المهاجم من تفاديها. بالإضافة إلى ذلك، يكون للعديد من التطبيقات واجهة أمامية مختلفة للمتصفحات العادية عن متصفحات الجوّال، وغالبًا ما تطبق تقنيات الهروب من التأطير على نسختها العادية فقط. حاليًا، استخدام ترويسة **X-Frame-Options(XFO)** في تزايد مستمر[33] حيث وصل عدد المواقع التي تستخدمه في شباط 2019 إلى 152.231 موقعا من المواقع المليون الأكثر استخداماً.

4.2.3 أفضل الممارسات:

من الناحية المثالية، يجب أن تستخدم التطبيقات كلاً من الكود البرمجي الذي يقيد الإطارات **frame busting code** حيثما أمكن ذلك (المثال في الشكل الآتي) بالإضافة لتقنيات تقييد الإطارات التي تحدثنا عنها، إما مع ترويسة **XFO** أو من خلال خاصية **frame-ancestors**. من المؤكد أن التفاعل الإضافي مع المستخدم، مثل مربع حوار تأكيد واضح، سيزيد صعوبة هذه الهجمات ولكن لن يكون دائمًا كافياً للقضاء عليها[72].

```
if( self == top){
    document.documentElement.style.display = 'block' ;
} else {
    top.location = self.location ; }
```

شكل 4-8

*مثال لكود برمجي يقيد الإطارات **frame busting code***

الفصل الخامس

الهجمات على الجلسة

5 الهجمات على الجلسة

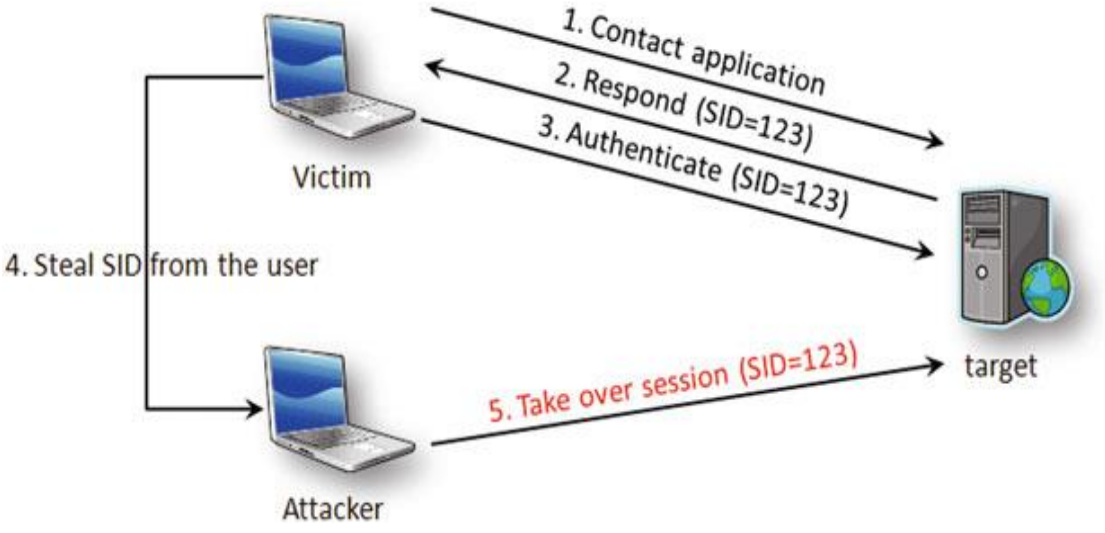
الهجمات السابقة على الطلبات الصادرة من المستخدم تمكّن المهاجم من إرسال طلبات من داخل المستعرض.

الهجمات على الجلسة عامةً لها أثر أكبر حيث تعطي المهاجم سيطرة كاملة على الجلسة. وتحتل المرتبة الثانية للمخاطر الأمنية الأكثر أهمية بحسب قائمة منظمة **OWASP** لعام 2017 [23]. بالسّيطرة على جلسة موثّقة يحصل المهاجم على نفس سماحيات المستخدم الضحيّة. بعض الهجمات تسمح للمهاجم بالتحكّم بالجلسة في مستعرض الضحيّة، هجمات أخرى تسمح للمهاجم بسرقة بيانات الترخيص الخاصّة بالضحيّة.

سبب نجاح هذه الهجمات هو استخدام أنظمة توثيق ضعيفة وعدم تقديم الحماية الكافية للجلسات الموثّقة.

5.1 هجوم اختطاف الجلسة Session Hijacking

يهدف إلى نقل الجلسة الموثّقة إلى مستعرض آخر أو إلى جهاز آخر، مما يسمح للمهاجم بإكمال العمل ضمن جلسة الضحية.



شكل 5-1
آلية تنفيذ هجوم اختطاف الجلسة

يعتمد هذا الهجوم على قدرة المهاجم على الحصول على معرف الجلسة وهذا ممكن بالاعتماد على عدة هجمات مثل الـ**XSS** أو النفاذ إلى مخزن ملفات تعريف الارتباط ضمن المستعرض مثلاً عبر إضافة مستعرض خبيثة **malicious browser extension** أو عبر التنصت على الشبكة أو مثلاً عبر توقع قيم للجلسات مثلاً إذا كانت القيم تسلسلية أو عبر هجوم **brute-force**. بالمختصر هذا الهجوم خطر جدا وممكن لأنه من السهل الحصول على معرف الجلسة ونقله.

5.1.1 طرق المواجهة:

- إحدى الطرق التقليدية هو ربط الجلسة بعنوان IP وهي طريقة فعالة إلا إذا كان عنوان الـIP العام مشترك بين عدة أجهزة أو عندما يتغير هذا العنوان خلال الجلسة وهاتان الحالتان متواجدتان بكثرة في البنية الحالية للشبكة
- حالياً يوجد تقنية محسنة وهي عبر استخدام بصمة المستعرض browser fingerprinting [75] [76] حيث تجمع عدة خصائص للمستعرض لتشكيل بصمة
- مقاربة أخرى تعتمد على منع سرقة المعرف والذي غالباً ما يكون مخزناً ضمن ملف تعريف الارتباط وذلك باستخدام خصائص جديدة لملف تعريف الارتباط:
  - HttpOnly: تمنع النفاذ لملف تعريف الارتباط من خلال الجافا سكريبت
  - Secure: تمنع نقل ملف تعريف الارتباط عبر قناة غير مشفرة مما يوقف هجمات التنصت.

التطبيق الصحيح لكلتا السمتين على ملفات تعريف الارتباط التي تحمل معرف جلسة يحبط بشكل فعال هجمات XSS، بالإضافة إلى هجمات اختطاف الجلسة من خلال التنصت على حركة مرور الشبكة.

هناك توجه بحثي قديم ومستمر يعتمد على تأمين الحماية ضد سرقة الجلسة من داخل التطبيق نفسه من دون الاعتماد على تقنيات مدعومة عند المستخدم. الفكرة هي في عدم اخفاء معرف الجلسة وانما ضمان أن معرفة هذا المعرف غير كافية لسرقة الجلسة.

- SessionLock [77]: يتفاوض على سر مشترك بين المستعرض والمخدم ويخزنه في سياق المستعرض. يتم استخدام السر لاختبار نزاهة الطلبات الصادرة. نظرًا لأن القيمة السرية لا يتم نقلها

مطلقًا بشكل واضح، فإنه يمنع المهاجم الذي لديه معرف جلسة مسروقة من تقديم طلبات صالحة. لسوء الحظ، نظرًا لأن السر يتم تخزينه في سياق جافا سكريبت، فلا يمكن حمايته من الهجمات القائمة على البرامج النصية.

- GlassTube [78]: يضمن النزاهة في نقل البيانات بين المستعرض والمخدم، ويمكن تطبيقه سواء داخل التطبيق أو كتعديل للبيئة من جانب المستعرض، على سبيل المثال كمكون إضافي للمستعرض

مقاربات تعتمد على تقوية ملفات تعريف الارتباط، على سبيل المثال:

- One-Time Cookies [79] تعتمد على استبدال معرفات الجلسة الساكنة بtokens استهلاكية اي tokenخاص بالطلب ويتم التخلص منه بعد كل طلب. ويتم ذلك باستخدام سر مشترك في البدء.

- Macaroons [80] وضع قيود على كيفية ومكان وزمن استخدام السلطة الضمنية التي يحملها المعرف. التقنية خلفها تعتمد على سلسلة متداخلة من الـ HMAC

مقاربات تهدف الى تقديم الحماية من جهة المستعرض من دون الاعتماد على التطبيق المستهدف:

- SessionShield [81] يواجه الهجمات عبر ضمان ان كل ملفات تعريف الارتباط المرتبطة بالجلسة موصفة كـhttpOnly قبل وصولها الى المستعرض.

- Serene [82] هو اضافة لمستعرض Firefox لجعله يدعم إدارة الجلسات المعتمدة على الوسطاء parameter-based.

5.1.2 الحالة الحالية:

العديد من المواقع ما تزال تستخدم ملفات تعريف الارتباط غير المحمية لتخزين معرفات الجلسات تاركين المستخدمين عرضة لهجمات سرقة الجلسة. ولكن من وجهة نظر تفاؤلية يوجد تزايد كبير لاستخدام خاصتي **httpOnly** و **Secure**.

5.1.3 أفضل الممارسات:

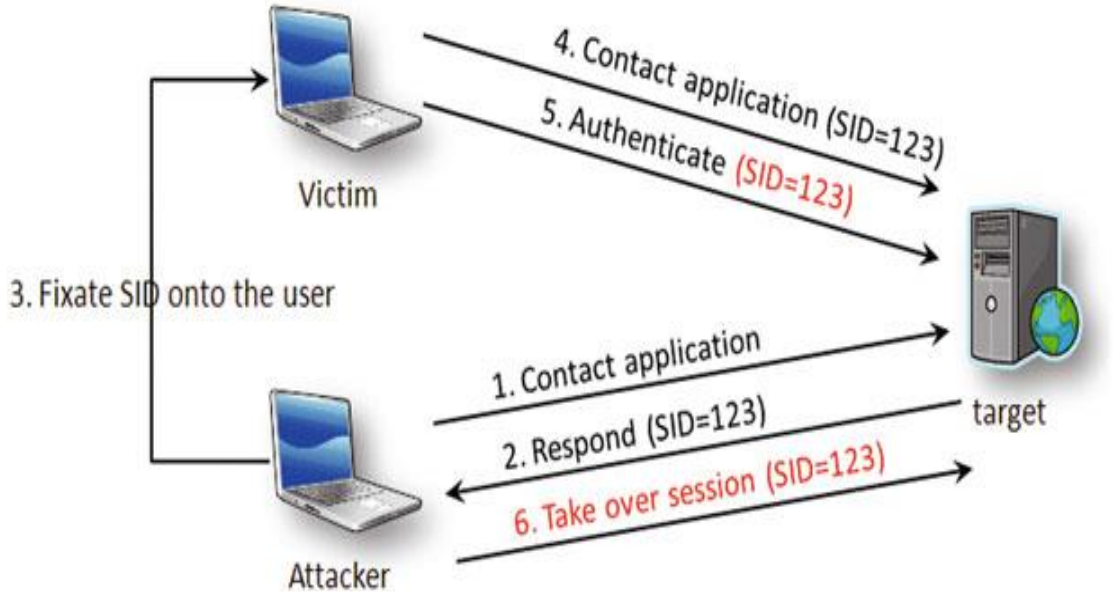
استخدام معرفات جلسة قوية وعشوائية [83] وتفعيل التطبيق على **https** واستخدام خاصتي **httpOnly** و**Secure** لجميع ملفات تعريف الارتباط التي لا نحتاج للنفاذ اليها من جافا سكريبت وخاصة التي تحتوي معرفات الجلسات.

```
Set-Cookie: __Host-sess=123; path=/; Secure; HttpOnly; SameSite
```

شكل 5-2
الإعدادات الأفضل لحماية معرفات الجلسات

5.2 هجوم تثبيت الجلسة Session Fixation

يتمكن المهاجم من إجبار المستعرض على استخدام معرف جلسة محدد معروف مسبقاً للمهاجم. هدف المهاجم هو أن ينتظر قيام المستخدم بأفعال مغيرة للحالة مثلاً القيام بتسجيل الدخول وبعدها يستولي على الجلسة. آثار تثبيت الجلسة مشابهة لسرقة الجلسة.



شكل 5-3
آلية هجوم تثبيت الجلسة ***Session Fixation***

هذا الهجوم أصعب من السابق ويتطلب القدرة على نقل معرف الجلسة إلى مستعرض الضحية. لا يوجد مؤشرات دقيقة على مدى انتشار هذا النوع من الهجمات إلا أن الكمية الكبيرة لطرق الهجوم التي تؤدي إلى هذا الهجوم هو مؤشرٌ كافٍ على انتشاره.

في نظم إدارة الجلسات المعتمدة على ملفات تعريف الارتباط، يقوم المهاجم أولاً بالحصول على معرف جلسة إما بزيارة التطبيق المستهدف بنفسه أو بتأليف معرف. في الخطوة التالية يتوجب على المهاجم تثبيت المعرف في مستعرض الضحية وهذا يعتمد على تقنية إدارة الجلسة المستخدم. ما إن يتم تثبيت الجلسة ويقوم المستخدم بزيارة التطبيق سيكون ضمن جلسة المهاجم. وهذا يعني أن الحالة المرخصة للمستخدم ستنتقل للمهاجم.

الجزء الأكثر أهمية هو تثبيت المعرف في مستعرض الضحية وذلك يحتاج لوجود ثغرة أخرى يتم استغلالها للقيام بهجوم **XSS** أو **header injection** [82]. مثلاً قد يعطي المهاجم قيمة لمعرف الجلسة باستخدام خاصية "**document.cookie**" باستخدام جافا سكريبت أو عن طريق حقن عنصر **meta** يقوم بتقليد عمليات الترويسة أو عبر التلاعب بالبيانات على الشبكة.

5.2.1 طرق المواجهة:

بسبب وجود عدة طرق للقيام بهذا الهجوم فإن الإحاطة بكل هذه الهجمات صعب. ولكن حماية ملفات تعريف الارتباط للجلسات باستخدام خاصيتي **httpOnly** و**secure** تجعل هذا الهجوم أصعب بكثير، بما أنها تمنع المهاجم من الحصول بسهولة على معرف جلسة موجود سابقاً. لكن هذه الحمايات يمكن تجاوزها، على سبيل المثال من خلال ملء مخزن ملفات تعريف الارتباط في المستعرض، مما يتسبب بقيامه بحذف الأقدم منها والسماح للمهاجم بتثبيت معرف جديد للجلسة.

إحدى التقنيات الفعالة هي إرسال معرف جديد للمستخدم عند تغيير الصلاحيات للمستخدم مثلاً عند تسجيل الدخول أو الخروج، أو النفاذ إلى جزء إداري من التطبيق. وهذه الطريقة هي مسؤولية المخدم وغالبا ما تكون مدعومة من قبل لغة البرمجة أو بيئة التطوير. لا تتطلب دعم من الزبون بما أن المخدم قادر على تغيير المعرف باستخدام ترويسة **Set-Cookie**.

أما في التطبيقات القديمة حيث من الصعب التعديل فإنه هنالك العديد من الحلول المقترحة سواءاً من جهة المخدم أو الزبون.

بحسب البيئة المستخدمة من جهة المخدم فإن تجديد المعرف من الممكن تضمينه في آلية إدارة الجلسة لبيئة التطوير أو تقديمه كـ[84] **server-side reverse proxy solution** . من الجانب الآخر هنالك تقنية حماية من جهة الزبون تدعى **Serene** [82] تقدم الحماية للمستخدم من دون تدخل المخدم.

5.2.2 الحالة الحالية:

العديد من بيئات التطوير تدعم تجديد معرف الجلسة ولكن ذلك يتطلب تدخل مباشر من المطور ليفعّل هذا السلوك. بالإضافة لذلك تفعيل خاصية **httpOnly** يمكن له منع الكثير من الهجمات الممكنة.

5.2.3 أفضل الممارسات:

أفضل ممارسة للحماية من هجمات تثبيت الجلسة هي تجديد معرف الجلسة في كل تغيير لامتيازات المستخدم داخل التطبيق. هذا يضمن بشكل فعال أن مستوى الامتياز الجديد لا يمكن الوصول إليه باستخدام معرف الجلسة الأصلي، وبالتالي منع هجمات تثبيت الجلسة. بالإضافة إلى ذلك، يجب دائماً تطبيق خاصية **HttpOnly**، والتي تمنع تعديل المعرفات باستخدام الجافا سكريبت أو الترويسات.

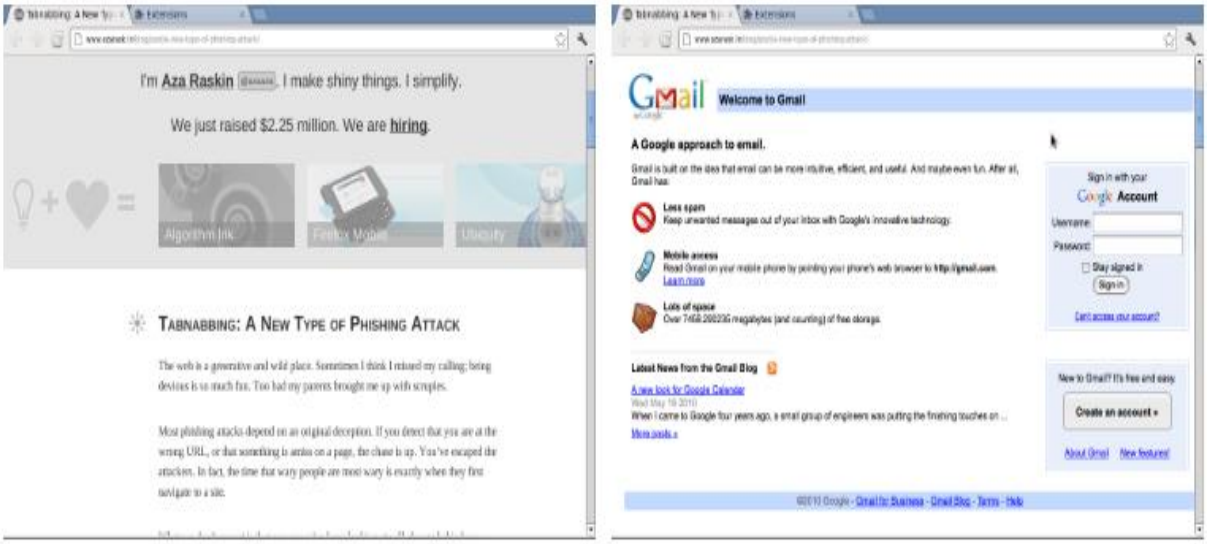
5.3 توثيق المستخدم باستخدام بيانات مسروقة

باستخدام بيانات اعتماد المستخدم، يمكن للمهاجم انتحال شخصية المستخدم نحو التطبيق المستهدف، مما يؤدي إلى تحكم كامل في بيانات وإجراءات الضحية. في شبكة الويب الحديثة المترابطة، غالبًا ما يسمح اختراق أحد الحسابات بالتصعيد نحو حسابات أخرى وتهديد وجود الضحية بالكامل عبر الإنترنت [85], [86]. غالبًا ما يستخدم المهاجمون تقنيات الهندسة الاجتماعية **Social Engineering** لخداع الضحايا للتنازل عن بيانات اعتمادهم عن طيب خاطر. المثال الأكثر شيوعًا لمثل هذا الهجوم هو التصيد الاحتيالي **Fishing**، حيث يستفيد المهاجم من عجز المستخدم عن تمييز صفحة شرعية عن صفحة تبدو شرعية ولكنها في الحقيقة مزورة. عن طريق جذب المستخدم إلى الصفحة المزورة، على سبيل المثال مع رسالة بريد إلكتروني "عاجلة" مصممة بعناية، يتم خداع المستخدم لإدخال بيانات اعتماده، مما يؤدي إلى إرسالهم إلى المهاجم. يمكن إجراء هجمات التصيد الاحتيالي على نطاق واسع وصغير، اعتمادًا على أهداف المهاجم. تعتبر الهجمات واسعة النطاق عامة جداً ويسهل اكتشافها عموماً. تستهدف الهجمات الصغيرة الحجم، والمعروفة أيضاً باسم التصيد بالرمح **Spear Fishing**، أفراداً أو شركات محددين ويصعب اكتشافها.

أحد أنواع هجوم التصيد التقليدي هو **TabNabbing**[87]. في **TabNabbing** (كما هو موضح في الشكل الآتي)، يتم إغراء المستخدم بزيارة موقع ضار، ومع ذلك يبدو غير ضار. إذا أبقى المستخدم موقع المهاجم مفتوحًا واستخدم علامة تبويب أخرى في متصفحه للتصفح إلى موقع ويب مختلف، فإن الصفحة المزورة تستفيد من عدم تشتت انتباه المستخدم (يمكن الوصول إليه من خلال **JavaScript** ك

window.onBlur) لتغيير مظهرها (اسم الصفحة ، ايقونة الموقع ومحتوى الصفحة) لتبدو متطابقة مع شاشة تسجيل دخول لموقع مشهور ما. عندما يعود المستخدم مرة أخرى إلى علامة التبويب المفتوحة، ليس لديه أي سبب لإعادة فحص عنوان **URL** الخاص بالموقع المعروض، لأنه فعل ذلك بالفعل في الماضي. يفصل هذا النوع من التصيد الاحتيالي فتح موقع ما عن هجوم التصيد الفعلي ويمكن، من الناحية النظرية، حتى خداع المستخدمين الذين لن يقعوا ضحية لهجمات التصيد التقليدية.

في جوهرها، تستفيد هجمات الهندسة الاجتماعية من عجز المستخدم عن اكتشاف تطبيق ويب احتيالي، وهذا ما يزال يمثل مهمة صعبة في الويب الحديثة. بالإضافة إلى ذلك، تعتبر بيانات اعتماد المستخدم ذات قيمة كبيرة للمهاجمين، كما يمكن بسهولة نقل بيانات الاعتماد التقليدية القائمة على اسم المستخدم / كلمة المرور وغالبا ما يتم إعادة استخدامها وتخزينها بشكل غير آمن.



شكل 5-4

*في هجوم **TabNabbing** ، يقوم المهاجم بتبديل علامة تبويب غير ضارة (يسار) إلى صفحة تصيد (يمين) ، مما يجنبه من الاكتشاف عند قيام المستخدم بالتحقق من **URI** لعلامة تبويب تم تحميلها حديثًا*

5.3.1 طرق المواجهة:

الآلية الحديثة للحد من تأثير سرقة بيانات باستخدام الهندسة الاجتماعية هو استخدام المصادقة متعددة العوامل **Multi Factor Authentication**. في عملية المصادقة متعددة العوامل، لم يعد التطبيق يعتمد على معرف واحد، مثل مجموعة من بيانات الاعتماد، ولكنه يتطلب عوامل إضافية، مثل الرمز المميز الذي

تم إرساله إلى هاتف المستخدم عبر رسالة نصية، وهو رمز تم إنشاؤه بواسطة جهاز مخصص [90]، بطاقة ذكية، بصمات...

المصادقة متعددة العوامل تجعل بيانات الاعتماد التقليدية أقل قيمة، لأن أحد عوامل المصادقة الإضافية هو جهاز خارج النطاق، خارج عن سيطرة المهاجم. ومع ذلك، يقدم إدخال عوامل مصادقة إضافية أيضاً مخاوف إضافية. على سبيل المثال، إذا كان الهاتف الذكي الخاص بالمستخدم يعمل كعامل ثانٍ في عملية المصادقة، فستظهر مشكلة عند سرقة الهاتف. وبالمثل، غالبًا ما تُعتبر القياسات الحيوية **biometrics** بديلاً قابلاً للتطبيق لكلمة المرور [89][88] ، ولكنها تمتلك خصائص مختلفة مقارنة ببيانات الاعتماد التقليدية. على سبيل المثال، يتم ترك بصمات الأصابع في كل مكان، ويمكن بسهولة خداع قارئ البصمة [90].

بالإضافة إلى المصادقة متعددة العوامل، تعمل المواقع الرئيسة على تحسين إجراءات المصادقة الخاصة بها عن طريق إجراء فحوصات أمنية إضافية عند تسجيل الدخول من جهاز غير موثوق به. تسمح لك **Microsoft** و**Facebook** و**Google** بتسجيل أجهزة الكمبيوتر الموثوق بها، حيث يمكن استخدام مصادقة تقليدية تعتمد على اسم المستخدم / كلمة المرور. تتطلب جميع الأجهزة الأخرى مصادقة ثنائية مع رمز التحقق [92] ,[91].

منذ أكثر من عشرين عام وحتى الوقت الحالي يحاول المهاجمون إقناع المستخدمين بالتخلي طوعًا عن بيانات اعتمادهم [93]. تم إجراء العديد من الدراسات، في محاولة لتحديد سبب وقوع المستخدمين ضحية لهجمات التصيّد [95] ,[94] واقتُرحت حلول مختلفة، مثل استخدام "**page skinning**" في الموقع [96]، أشرطة أدوات الأمان [97] واستخدام المعرفة بالنشاط السابق [98]. أخيرًا، يمكن للمستخدمين أيضاً تثبيت الإجراءات المضادة من جانب العميل لحماية أنفسهم من التصيّد [99] و**TabNabbing** [100].

5.3.2 الحالة الحالية:

في الممارسة العملية، تعد بيانات الاعتماد والمعلومات المالية المسروقة أحد الموارد القيمة، كما يتضح من ارتفاع الطلب في الأسواق السرية [101]. لمنع إساءة استخدام بيانات الاعتماد، تقدم مواقع الويب الرئيسية مصادقة قوية متعددة العوامل، جنبًا إلى جنب مع الأجهزة الموثوقة، مما يقلل بشكل فعال معظم المخاطر المرتبطة بسرقة بيانات الاعتماد. بالإضافة إلى ذلك، يقدم اللاعبون الرئيسيون، مثل **Google**

و**Facebook**، أيضاً حلول تسجيل الدخول الواحد "**single-sign on**"، مما يسمح للمواقع الأخرى بالاستفادة من إجراءات المصادقة الآمنة. على الجانب الآخر، لا تزال العديد من المواقع الأصغر تستخدم بيانات الاعتماد التقليدية.

لسوء الحظ، فإن مكافحة التصيد بطريقة آلية أمر صعب، وهذا هو السبب في أن آليات مكافحة الخداع التي تم نشرها حاليًا في المتصفحات الشائعة تستند جميعها إلى الاستبعاد وفق القائمة السوداء [102] يتم إنشاء القوائم السوداء تلقائيًا بواسطة العناكب الآلية، التي تبحث عن صفحات التصيد الاحتيالي على الويب [103]. وبالمثل، تستخدم الشركات والمؤسسات المالية الكبرى شركات أمنية تبحث يدوياً عن صفحات الخداع وانتحال الهوية، مما يتيح الإزالة السريعة لها.

الفصل السادس

الهجوم على السياق من جهة الزبون

6 الهجوم على السياق من جهة الزبون

في هذه الفقرة، نلقي نظرة على ثلاث هجمات يمكن أن تؤدي إلى سيطرة المهاجم على سياق تنفيذ التطبيق. الهجوم الأول هو البرمجة العابرة للموقع(**XSS**)، حيث يقوم المهاجم بإدخال كود **JavaScript** في تطبيق الضحية. بعد ذلك، نغطي الهجمات التي لا تتضمن نصًا برمجيًا، حيث يتم أيضاً حقن المحتوى في التطبيق المستهدف، ولكن المحتوى ليس كود برمجي. أخيرًا، نتحقق من مخاطر تضمين ملفات جافا سكريبت خارجية، وهو أمر شائع جداً ولكنه أيضاً يجعل التطبيق عرضة للهجمات.

6.1 البرمجة العابرة للموقع Cross-Site Scripting

يتمكن المهاجم من تنفيذ كود جافا سكريبت خاص به ضمن سياق التنفيذ للتطبيق مما يعطيه كامل صلاحيات كود التطبيق. أي يستطيع النفاذ لجميع بيانات المستخدم ومصادره والواجهات البرمجية الأخرى التي يقدمها المستعرض بالإضافة إلى القدرة على إرسال طلبات إلى المخدم.

المشكلة خلف هجوم الـ**XSS** هو فشل التطبيق المستهدف في التعرف على الكود المضاف مما يسمح بتنفيذه.

إمكانية وضع الكود في أي مكان من الصفحة ومحاولة المستعرض تصحيح الأخطاء النحوية بدلاً من رفضها أو تجاهلها يساعد على الاستغلال السهل لنقاط الضعف.

لقد تراجعت حدة هذا النوع من الهجمات بفضل الدراسات والتقنيات، فبعد أن كانت في المرتبة الثالثة ضمن قائمة **OWASP Top 10** [104] لعام 2013 تراجعت للمرتبة السابعة في قائمة عام 2017. ولكنها ما تزال قائمة وتهدد حتى الشركات الكبرى الواعية أمنياً مثل غوغل [105].

لدى المهاجم العديد من الاستراتيجيات لحقن حمولة في التطبيق الهدف:

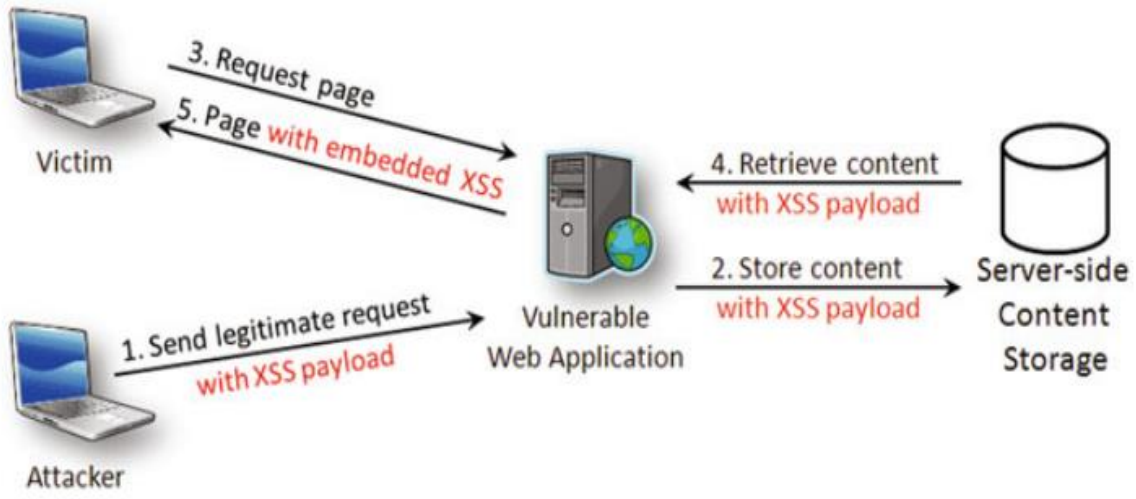
تتمثل الطريقة الأولى في تعديل الـ**URI** لإدخال كود برمجي ضمن الطلب الذي بالتالي ستتم معالجته من قبل الكود البرمجي من جهة الزبون، وهكذا سيتم تنفيذ كود المهاجم إلى جانب كود التطبيق الشرعي. يُعرف هذا النوع من هجمات **XSS** باسم البرمجة العابرة للموقع المستندة إلى الـ**DOM** أو " **XSS type 0**"

تتكون الفئة الثانية من هجمات **XSS** من خداع المخدم ليقوم بتضمين كود المهاجم في رده. على سبيل المثال، إذا قام المهاجم بجعل متصفح الضحية يزور عنوان **URI** كما في الشكل الآتي، فسوف يعكس المخدم قيمة الوسيط الممرّر في العنوان في ضمن الرد، حيث سيتم تنفيذها كجزء من الصفحة المطلوبة. يُعرف هذا النوع باسم **reflected XSS** أو **XSS** المنعكس أو "**XSS type 1** "

```
http://example.com/search.php?q=%3Cscript%3Ealert(%22XSSed!%22)%3C%2Fscript%3E
```

شكل 6-1
*مثال على **reflected XSS** أو **XSS** المنعكس*

أخيرًا، يمكن للمهاجم أيضاً تخزين كود الهجوم في بيانات التطبيق، على سبيل المثال عن طريق إخفائه في منشور في المنتدى أو تعليق في المدونة. كلما طلب الضحية صفحة تتضمن محتوى المهاجم، سيتم تضمين الكود الضار في الصفحة أيضًا. يُعرف هذا النوع من **XSS** باسم **XSS** المخزن أو "**XSS type 2** "، وهو موضح في الشكل الآتي:



*شكل 6-2 مثال على هجوم **XSS** المخزن أو **stored xss***

6.1.1 طرق المواجهة:

تعتمد تقنيات المعالجة التقليدية لهجمات **XSS** على تطهير المدخلات والمخرجات، ومنع أي مدخلات خطيرة من الوصول إلى المخرجات النهائية. حاولت تقنيات التعقيم هذه استبدال الأحرف الخطرة أو إزالتها ببساطة

مثل <> & "" أو التحقق من قائمة المحارف المسموح باستخدامها "**white List**"، لكن تقنيات التعقيم الحديثة تأخذ السياق بالحسبان، حيث تختلف طريقة المعالجة والتطهير من سياق إلى آخر.

تتضمن تطبيقات الويب الحديثة سياقات مختلفة، ولكل سياق تنسيقاته المختلفة وهجمات الحقن المخصصة. بعض الأمثلة هي عناصر **HTML**، وسمات عنصر **HTML**، وكود **CSS**، وكود **JavaScript**، وما إلى ذلك. توفر العديد من المكتبات المتاحة معقمات حساسة للسياق، وتخفف بشكل فعال من هجمات **XSS**.

- من الأمثلة الشائعة لتطبيقات Java، مشروع OWASP Java Encoder Project [106]، الذي يوفر العديد من عمليات التعقيم الخاصة بكل سياق
- يوفر HTML Purifier [107] التعقيم التلقائي لتطبيقات PHP، ويضمن أن المخرجات متوافقة مع المعايير.
- إن أتمتة عملية التعقيم الحساسة للسياق هي موضوع بحث نشط. يركز ScriptGard [108] على اكتشاف الاستخدام غير الصحيح لمكتبات التعقيم (مثل التعقيم غير المتطابق مع السياق)، وهو قادر على اكتشاف وإصلاح الموقع غير الصحيح للمطهرات في تطبيقات ASP.NET.
- يركز بحث آخر [109] على تحقيق التطهير الصحيح، وذلك عبر محرك تعقيم اوتوماتيكي حساس للسياق يمكن تطبيقه مباشرة ضمن أطر قولبة تطبيقات الويب "Web templating frameworks"

أما من ناحية المستخدم:

- جميع المستعرضات الحديثة كانت تقوم بمحاولة تعقيم الدخل والخرج لكن أثبتت التجارب أنه لا يمكن الاعتماد عليها وأنها حتى تفتح ثغرات جديدة قد يتمكن المهاجم من استغلالها أو قد تسبب فشل عرض تطبيقات سليمة كليا. حتى أن غوغل قامت بتاريخ 5\8\2019 بإزالة المعقم الخاص بها XSS Auditor [110] من المستعرضات التي تصدرها. وسبقتها إلى ذلك مستعرضات Edge من مايكروسوفت. كانت ترويسة X-XSS-Protection تستخدم لتحدد للمستعرض ما إذا كان يتوجب عليه إيقاف التعقيم والتصحيح التلقائي للكود عبر إعطاء القيمة صفر لهذه الترويسة (X-XSS-Protection: 0) ويمكن أيضاً للتطبيقات الطلب من المستعرض عدم القيام بعملية التصحيح وإنما أن يوقف عرض الصفحة في حال وجود شيفرة برمجية مشكوك بأمرها عبر إرسال الترويسة التالية (X-XSS-Protection: 1; mode=block) وهذا قد يوقف فتح ثغرات جديدة لكنه قد يسبب فشل عرض التطبيق. حاليا التوجه قائم لإلغاء هذه الترويسة ولم يعد وجودها عاملاً إيجابياً في تقييم مدى أمان موقع ما.
- ينصح المستخدمون باستخدام إضافات للمستعرض مثل NoScript [26].

- خاصية sandbox في HTML5 قد تساعد في تقييد محتوى الـiframe مما يؤمن بعض الحماية، فهي توجب على المستعرض أن يمنع هذا المحتوى من إرسال أي بيانات ومن أن يقوم بتنفيذ كود جافا سكريبت وتمنع ما داخل الـiframe من النفاذ إلى سياق الصفحة وما إلى ذلك. يمكن للمطور أن يقوم بإعطاء مجموعة من الاستثناءات كأن يسمح بإرسال بيانات الإدخال (المثال في الشكل الآتي) أو يسمح بتنفيذ الجافا سكريبت.

```
<iframe src="demo_iframe_sandbox_form.htm" sandbox="allow-forms"></iframe>
```

شكل 6-3
*استخدام خاصية **sandbox** مع السماح للإطار بإرسال بيانات نماذج الإدخال*

- الاستخدام الصحيح لـ CSP [25] وهي سياسة مقادة من قبل المخدم ومطبقة من قبل المستعرض. يمكّن المطورين أو مديري التطبيق من أن يحددوا وبصرامة مصادر المحتوى الموثوقة مثل الشيفرات البرمجية أو أنماط التنسيق أو الصور... وهذا يمنع من تضمين أي كود من مصدر آخر.

كما أنها تقدم العديد من الميزات الأخرى مثلاً إذا كان المطور مضطراً لاستخدام أوامر جافا سكريبت سطرية **(inline)** أو أوامر تنسيق سطرية **(inline styling)** أو ملفات جافا سكريبت أو ملفات تنسيق من مصادر خارجية وهذا أمر خطر ولا ينصح به. فإن **CSP** تقدم تقنيتين تمكنان من الاستخدام السليم لما سبق هما استخدام رموز تقطيع **hash** أو رموز مميزة عشوائية **nonce**.

```
Content-Security-Policy: default-src 'self' ; script-src https://cdn.example.com;
style-src https://cdn.example.com; img-src https://cdn.example.com;
```

شكل 6-4
*تتيح هذه السياسة تحميل أي مادة عرض من الموقع المضيف والسماح فقط بتحميل البرامج النصية والأنماط والصور من **cdn.example.com** باستخدام **HTTPS***

```
Content-Security-Policy: default-src 'self'; script-src
'nonce-4AEemGb0xJptoIGFP3Nd'

<script type="text/javascript" nonce="4AEemGb0xJptoIGFP3Nd">
```

شكل 6-5
*التضمين السليم لكود جافا سكريبت الخارجية باستخدام **nonce***

```
Content-Security-Policy: default-src 'self'; script-src
'sha256-blLDIhKaPEZDhc4WD45BC7pZxW4WBRp7E5Ne1wC/vdw='
```

شكل 6-6
التضمين السليم لملفات الجافا سكريبت الخارجية باستخدام رموز التقطيع

- **استخدام** (SRI) **SubResource Integrity** [111]: وهي ميزة أمان تمكن المتصفحات من التحقق من أن الموارد التي تجلبها (على سبيل المثال ، من CDN) يتم تسليمها دون تلاعب. عن طريق السماح للتطبيق بتوفير رمز تقطيع يجب أن يتطابق مع المورد الذي سيتم جلبه.

```
<script src="https://example.com/example-framework.js"
      integrity="sha384-
      oqVuAfXRKap7fdgcCY5uykM6+R9GqQ8K/uxy9rx7HNQlGYl1kPzQho1wx4JwY
      8wC"
       crossorigin="anonymous"></script>
```

شكل 6-7
مثال لاستخدام خاصية ***SRI***

ويمكن استخدام الـ**CSP** لفرض وجود هذه الخاصية لجميع الموارد من نوع ما:

```
Content-Security-Policy: require-sri-for style;
```

شكل 6-8
استخدام ***CSP*** لفرض استخدام ***SRI***

- ترويسة HTTP X-Content-Type-Options: هي علامة تستخدم من قبل المخدم للإشارة إلى أن أنواع MIME المعلن عنها في حقول Content-Type لا ينبغي تغييرها. يسمح هذا بإلغاء

التعرف على نوع MIME (MIME Snifiing)، أو بمعنى آخر، إنها طريقة لقول إن مشرفي المواقع يعرفون ما الذي يقومون به.

تم تقديم هذه الترويسة من قبل **Microsoft** في **IE 8** كوسيلة لمشرفي المواقع لحظر التعرف على المحتوى الذي قد يحول أنواع **MIME** غير القابلة للتنفيذ إلى أنواع **MIME** قابلة للتنفيذ. منذ ذلك الحين، قامت المتصفحات الأخرى بتضمينها، حتى لو كانت خوارزميات التعرف على الـ **MIME** أقل عدوانية.

```
X-Content-Type-Options: nosniff
```

شكل 6-9 **استخدام ترويسة** *X-Content-Type-Options*

حيث سيتم حظر جميع الطلبات للملفات التي نوعها الحقيقي لا يوافق النوع المصرح عنه ضمن ترويسة الـ **Content-Type**.

6.1.2 أفضل الممارسات:

من الصعب للغاية كتابة التعليمات البرمجية الخاصة بك لتعقيم الدخل والخرج. لا يوصى أبداً بتعقيم البيانات التي يوفرها المستخدم يدوياً. بدلاً من ذلك، ينصح بشدة باستخدام نظم **templating** أو أطر عمل لتطوير المواقع توفر إمكانية التعقيم التلقائي الحساس للسياق. إذا كان هذا مستحيلًا بالنسبة لتطبيق ما فيجب استخدام المكتبات الموجودة والمعروفة والمتوافقة مع السياقات المستخدمة لتعقيم جميع البيانات التي يقدمها المستخدم وجميع البيانات الخارجية بالنسبة للتطبيق.

وكخط دفاعٍ ثانٍ ينصح بشدة باستخدام سياسة محتوى **CSP** مكتوبة بإحكام مع الانتباه للأخطاء الشائعة في استخدامها مع العلم أنه حالياً العمل جارٍ لإصدار النسخة الثالثة منها.

6.2 الهجمات من دون كود Scriptless Injection Attacks

هجمات الحقن غير البرمجي هي في الأساس نفس هجمات **XSS**، مع الاختلاف الأساس في أن الحمولة المحقونة ليست كود **JavaScript**. فمن خلال عدم إدخال كود جافا سكريبت، ينجح هذا الهجوم في تجاوز العديد من المعقمات وتقنيات التخفيف الأخرى. في هجوم الحقن غير البرمجي، يمتلك المهاجم مجموعة

متنوعة من أنواع المحتوى ليختار منها. أحد متجهات الهجوم هو حقن محتوى **HTML** [112]، مما يسمح للمهاجم بتعديل وجهة النماذج ، واستخراج الرموز الأمنية المخفية ، وما إلى ذلك.

كطريقة ثانية للهجوم حقن أنماط تنسيق **CSS** أو صورة **(SVG)**، مما يسمح للمهاجم باستخراج كلمات المرور من حقل الإدخال. في جوهرها، فإن منع الجافا سكريبت لا يحل القضية. باختيار الحمولة بعناية، لا يزال المهاجم قادرًا على تنفيذ عمليات حساسة، على الرغم من أنه يحتاج إلى أن يكون أكثر إبداعًا من هجوم **XSS** المباشر.

```
<img src='http://evil.com/log.cgi?          ← Injected line with a non-terminated parameter
...
<input type="hidden" name="xsrf_token" value="12345">
...
'                                           ← Normally-occurring apostrophe in page text
...
</div>                                      ← Any normally-occurring tag (to provide a closing bracket)
```

شكل 6-10

*كل كود الـ**HTML** الموجود بعد فاصلة الاقتباس الخاصة بعنوان الصورة وحتى الوصول إلى فاصلة اقتباس لإغلاق العنوان ...سيتم اعتباره جزءاً من عنوان الصورة وسيتم إرسال الطلب الى العنوان الموافق مع قيمة رمز الحماية*

6.2.1 طرق المواجهة:

كما في هجوم الـ**XSS** فإن أفضل طريقة للمعالجة هي التحقق والتعقيم للدخل والخرج، وكذلك فإن **CSP** تقدم العديد من الخيارات لتقييد المحتوى مهما كان نوعه كما شرحنا سابقاً.

6.3 تضمين كود غير موثوق Compromised Script Inclusions

ويدعى أيضاً **Cross Site Script Inclusion(XSSI)**. يتيح تضمين ملف **JavaScript** مسيطر عليه من قبل المهاجم تنفيذ التعليمات البرمجية التي يسيطر عليها المهاجم ضمن سياق تنفيذ التطبيق، مما يمنحه نفس الامتيازات التي يتمتع بها كود التطبيق نفسه. القدرات التي يمتلكها هذا الهجوم هي في الأساس مماثلة لهجوم **XSS**، ولكن طريقة الهجوم والقدرات المطلوبة لتنفيذه تختلف اختلافًا كبيرًا. يعد تضمين ملفات

JavaScript خارجية أمراً شائعاً على الويب. ومن الأمثلة على ذلك استخدام المكتبات الشعبية وإدراج كود الإعلان. التطبيق المستهدف يثق بهذه الأطراف الخارجية وبأنها تقدم خدمات آمنة. لسوء الحظ، في الواقع، هذا ليس هو الحال. على سبيل المثال، في عام 2014، تعرض موقع رويترز على الويب للخطر بسبب مزود إعلانات غير آمن [6]. ومن المعروف أن شبكات الإعلانات تدع الشفرات الخبيثة تتسلل بين الحين والآخر [113].

6.3.1 طرق المواجهة:

أفضل ما يمكن للتخفيف من الخطر الذي يهدد الملفات المخزنة على المخدم هو حماية المخدم وتطبيقه ضد الخصوم المحتملين. بالنسبة للتطبيقات التي تتطلب درجة عالية من التحكم في هذه الملفات، قد يكون من المفيد نسخ الملفات الخارجية إلى مخدماتها الخاصة، حيث يمكن حمايتها على النحو الأمثل. وبطبيعة الحال، فإن النسخة ليست سوى نسخة من كود الطرف الثالث ويجب أن تظل محدّثة مع الإصدارات الجديدة عند إصدارها. يمكن التخفيف من خطر التلاعب من خلال نشر التطبيق عبر **HTTPS**، بما في ذلك الملفات البعيدة المستلمة عبر اتصالات آمنة فقط. يجب تجنب صفحات الويب ذات المحتوى المختلط، حيث غالباً ما تشتمل الصفحة الآمنة على محتوى غير آمن، أي أن الصفحة الأساسية تحمّل عبر **https**، أي تكون مشفّرة، أما ملفات الجافا سكريبت والتنسيقات والصور وما إلى ذلك تكون محمّلة عبر **http**، أي أنها تكون غير مشفّرة. هنالك العديد من الدراسات للتضمين الآمن لملفات جافا سكريبت غير الموثوقة، مما أدى إلى عدة مقترحات. يمكن تحقيق تحكم دقيق في سلوك الملف المضمن بمراقبة نقاط مرجعية سطرية باستخدام مغلفات (حاويات) أمنية مكتوبة بلغة جافا سكريبت [115][114]، وكذلك باستخدام تقنيات العزل (**sandboxing**) التي تفرض سياسة تنفيذ من خلال مراقبة النقاط المرجعية التقليدي [118]–[116].

وكما ذكرنا في الحديث عن طرق مواجهة الـ**XSS** فإن الـ**CSP** تمكننا من فرض قيود على استخدام الملفات الخارجية باستخدام رموز التقطيع التي تفرض على المستعرض التأكد من سلامة الملف المضمن.

الفصل السابع

الهجمات على جهاز الزبون

7 الهجمات على جهاز الزبون

غطت الفصول السابقة الهجمات التي كانت تزداد قرباً من الضحية وأثراً عليه فصلاً بعد فصل. في هذا الفصل، نغطي متجهات الهجوم التي تؤدي إلى تهديد جهاز المستخدم. هذا التهديد له تأثير كبير، حيث أن العديد من الإجراءات المضادة التي تمت تغطيتها سابقاً تعتمد على بيئة موثوق بها من جانب المستخدم، والتي لم يعد من الممكن ضمانها إذا تم اختراق المستعرض أو الجهاز. سنغطي اثنين من متجهات الهجوم الهامة ضد جهاز المستخدم. أولها يستخدم تقنيات التنزيل العرَضي، حيث يتم خداع الضحية ليقوم بتحميل البرامج الضارة. تستغل البرامج الضارة بدورها ثغرة أمنية في المتصفح أو مكون إضافي مثل **Flash Player** أو بيئة تشغيل **Java** أو أي شيء آخر قام المستخدم بتثبيته. الهجوم الثاني يستخدم ملحقات المستعرضات الضارة، والتي عادةً ما تتمتع بدرجة عالية من التحكم في جهاز المستخدم.

7.1 التنزيل العرَضي Drive-By Downloads

يتضمن تقديم صفحة ويب بسيطة الكثير من المكونات من جانب المستخدم، مثل المستعرض ومحرك العرض **rendering engine** والإضافات **plugins** والملحقات**extensions** ، وكلها يمكن أن تحتوي على ثغرات أمنية.

هجمات التنزيل العرَضي شائعة على الويب، مدفوعةً بالاقتصاد السري للجريمة المنظمة. فكل كمبيوتر مخترق له ثمن، ويمكن استخدامه في أنشطة إجرامية، مثل ضمه إلى **botnet**، وما إلى ذلك.

وكالة الأمن القومي الأمريكية (**NSA**) اتبعت نفس النهج في برنامج **FOXACID** الخاص بها [119]، حيث قامت بالسيطرة على أجهزة المستخدمين بهدف جمع المعلومات الاستخبارية. من الناحية الفنية، تتم إعادة توجيه المستخدم إلى مخدم **FOXACID**، حيث تحدد الخوارزمية المهارات الفنية للمستخدم. بناءً على درجاتهم، يحصلون على نوع معين من البرامج الضارة. يحصل المستخدمون الأذكياء تقنيًا على برامج ضارة غبية، بينما يحصل المستخدمون الأقل قدرة من الناحية الفنية على البرامج الضارة المتقدمة. السبب وراء هذا القرار هو أن هذا الأخير سيكون أقل عرضة للكشف عن البرامج الضارة، والتي تعدّ مورداً مهمّاً ومكلفاً.

يمكن للمهاجم محاولة خداع المستخدم لتثبيت برامج ضارة بشكل صريح، على سبيل المثال، من خلال تقديمها كحزمة مكافحة فيروسات. يحدث هجوم التنزيل من خلال عدة خطوات. أولاً، المهاجم يضع كود جافا سكريبت الذي سيؤدي إلى بدء التنزيل من المخدم. يمكن للمهاجم استخدام مخدمه الخاص، أو يمكنه استغلال مخدم آخر من خلال ثغرة أخرى، مثل **XSS**، أو باستخدام الهجوم من جانب المخدم، مثل هجوم **SQL Injection**. عند تحميل كود **JavaScript** هذا في مستعرض الضحية، سيتصل بخدمة تقوم بإعادة توجيه المستخدم نحو مخدم مناسب، وفقًا لنظام التشغيل المكتشف وإصدار المستعرض والإضافات المتاحة. إذا نجح مخدم الاستغلال في تقديم برنامج ضار مطابق للثغرة الأمنية المستهدفة، فسيتم تنزيله، وتثبيته على جهاز المستخدم. بمجرد التثبيت، يتم التحكم في البرامج الضارة عمومًا باستخدام نظام تحكم وإصدار أوامر. من الناحية الفنية، يمكن أن يكون استغلال الثغرة الأمنية في جانب المستخدم بسيطًا مثل تقديم صورة مصممة خصيصاً، بهدف استغلال ثغرة أمنية في محرك العرض [120]، أو استغلال ثغرة أمنية في مكون إضافي [121].

في جوهرها، تُعد هجمات التنزيل العرَضي أحد أنواع هجمات الكود الخارجي التقليدية، مثل هجمات تجاوز سعة المخزن المؤقت [122]. الكود المستغل مسؤول عن معالجة محتوى الويب، الذي يأتي من مصادر مختلفة، مما يجعل من السهل إدراج محتوى ضار في مكان ما على طول الطريق، وبالتالي استغلال التعليمات البرمجية الضعيفة من جانب العميل.

7.1.1 طرق المواجهة:

تتمثل إحدى تقنيات المعالجة الفنية في عزل بيئة تنفيذ إضافة المستعرض في صندوق رمل على مستوى العملية، مما يجعل الحصول على وصول كامل على مستوى النظام أكثر صعوبة. هذه التقنية مستخدمة في الوقت الحالي بواسطة جميع المتصفحات الرئيسية من أجل تشغيل مشغل الفلاش ذي السجل الأمني المهزوز على الويب.

تتمثل إحدى تقنيات التخفيف غير الفنية في تحديث الإضافات المثبتة بشكل متكرر، للاستفادة من تحديثات الأمان. هناك تحسن كبير [123] في هذا المجال هو عمليات التحديث التلقائي، إما التي تنشرها الإضافات نفسها، أو عن طريق دمجها في المتصفح. مثال على الطريقة الأولى هو إضافة **Java**، التي تقوم بتثبيت آلية التحديث التلقائي الخاصة بها، ومثال على الثانية هو **Google Chrome** مع إضافة **Flash**، حيث يتم تجميعهما وتحديثهما تلقائيًا عند الضرورة.

يعد اكتشاف وتحليل البرامج الضارة مجالًا بحثيًا واسعاً، يغطي الكشف عن هجمات التنزيل العرَضي أو النماذج الاقتصادية لصناعة البرامج الضارة ومنعها. يركز مجال واحد من مجالات البحث على الكشف الساكن عن البرامج الضارة [125] ,[124]، بينما يعتمد الآخر على استخراج المواصفات.

تركز الأفكار البديلة على المهام الداعمة للكشف، مثل إزالة برامج التجسس الخبيثة[124].

يبحث باحثو الأمن أيضاً في النماذج الاقتصادية الأساسية وراء صناعة البرمجيات الخبيثة. تطور اقتصاد البرمجيات الخبيثة الخفي بسرعة، مع تقديم خدمة الدفع لكل تنزيل كعملة أساسية. تبحث دراسة مستفيضة [126]في مختلف عائلات البرامج الضارة، واستراتيجيات تجنب الاكتشاف، واستهداف بلدان محددة. كشفت دراسة أخرى تبحث في الاقتصاد السري لبرامج مكافحة الفيروسات المزيفة [127] أن ثلاث شركات كبيرة الحجم حققت إيرادات مجتمعة قدرها 130 مليون دولار. تخطو شركات مكافحة الفيروسات المزيفة خطوة أبعد من ذلك، وترصد بنشاط عمليات إعادة تحميل بطاقة الائتمان لعملائها المغفلين.

7.1.2 الحالة الحالية:

لا تزال هجمات التنزيل العرَضي في ارتفاع، وتعتبر تهديدًا مهمًا لأمان المستخدم على الويب.

7.1.3 أفضل الممارسات:

أفضل ممارسة لمستخدمي الويب هي تقليل عدد الإضافات المثبتة إلى الحد الأدنى وتحديث جميع برامج المستخدم، بما في ذلك نظام التشغيل وبرامج التشغيل والمتصفحات والإضافات، إلخ. بالإضافة إلى ذلك، استخدام خاصية **click-to-play** في المستعرضات يمكن أن يقلل من سطح الهجوم بشكل كبير. في بيئات الشركات، يجب التحكم في البرامج الموجودة على الأجهزة وتحديثها قدر الإمكان.

يجب أن يضمن مطورو الويب أن تطبيقاتهم محمية بشكل جيد، خاصة ضد هجمات الحقن، ومنع الاستفادة من موقع الويب الخاص بهم كمنصة توزيع البرمجيات الخبيثة. يجب اختيار مكتبات الجهات الخارجية ومزوديها بعناية، حيث قد يؤدي استغلال مزود المكتبة أيضاً إلى تهديد جميع تطبيقات الويب المعتمدة عليه.

7.2 ملحقات المستعرض الخبيثة

توفر ملحقات المستعرضات كوداً برمجياً إضافياً يتم تشغيله داخل المستعرض، وتتمتع بامتيازات أكبر بكثير من كود الويب التقليدي. يمكن للمهاجمين القادرين على اختراق الملحقات المشروعة أو خداع المستخدمين لتثبيت الملحقات الضارة، أن يكتسبوا القدرة على اختراق جميع تطبيقات الويب التي تعمل داخل المتصفح، وربما يكونون قادرين على اختراق نظام الضحية.

أصبحت ملحقات المستعرضات شائعة جدًا، وتقريباً كل مستخدم لـ**Firefox** و **Chrome** يستخدمهم. تحتوي مستودعات **Firefox** و **Chrome** الرسمية على آلاف الملحقات، بعضها قد يكون ضارًا [128]، أو قد تتحول إلى ضارة بعد ذلك. في عام 2014، تم الكشف عن أحد الأمثلة من الحياة الواقعية حول كيف يمكن أن تصبح امتدادات المتصفح ضارة[129]. اشترى بائعو **Adware** العديد من ملحقات **Chrome** مما أتاح لهم التحكم الكامل في الملحقات. ثم قاموا بتعديلها لنشر الإعلانات في كل مكان ودفعوا التحديث من خلال سوق **Chrome** الإلكتروني، حيث وصل عدد المستخدمين إلى حوالي 30،000 مستخدم.

الهدف من التحكم في ملحق المستعرض هو تشغيل كود متميز يسيطر عليه المهاجمون داخل المستعرض. قد يتيح ذلك للمهاجم الوصول إلى الحالة الداخلية للمستعرض، وقد يكون عاملاً مساعداً يسمح بتصعيد الهجوم نحو تهديدٍ كاملٍ لجهاز العميل. نظرًا لأن ملحقات المستعرض تعمل ضمن مستوى امتياز أعلى، خارج سياسات أمان المستعرض التقليدية، وتتمكن من الوصول إلى مواقع متعددة ومعالجتها، فهي هدف جذاب ذو قدرة عالية للمهاجمين. يصبح استغلال ملحقات المستعرض الشرعية أمرًا ممكنًا عندما الملحق يعامل محتوى غير موثوق به بلا مبالاة. نظرًا لأن الإضافات تتم كتابتها غالباً باستخدام **JavaScript**، يمكن للمهاجم تنفيذ هجوم عن طريق الحقن النصي، على سبيل المثال، عندما تعالج الإضافة صفحة تم تحميلها في المستعرض. تؤدي معالجة مثل هذه المدخلات بلا مبالاة إلى تمكين هجوم الحقن النصي، مما يسمح للمهاجم بتنفيذ تعليمات برمجية عشوائية ضمن سياق الملحق (على غرار هجمات **XSS** في الفصل 6).

خداع المستخدم لتثبيت ملحق ضار يمكن القيام به بطرق مختلفة. إن أبسط طريقة هي ببساطة توفير الملحق على موقع ويب، على أمل خداع المستخدم في تثبيته يدويًا. هناك طريقة أخرى تتمثل في تقديم الإضافة عبر قنوات التنزيل الرسمية، مثل متجر ملحقات المستعرض. أخيراً، يمكن للمهاجم القوي أن يفسد متجر الإضافات بالكامل، مما يسمح له بالتنكر كملحق شرعي وشعبي.

في جوهرها، يمكن اختراق الملحقات عندما يفشل كود الملحق في تعقيم المحتوى غير الموثوق به بشكل كاف، مما يسمح للمهاجم بضخ الشفرة الضارة في سياق التشغيل ذي السماحيات المتعددة. المشكلة الأساسية المتمثلة في الإضافات الضارة هي الهندسة الاجتماعية، حيث يمكن خداع المستخدمين لتثبيت ملحق من مصادر غير موثوق بها.

7.2.1 طرق المواجهة:

تقنيات المعالجة محدودة إلى حد ما، وهي جزء من بنية المستعرض ومنصة متجر الملحقات. بشكل عام، يمكن معالجة تهديد الملحق الشرعي من خلال بنية شاملة للملحقات، حيث يتم عزل الكود بشكل صارم وتقتصر واجهة برمجة التطبيقات (**API**) على الأذونات. إن منع تثبيت الملحقات الخبيثة ليس سهلاً، ويمكن معالجته من خلال منع التحميل من خلال قنوات غير رسمية، وإجراء مراجعات الكود على الملحقات المنشورة عبر القناة الرسمية.

أدت الأبحاث إلى اقتراح نظام ملحق يستند إلى مبدأ أقل الامتيازات والعوالم المعزولة وأنظمة الأذونات [130]، وهو نموذج تم اعتماده كنظام ملحقات **Google Chrome**. أحد الأبحاث [131]في بنية أمان ملحق **Google Chrome**، خلص إلى أنه حتى مع وجود جميع القيود، يمكن للمهاجم اختراق العديد من الملحقات. ونتيجة لذلك، تم اقتراح ونشر دفاعات إضافية، مثل التطبيق الافتراضي لسياسة أمان المحتوى [25] على ملحقات **Chrome**. يستخدم بحث آخر أنظمة رسمية"**formal systems**" للتحقق من خصائص الأمان، وغالبًا ما يتم العثور على ثغرات أمنية وإصلاحها في هذه العملية. أخيرًا، في نتائج الأبحاث الحديثة، تم اقتراح طريقة تلقائية لاستنباط السلوك الضار في ملحقات المستعرض في [17]. تستخدم هذه التقنية ***HoneyPages*** والمنطق الضبابي لاكتشاف السلوك الضار، عثرت هذه الطريقة على عدة فئات من الملحقات الخبيثة، بعضها له أكثر من 5.5 مليون عملية تثبيت.

7.2.2 الحالة الحالية:

يتم تعريف الحالة الحالية للتهديد المفروض من الملحقات، بواسطة المستعرضات الشهيرة حاليًا. سنغطي المستعرضين الأكثر انتشاراً: موزيلا فايرفوكس وجوجل كروم، وكلاهما يتمتع بدعم واسع النطاق للملحقات، ويقدمان عددًا كبيرًا من الملحقات عبر قنواتهما رسمية. تدعم بنية **Firefox** الملحقات ذات السماحيات العالية، والتي لها حق الوصول إلى مجموعة كبيرة من واجهات برمجة التطبيقات التي يوفرها المستعرض،

وتقدم العديد من الخدمات، بالإضافة إلى الوصول إلى موارد المتصفح الداخلية ونظام التشغيل، مثل ملفات القراءة / الكتابة، وبدء عمليات جديدة، وما إلى ذلك. يمكن لمتصفحات **Firefox**، تحديد مكونات أساسية يمكن استدعاؤها من خلال واجهة برمجية. تخضع الملحقات في **Firefox** لقيود قليلة جدًا، ويمكنها بسهولة مشاركة وظائفها بين المكونات الأساسية والبرامج النصية التي تتفاعل مع محتوى الويب.

منصة إضافات **Mozilla**، التي تسمى **Mozilla Add-Ons**، هي القناة الرسمية لتقديم الملحقات للمستخدمين. يضمن أن تكون الإضافات المنشورة قد خضعت لمراجعة من قبل محرر مكلف بمراجعة وظائف الإضافة وسلوكها. يدعم **Firefox** أيضاً تثبيت ملحقات غير رسمية من مواقع عشوائية ولكن ليس من دون موافقة صريحة من المستخدم.

تعتمد بنية **Chrome** على مبادئ الامتياز الأقل والعوالم المعزولة والأذونات. تحتوي الإضافات على مكون أساسي، يعمل بشكل منفصل عن البرامج النصية للمحتوى، والذي يتفاعل مع محتوى الويب الفعلي. التواصل بين كلا السياقين متاح من خلال واجهة برمجة تطبيقات **Web Messaging** [21]. بالإضافة إلى ذلك، جميع الامتدادات لها مساحة اسم مميزة ومعزولة عن بعضها البعض. تعد واجهات برمجة التطبيقات الخاصة بالمتصفح التي يقدمها **Chrome** أكثر محدودية من خاصة **Firefox**، وخاصةً إذا حاولت الوصول إلى نطاق خارج بيئة المتصفح، مثل نظام التشغيل. علاوة على ذلك، يتعين على ملحقات **Chrome** أن تطلب صراحةً مجموعة من الأذونات لمجموعة محددة من مواقع الويب عند التثبيت. بدون الحصول على الأذونات اللازمة، يتعذر الوصول إلى العديد من واجهات برمجة التطبيقات، مما يمنع الملحقات من زيادة قوتها داخل المستعرض. منصة ملحقات **Chrome**، المسماة **Chrome Web Store**، هي قناة **Chrome** الرسمية لتوزيع الإضافات. لا يقوم **Chrome** بإجراء أية مراجعات، مما يجعل سوق الويب نظامًا قائمًا على السمعة، حيث يُتوقع من المستخدمين تقديم تقارير عن سوء الاستخدام في حالة وجود سوء تصرف. لا يدعم **Chrome** الامتدادات غير الرسمية، إلا من المجلدات المحلية في وضع مطور البرامج. يقوم **Chrome** أيضاً بتعطيل الإضافات افتراضيًا في وضع التصفح الخاص "**incognito mode** "، الذي يطلق عليه وضع التصفح المتخفي، نظرًا لأنها قد تشكل خطراً على خصوصية المستخدم. ومع ذلك، يمكن تمكينها بشكل صريح في وضع التصفح الخاص، إذا رغبت في ذلك.

نختتم بمناقشة **GreaseMonkey**، وهو ملحق قابل للتوسع. **Greasemonkey** هو ملحق لفايرفوكس يسمح للمستخدمين بتشغيل البرامج النصية المخصصة على أي صفحة ويب، مما يتيح لهم إزالة الميزات غير المرغوب فيها من تطبيقات الويب، أو إضافة ميزات إضافية مطلوبة. لدى **Greasemonkey** سوق

نصوص، حيث يستضيف أكثر من 140،000 سكريبت، تم تحليلها في دراسة حديثة [132]. من بين 592 مخطوطة وصفت بأنها برامج ضارة تحاول 126 منها في الحقيقة سرقة البيانات الخاصة. يكشف تحليل الأمان الإضافي لـ 86،358 من البرامج النصية عن 1736 نصًا برمجيًا يحوي الثغرات الأمنية **XSS** المستندة إلى (**DOM**). في 944 حالة، يمكن استخدام هذه الثغرات الأمنية من قبل المهاجمين لتشغيل ثغرة أمنية **XSS** على أي موقع، وذلك ببساطة عن طريق إرسال الضحية **URI** معد.

7.2.3 أفضل الممارسات:

من أفضل الممارسات لأي مستخدم على الويب هي الحدّ من عدد الإضافات والملحقات إلى الحد الأدنى، وإلغاء تثبيت، أو تعطيل الملحقات غير المطلوبة. بالإضافة إلى ذلك، عند استخدام شكل من أشكال التصفح السري، قد يكون من المفيد تعطيل الإضافات، نظرًا لأنها يمكن أن تعرض الطبيعة الخاصة لوضع التصفح للخطر [133]. في أنظمة الشركات، من المنطقي منع تثبيت الملحقات تماماً.

يجري العمل حالياً على ترويسة **feature-policy** [134] وهي آلية تسمح للمطورين بتمكين وتعطيل استخدام ميزات المتصفح وواجهاته البرمجية المختلفة، وهي في حال انتشارها تقدم خيارات أمان تحمي التطبيق من الملحقات والإضافات الخبيثة.

```
Feature-Policy: fullscreen 'none'; geolocation 'none'
```

شكل 7-1 تعطيل استخدام واجهات برمجة التطبيقات لملء الشاشة وتحديد الموقع الجغرافي

```
Feature-Policy: camera https://other.com; microphone https://other.com
```

*شكل 7-2 تستضيف شركة ما تطبيقًا على **"https://example.com"** وتريد تعطيل إدخال الكاميرا والميكروفون لأصلها مع تمكينه للموقع (**https://other.com**)الذي يقوم بتضمين موقعها.*

الفصل الثامن

دراسة الواقع الأمني للمواقع السورية

8 دراسة الواقع الأمني للمواقع السورية

لتقييم مدى أمان موقع ما، غالباً ما يلجأ مدراء هذه المواقع إلى الشركات الاستشارية للقيام بعمليات فحص للاختراق **penetration testing** وعمليات مراجعة للكود البرمجي. إلا أنه من الصعب لطرف خارجي كالحكومة أو المنظمات الرقابية القيام بتقييم مدى أمان موقع ما خارجياً، خاصة إذا كان هذا التقييم يجري على مجال واسع، كأن يتضمن عدداً كبيراً من المواقع التابعة لبلد ما او قطاع صناعي معين. هذا النوع من التقييم ضروري بما أن المواطنين يعتمدون بشكل متزايد يوماً بعد يوم على تطبيقات محددة. وهذا مشابه مثلاً لضرورة التقييم الإلزامي للسلامة الإنشائية للمباني من أجل حماية المواطنين من كوارث مستقبلية كان يمكن تجنبها.

لتقييم أمان موقع ما، تركز الأساليب الحالية عادةً على اكتشاف نقاط الضعف في الموقع. على سبيل المثال، تنشر **WhiteHat** تقارير سنوية حول إحصائيات أمان مواقع الويب [9]، مع إبراز نقاط الضعف العشرة الأكثر شيوعًا، ومناقشة متجهات الهجوم الجديدة. على النقيض من ذلك، لا تقتصر مقاربتنا على مراعاة الثغرات ونقاط الضعف الموجودة، ولكنها تأخذ بالحسبان أيضاً وجود آليات أمنية مطبقة على مواقع الويب التي تم تقييمها. يمكن اكتشاف وجود أو غياب كل من هذه الآليات بشكل ساكن غير فاعل ويمكن استخدامه كمؤشر على "الوعي الأمني" لكل موقع على حدة. بالإضافة إلى ذلك، من أجل أن نكون قادرين على مقارنة المواقع عن طريق الوضع الأمني الخاص بهم نقترح أيضاً نظامًا لتقييم مستوى أمان موقع الويب، وبناءً على نظام التقييم، نقدم تحليلًا أمنيًا مقارنًا للمواقع السورية.

في هذه الأطروحة نحاول تقديم نظرة عامة عن مدى تبني المواقع السورية للآليات الأمنية من جهة الزبون وكذلك تقديم نموذج مرجعي للحالة الحالية لأمن الشبكة.

كما سنطرح نظاماً لتسجيل النقاط الأمنية **security scoring system** لتقدير مستوى الأمان الذي يقدمه كل موقع. وكذلك ندرج التحديات التي واجهناها في تجربتنا، ونقدم التوجهات الممكنة لمجالات البحث في المستقبل.

8.1 جمع البيانات:

لاستخراج المواقع السورية ذات النطاق "**sy.**" قمنا أولا باستخراج المواقع السورية من قائمة **ALEXA Top Million** إلا أنها بعد التصحيح وإزالة أسماء المواقع التي لم تعد موجودة أو تغير نطاقها لم تتجاوز النتيجة التسعين موقعاً. قمنا بعدها باستخدام أداة **APIFY** لاستخراج المزيد من المواقع. وقمنا بالتركيز في بحثنا على المواقع الحكومية "**gov.sy.**" والمواقع التعليمية "**edu.sy.**". بعد حل التقاطعات بين النتائج واستبعاد النتائج غير الصالحة، حصلنا على ما مجموعه 134 موقعاً، بينها 57 موقع حكومي و24 موقع تعليمي.

8.2 أدوات الدراسة:

استخدمنا أداة APIFY [135] لاستخراج بعض أسماء المواقع السورية.

واخترنا لغة Python لبرمجة أداة فحص إعدادات المواقع. لأنها تقدم العديد من المكتبات التي تساعد في مجال استخراج المعلومات من المواقع web scraping مثل:

- مكتبة Requests [136] لإرسال وتلقي الطلبات من المخدمات بشكل مؤتمت.
- مكتبة Beautiful soup [137] التي تقوم بمعالجة ملفات html و xml واستخراج المعلومات منها.
- مكتبة SSLyze [138] من أجل مسح إعدادات الـSSL للمواقع قيد الدراسة.

8.3 آلية الدراسة:

من أجل مقارنة مستوى الوعي الأمني للمواقع سنقوم بتطبيق نظام نقاط أمني يعطي لكل موقع علامة أو درجة رقمية. الدرجة المعطاة للموقع تنقسم إلى قسمين: درجة إيجابية تمثل تبني الموقع لآلية أمنية ما ودرجة سلبية لوجود ثغرات أو نقاط ضعف مثل تضمين ملفات جافا سكريبت خارجية أو تطبيق الـ**SSL** بطريقة غير آمنة.

لكل آلية أمنية ونقطة ضعف نقوم بإسناد درجة ايجابية أو سلبية مثقَلة. الدرجة الكلية الإيجابية أو السلبية لموقع ما يتم الحصول عليها بجمع الدرجات الجزئية المثقلة.

تحقيقا لأخلاقيات العمل، سنقوم بإجراء التجارب بطريقة غير فاعلة لا تؤثر على عمل المواقع. وستنحصر اختباراتنا بالآليات التي يمكن كشفها بهذه الطريقة. كحصيلة لدينا 12 آلية حماية وأربعة نقاط ضعف لكل

موقع. في الفقرات التالية سنصف بشكل مختصر هذه الآليات والنقاط ونتوسع بالشرح عن نظام التقييم الذي اعتمدناه.

8.4 الميزات الأمنية تحت الدراسة:

في هذه الأطروحة سنركز على عدة ميزات أمنية يمكن كشفها بشكل ساكن من دون العودة إلى التطبيق أو المخدم. وسنصنف هذه الميزات ضمن ثلاث فئات:

8.4.1 الميزات التي تساهم في تأمين الاتصال

- النقل الآمن للبيانات عبر HTTPS
- **استخدام خاصية Secure لملفات تعريف الارتباط:** تمنع نقل ملف تعريف الارتباط عبر قناة غير مشفرة مما يوقف هجمات التنصت.
- **HTTP Strict-Transport-Security (HSTS):** والتي تسمح للمخدم أن يطالب المستعرضات التي تدعم هذه التقنية بأن تتصل معه باستخدام https حصرا. يمكن للمخدم تفعيل الـ HSTSببساطة عبر تضمين الترويسة Strict-Transport-Security في الردود مصرحاً بطول حياة حماية الـHSTS . تفيد هذه الآلية بإيقاف هجوم تعرية الـSSL
- **HTTP Public Key Pinning (PKP):** هي ميزة تطلب من المستعرض أن يربط مفتاح تشفير عام محدد مع مخدم ويب محدد لتقليل مخاطر هجمات MITM بشهادات مزورة.

8.4.2 الميزات التي تواجه هجوم البرمجة العابرة للموقع

- **خاصية HttpOnly:** تمنع النفاذ لملف تعريف الارتباط من خلال الجافا سكريبت
- **خاصية SameSite:** تسمح للمخدم بالتأكيد على أن ملفات تعريف الارتباط يجب ألا يتم إرسالها ضمن طلبات عابرة للمواقع. هذا التأكيد يسمح للمستعرضات بالتخفيف من خطر تسرب المعلومات لجهة غريبة، ويوفر بعض الحماية ضد هجمات CSRF

- **سياسة أمن المحتوى CSP:** وهي سياسة مقادة من قبل المخدم ومطبقة من قبل المستعرض. تمكّن المطورين أو مديري التطبيق من أن يحددوا وبصرامة مصادر المحتوى الموثوقة مثل الشيفرات البرمجية أو أنماط التنسيق أو الصور... وهذا يمنع من تضمين أي كود من مصدر آخر.

- **ترويسة X-Content-Type-Options(XCTO):** هي علامة تستخدم من قبل المخدم للإشارة إلى أن أنواع MIME المعلن عنها في حقول Content-Type لا ينبغي تغييرها. يسمح هذا بإلغاء التعرف على نوع MIME (MIME Snifiing)

- **استخدام Subresource Integrity (SRI):** ميزة أمان تمكن المتصفحات من التحقق من أن الموارد التي تجلبها (على سبيل المثال، من CDN) يتم تسليمها دون تلاعب. عن طريق السماح للتطبيق بتوفير رمز تقطيع يجب أن يتطابق مع مورد تم جلبه.

- **خاصية sandbox** قد تساعد في تقييد محتوى الـiframe مما يؤمن بعض الحماية، فهي توجب على المستعرض أن يمنع هذا المحتوى من إرسال أي بيانات ومن أن يقوم بتنفيذ كود جافا سكريبت وتمنع ما داخل الـiframe من النفاذ إلى سياق الصفحة وما إلى ذلك.

8.4.3 الميزات التي تمكّن من السماح بالتأطير بشكل آمن:

- **X-Frame-Options** (XFO): يتم استخدام هذه الترويسة ليحدد التطبيق ما إذا كان من المسموح لتطبيقات أخرى بوضعه ضمن <frame> أو <iframe> أو <embed> أو <object>.

- **ميزة frame_ancestors:** تم تصميم CSP في البداية للحماية من هجمات الـXSS وهجمات الحقن الأخرى. لكنها أيضاً تقدم خاصية frame-ancestors لتحديد المصادر التي يسمح لها بتضمين الصفحة ضمن <frame> أو <iframe> أو <embed> أو <object>.

بصرف النظر عن خاصية sandbox للأطر واستخدام رموز CSRF، يتم الوصول إلى جميع آليات الدفاع المذكورة أعلاه عبر ترويسات الاستجابة لطلبات الـ HTTP، وبالتالي يمكن اكتشافها بشكل مباشر عن طريق استخراج هذه الترويسات. من أجل فحص خاصية sandbox للأطر واستخدام رموز CSRF بحثنا عن جميع عناصر iframe وعناصر نماذج الإدخال في نص الاستجابة لكل صفحة ويب تم الوصول إليها.

8.5 نقاط الضعف تحت الدراسة:

سنقوم بتقييم نقاط الضعف التالية:

- تضمين ملفات جافا سكريبت خارجية غير موثوقة المصدر:

موقع الويب الذي يختار تضمين جافا سكريبت من مصادر خارجية غير موثوق بها، يضع نفسه أمام مجموعة من المشاكل الأمنية. الأبحاث التي أجراها نيكيفوراكيس وآخرون [12]. حددت أربعة أنواع مختلفة من الثغرات الأمنية المرتبطة بممارسات إدراج JavaScript غير الآمن عن بعد.

- تضمين محتوى مختلط mixed-content:

عند التحول إلى HTTPS، تفشل العديد من مواقع الويب في تحديث تطبيقاتها بالكامل، مما يؤدي إلى تضمين محتوى مختلط حيث يتم إرسال صفحة الويب الرئيسية عبر قناة HTTPS آمنة، بينما يتم تسليم بعض المحتوى الإضافي المتضمن في تلك الصفحة، مثل الصور والبرامج النصية، عبر اتصالات HTTP غير آمنة. ونتيجة لذلك، يمكن لمهاجم ما مهاجمة موقع الويب الذي يدعم بروتوكول TLS عن طريق اعتراض وتعديل أي محتوى مختلط يتم تحميله عبر HTTP

- التطبيق غير السليم لـ SSL:

استخدام SSL أمر حيوي بالنسبة لجميع المواقع، لكن التطبيق غير السليم لها يفتح ثغرات إضافية. في تقييمنا، نستخدم ماسح SSL سريعً يسمى sslyze للبحث عن مشكلات تطبيق SSL بما في ذلك دعم SSL V2.0 أو SSL V3.0 أو TLS 1 أو TLS 1.1.

- إيقاف خاصية X-XSS-Protection:

تستخدم لتحدد للمستعرض ما إذا كان يتوجب عليه إيقاف التعقيم والتصحيح التلقائي للكود عبر إعطاء القيمة صفر لهذه الترويسة (X-XSS-Protection: 0) ويمكن أيضاً للتطبيقات الطلب من المستعرض عدم القيام بعملية التصحيح وإنما أن يوقف عرض الصفحة في حال وجود شيفرة برمجية مشكوك بأمرها عبر إرسال الترويسة التالية (X-XSS-Protection: 1; mode=block) وهذا قد يوقف فتح ثغرات جديدة لكنه قد يسبب فشل عرض التطبيق. حاليا التوجه قائم لإلغاء هذه الترويسة ولم يعد وجودها عاملاً إيجابياً في تقييم مدى أمان موقع ما. ويفضل ترك الخيار للمستعرض بعد التطور الأمني الكبير الذي شهدته المستعرضات.

- استخدام برامج تخديم مهملة outdated server software:

من المهم الحفاظ على خوادم الويب محدثة، لأن المخدم القديم عادة ما يحتوي على ثغرات قد تؤدي إلى هجمات. في تقييمنا، بحثنا عن الإصدارات القديمة لمخدمات الويب الشهيرة Apache وMicrosoft-IIS وNginx مع التنبيه إلى أننا اعتمدنا على ترويسة sever لمعرفة إصدار المخدم والتي ينصح بأن تتم مراجعتها وحذف أي معلومات تفصيلية منها لأنها تقدم معلومات قد تساعد المهاجمين في العثور على ثغرات جديدة.

8.5.1 تفاصيل نظام التقييم للميزات الأمنية:

من أجل مقارنة مستوى الأمان بين مواقع الويب المختلفة، قمنا بتطبيق نظام تقييم لأمان الويب يوفر درجة أمان كمي لكل موقع، تعرّف درجة الأمان الإجمالية (**OverallScore**) كمتوسط مرجح لثلاث درجات جزئية:

الدرجة الكلية = $\frac{40}{100}$ درجة الاتصال الآمن+

$\frac{40}{100}$ درجة معالجة الـ**XSS** +

$\frac{20}{100}$ درجة التأطير السليم

الدرجات الجزئية هي متوسط مرجح لمجموعة من الميزات الأمنية المثقلة. يعتمد ثقل الميزة على مدى أهميتها الأمنية ومدى نضج هذه الميزة (قد تكون الميزة غير مستقرة الاستخدام أو جديدة نسبياً)

درجة الاتصال الآمن = $\frac{40}{100}$ لاستخدام **https** +

$\frac{25}{100}$ استخدام خاصية **secure** +

$\frac{25}{100}$ تفعيل **HSTS** +

$\frac{10}{100}$ تفعيل **PKP**

درجة التأطير الآمن = $\frac{100}{100}$ (ترويسة **XFO** أو ميزة **frame_ancestors**)

درجة معالجة **XSS** = $\frac{30}{100}$ لاستخدام **CSP** +

$\frac{20}{100}$ استخدام **http-only** +

$\frac{20}{100}$ استخدام **same-site** +

$\frac{10}{100}$ ترويسة **XCTO** +

$\frac{10}{100}$ خاصية **sandbox** +

$\frac{10}{100}$ استخدام **SRI**

8.5.2 تفاصيل نظام التقييم لنقاط الضعف:

نظام التقييم المستخدم لتقدير الحالة الأمنية للمواقع مبني على نظام CWSS (Common Weakness Scoring System) هذا النظام يقدم قياس كمي لنقاط الضعف في التطبيقات البرمجية ويستخدم بشكل رئيسي لإعطاء أولوية للنقاط الأكثر أهمية للإصلاح. فمثلاً عدم تعقيم البيانات المدخلة من قبل المستخدم قد يقود إلى هجوم XSS وقد يسمح للمهاجم باستخراج بيانات حساسة من المخدم، نقطة الضعف هذه لها تأثير كبير ويمكن استغلالها عن بعد لذا سيكون تقييمها أكبر من التطبيق غير السليم لـ SSL.

سبب اختيار هذا النظام له شقين. الأول أن CWSS عريقة ومستخدمة بشكل كبير لإعطاء تقييم كمي لنقاط الضعف، أي تمت مراجعتها بشكل كثيف ما يعطي لحد ما ضمانة أن التقييم المسند لنقطة ضعف يعكس حجم التهديد الناجم عنها. أما الشق الثاني فلأنه يعطي التقييم لنقطة الضعف وليس للثغرة نفسها وهذا مهم لان معظم الميزات التي سنقوم بدراستها هي مؤشرات أمنية وليست ثغرات.

الـ CWSS يستخدم 18 عامل ضمن ثلاث مجموعات قياس:

- المجموعة الأولى: النتائج الأساسية أو "Base Finding" تعكس خطورة نقطة الضعف ووجود آلية أمنية للدفاع.
- المجموعة الثانية: مجال الهجوم أو " Attack Surface" تعكس القدرة على استغلال نقطة الضعف. نقطة الضعف التي من السهل استغلالها كتضمين كود جافا سكريبت خارجي مثلاً ستتلقى تثقيل أكبر.

- المجموعة الثالثة: المجموعة البيئية "Environmental" تعكس بين عدة أشياء الأثر على مجرى العمل في حال تم استغلال نقطة الضعف هذه، كما تعكس احتمالية كشف واستغلال نقطة الضعف.

كل مجموعة يخصص لها وزن معين يتألف من الثقل الموزون لعواملها. التقييم الكلي المسند لنقطة ضعف يحسب بضرب ثقل مجموعة النتائج الأساسية (وهي قيمة بين 0 و100) بالمجموعتين الباقيتين (القيم بين 0 و1).

نقطة الضعف	الدرجة
تضمين ملفات خارجية غير موثوقة	67.50
إيقاف X-XSS-PROTECTION	28.33
تطبيق غير سليم لـ SSL	18.10
تضمين محتوى مختلط	13.42
استخدام برمجيات تخديم قديمة OUTDATED	8.71

جدول 8-1
القيم المسندة لنقاط الضعف

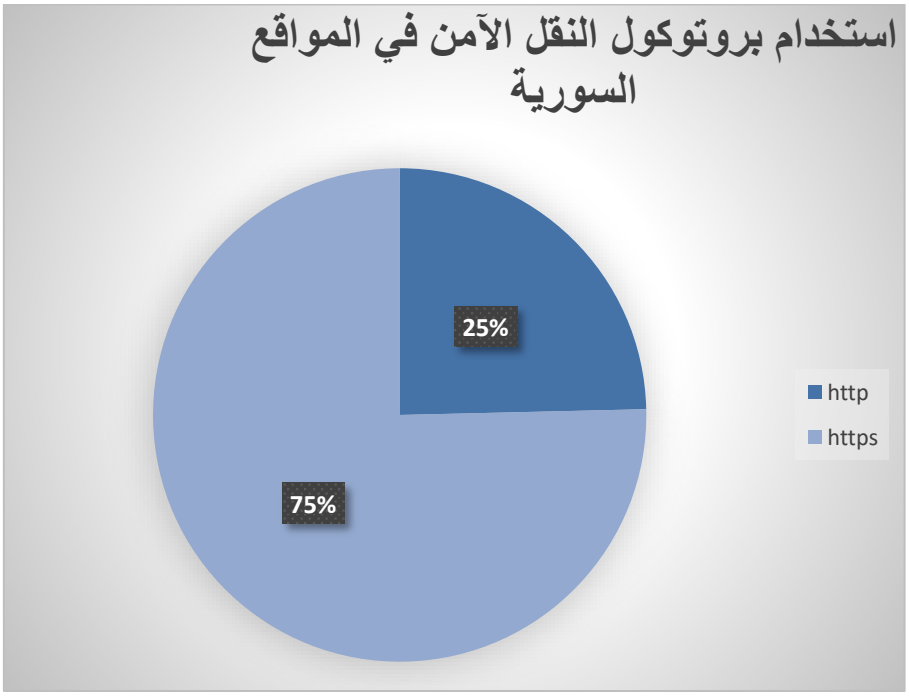
الجدول يوضح القيم المسندة لكل نقطة ضعف. واختصاراً سنشرح المنطق المستخدم للحصول على هذه القيم بمثال واحد:

كما هو واضح في الجدول فإن تضمين ملفات جافا سكريبت خارجية له الثقل الأكبر. فالأثر الكبير (تنفيذ شيفرات جافا سكريبت خبيثة ضمن سياق التطبيق) وسهولة الاستغلال (سهولة استغلال اسم نطاق قديم) هما العاملين الأساسيين وراء هذا التقييم العالي. بالإضافة لذلك لم يتم العثور على أية آلية دفاعية لمواجهة نقطة الضعف هذه (مثلاً استخدام CSP أو SRI). كنتيجة لهذا تحسب الدرجة الفرعية للمجموعة الأساسية 90. ومجال الهجوم سيأخذ القيمة 1. أما المجموعة البيئية فتأخذ القيمة 0.75، العامل الأساسي الذي ساهم بإعطاء هذه القيمة هو الأثر على مجرى العمل حيث هذا الأثر يختلف من حالة لأخرى فبينما قد يكون تنفيذ كود خبيث ذو تأثير هائل على المواقع الحساسة أمنياً مثل المواقع البنكية، فإن الأثر المتوقع على موقع تعليمي مثلاً لا يحمل بيانات حساسة فهو أقل بكثير. القيمة الكلية 67.50 تحسب بضرب القيم الثلاثة (90 * 1 * 0.75)

8.6 النتائج:

8.6.1 الميزات التي تساهم في تأمين الاتصال:

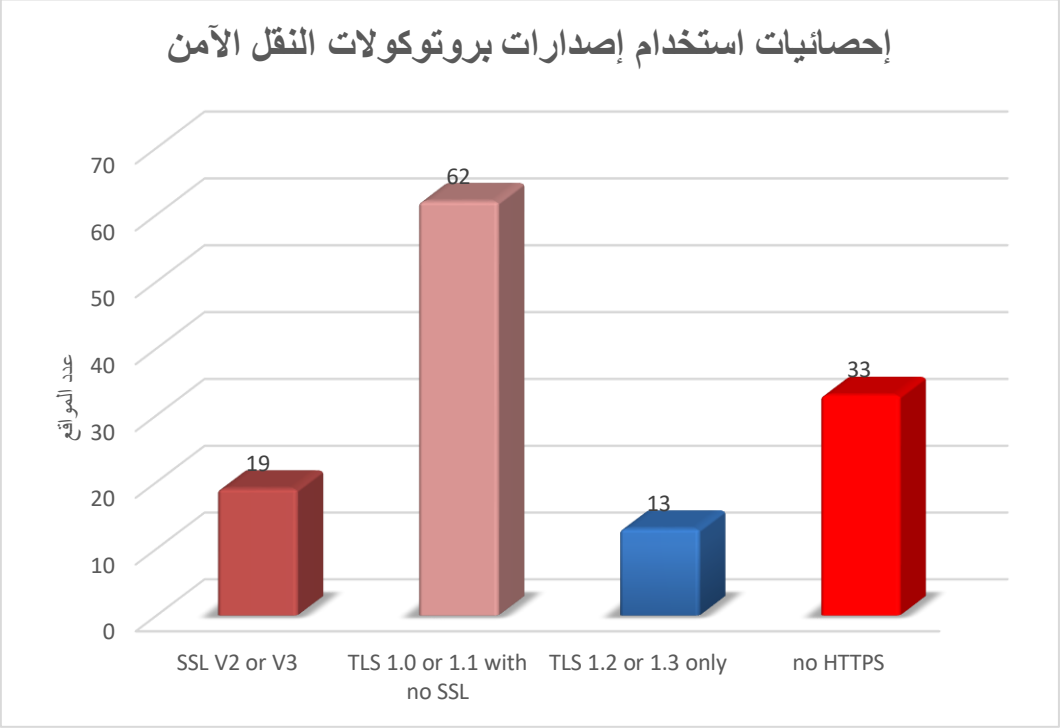
- من أصل 134 موقعا، وجدنا أن 101 موقع يستخدم **HTTPS**. أي حوالي ثلثي المواقع لكن للأسف 6 من هذه المواقع تستخدم شهادات غير صالحة لأسباب مختلفة. وفقط 14 موقع تقوم بإعادة توجيه الطلبات المرسلة عبر HTTP إلى HTTPS. وموقع وحيد يستخدم تقنية **HSTS** التي تحتم على الزبون عدم طلب التطبيق إلا عبر HTTPS. ولا يوجد أي موقع يستخدم تقنية **HPKP** لتلافي تزوير الشهادات الرقمية.



شكل 8-1
استخدام النقل الآمن في المواقع السورية

- تبين أن حوالي 75% من المواقع تستخدم بروتوكولات النقل الآمنة. قد تعتبر هذه النسبة جيدة نوعا ما، لكن إذا دققنا في إصدارات بروتوكول النقل المستخدم (النتائج الموضحة في الشكل الآتي) نجد أن 19 من هذه المواقع غير آمن لأنه ما يزال يقدم دعم بروتوكولات SSLV2 وSSLV3 واللتان تعتبرا غير آمنتين. وكذلك نجد في النتائج أن أغلب المواقع تدعم بروتوكولي TLS1.1 وTLS1.0 واللتان اتفقت معظم الشركات على أن تعتبرا غير آمنتين ابتداء من 2020\01\13. هذا يتركنا أمام 13 موقع فقط تقوم بتطبيق صحيح لـ TLS وموقع واحد فقط من هذه المواقع الثلاثة عشر يقوم

بإعادة التوجيه لاستخدام البروتوكول الآمن. بينما من جميع المواقع التي تستخدم بروتوكول النقل الآمن فقط 14% منها تقوم بإعادة توجيه الطلبات المرسلة عبر http إلى https



شكل 8-2
إحصائيات استخدام إصدارات بروتوكولات النقل الآمن

- أما من جهة استخدام خاصية **secure** لتأمين ملفات تعريف الارتباط فوجدنا موقع وحيد قد قام بتفعيل هذه الميزة. ووجود العديد من المواقع التي تستخدم هذه الخاصية لكن من دون تفعيلها. أي أن ('secure': False) وهذا غالبا ما يؤشر إلى أنها تولد اوتوماتيكيا من قبل بيئة التطوير المستخدمة وليس من قبل المطور.

8.6.2 الميزات التي تواجه هجوم البرمجة العابرة للموقع:

- بالنسبة لاستخدام خاصيتي **httpOnly** و **samesite** فالوضع مشابه لاستخدام خاصية **secure**. هنالك موقعين استخدما ('samesite':'Lax') بينما هنالك 77 موقع استخدموا ('samesite':None). كذلك يوجد 80 موقع استخدموا ('httponly': None') ولا يوجد أي موقع قام بتفعيلها بشكل صحيح.
- أما بالنسبة لـ **سياسة أمن المحتوى CSP** فوجدنا موقعاً وحيداً قام بتفعيل الترويسة التجريبية لـCSP والتي لم تعد مدعومة (X-Content-Security-Policy) لأنها كانت لأغراض تجريبية.

وموقع وحيد آخر قام بتفعيل سياسة CSP التالية:

```
Content-Security-Policy:
 script-src 'self' www.google.com cdnjs.cloudflare.com
www.gstatic.com 'unsafe-inline' 'unsafe-eval';
```

تقوم هذه السياسة بتقييد مصادر الجافا سكريبت لكنها تسمح باستخدام تابع eval() الخطير والذي يقوم بتنفيذ كود جافا سكريبت المضمن في سلاسل نصية وكذلك تسمح هذه السياسة بتنفيذ الشيفرات البرمجية السطرية الواردة ضمن كود HTML وهذا الأمر يعتبر ممارسة غير مرغوبة أمنيا.

- **ترويسة X-Content-Type-Options(XCTO)** قامت 8 مواقع فقط بتفعيل هذه الترويسة أي حوالي 6% فقط من المواقع.
- **Subresource Integrity** (SRI): لم يقم أي موقع باستخدام هذه التقنية.
- **خاصية sandbox:** كذلك لم يستخدم أي موقع هذه الخاصية في الأطر المستخدمة ضمن الموقع. علما أنه فقط 14 موقعاً يقوم باستخدام أطر ذات مصادر خارجية. ومعظم هذه الأطر تستخدم لتضمين ملفات فيديو من موقع YouTube أو تضمين خارطة الموقع الجغرافي من Google Maps.

8.6.3 الميزات التي تمكن من السماح بالتأطير بشكل آمن:

- **X-Frame-Options** (XFO): قام 19 موقع بتفعيل هذه الترويسة.
- **ميزة frame_ancestors:** لم يتم استخدام CSP إلا من قبل موقع وحيد والسياسة الموضوعة من قبله لم تستخدم هذه الخاصية.

8.6.4 تضمين ملفات جافا سكريبت خارجية غير موثوقة المصدر:

يعتبر ملف الجافا سكريبت خارجياً إذا كان غير مخزنٍ على مخدمٍ خاصٍ بالتطبيق وخاضع لسيطرته. وتم تضمينه من دون استخدام قيود ك SRI أو رموز التقطيع.
هنالك 34 موقعا يقوم باستخدام الملفات الخارجية بشكل غير موثوق.

8.6.5 تضمين محتوى مختلط mixed-content:

وذلك يعني استخدام التطبيق لملفات خارجية وتحميلها عبر HTTP وليس HTTPS علما أن التطبيق مخدم عبر HTTPS.

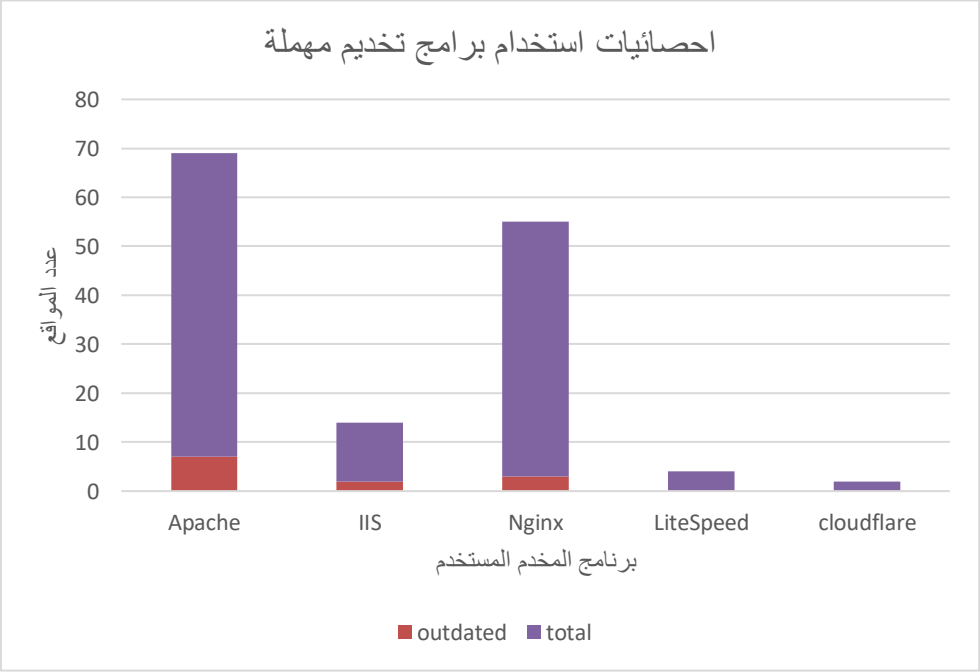
من أصل 101 موقع يستخدم HTTPS هنالك 32 موقع يستخدم محتوى مختلط.

8.6.6 إيقاف خاصية X-XSS-Protection:

وجدنا 4 مواقع تصرح بإيقاف حماية **XSS** المقدمة من قبل المستعرضات. كان يعتبر سلوكاً إيجابياً فيما مضى بسبب ضعف الحماية المضمنة في المستعرضات واحتمال فتح ثغرات جديدة. لكن حاليا يعتبر هذا السلوك سلبي لأنه يترك ثغرات يمكن استغلالها وكان يمكن للمستعرض تفاديها.

8.6.7 استخدام برامج تخديم مهملة outdated server software:

المخطط البياني الآتي يوضح النتائج التي حصلنا عليها نتيجة سبر ترويسة **server** المرسلة مع رد المخدم. علما أن هذه النتائج ليست دقيقة كون بعض المخدمات لا تقوم بالتصريح عن الإصدار الدقيق للبرنامج المستخدم، وهذا ما ينصح به لأن تقديم هذه المعلومات يتيح للمهاجمين استغلال نقاط ضعف معروفة في هذه البرامج.

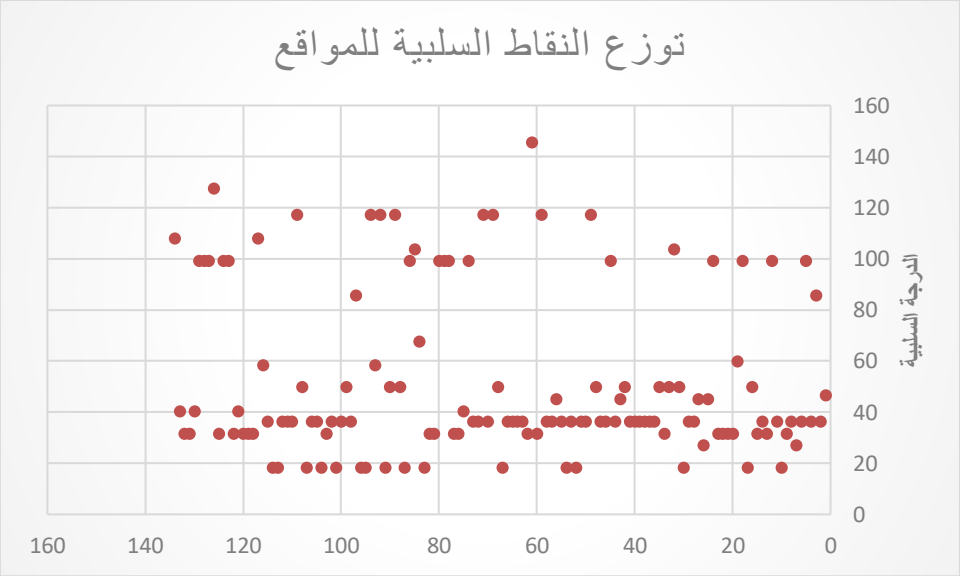


شكل 8-3
احصائيات استخدام برامج تخديم مهملة

8.7 الدرجات الأمنية للمواقع:

8.7.1 الدرجات السلبية:

يبين الشكل الآتي توزع الدرجات السلبية للمواقع.



شكل 8-4 توزع الدرجات السلبية للمواقع

نلاحظ أن النسبة الأكبر من المواقع حصلت على درجة سلبية منخفضة نسبياً (أقل من 40 درجة). لكن جميعها فعليا حصلت على درجات سلبية ما يعني وجود ثغرات ونقاط ضعف في جميع المواقع.

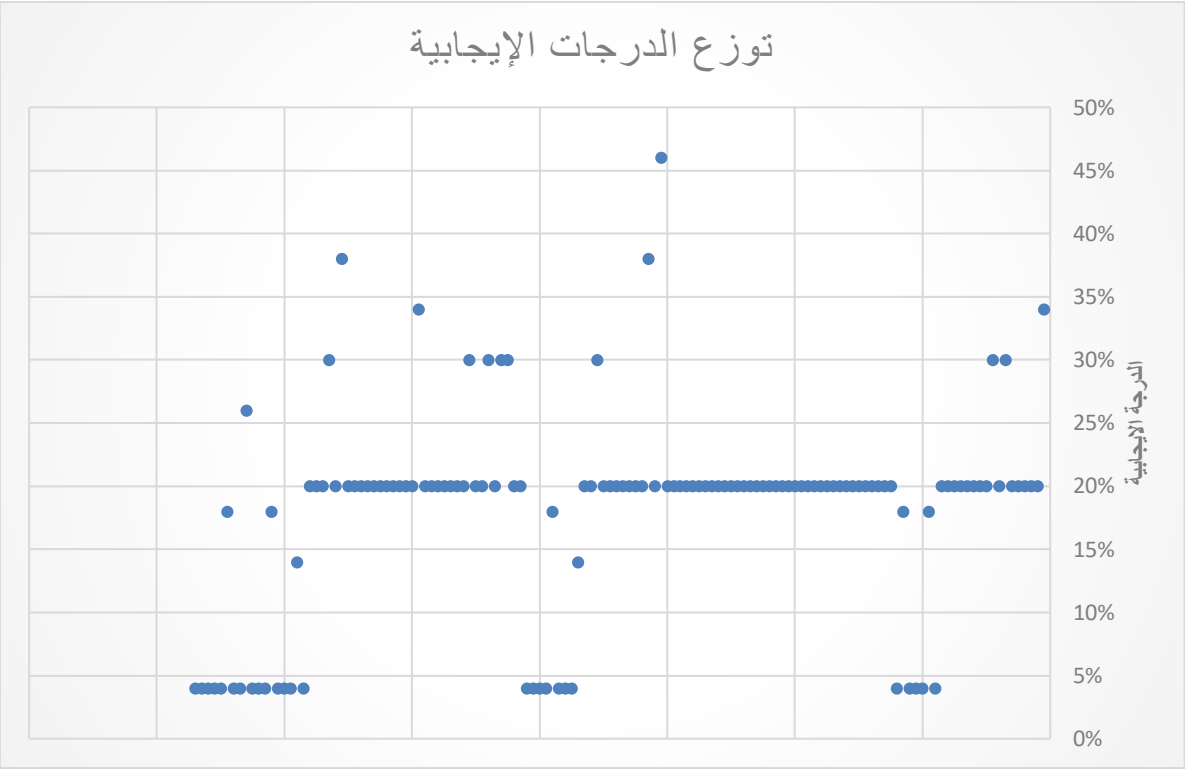
ملاحظة:

بالنسبة المواقع التي لا تستخدم **HTTPS** وبما أن تفعيل بروتوكول النقل الآمن أصبح أمرا بديهيا ولأن معظم نقاط الضعف المدروسة تعتمد على استخدامه، قمنا بإضافة القيمة:

$$(18.10 + 13.42 = 31.52)$$

والتي تعبر عن مجموع الدرجات المسندة لنقاط الضعف المدروسة المتعلقة باستخدام بروتوكول **HTTPS**، وهي: التطبيق غير سليم لـ **SSL** واستخدام محتوى مختلط.

8.7.2 الدرجات الإيجابية:



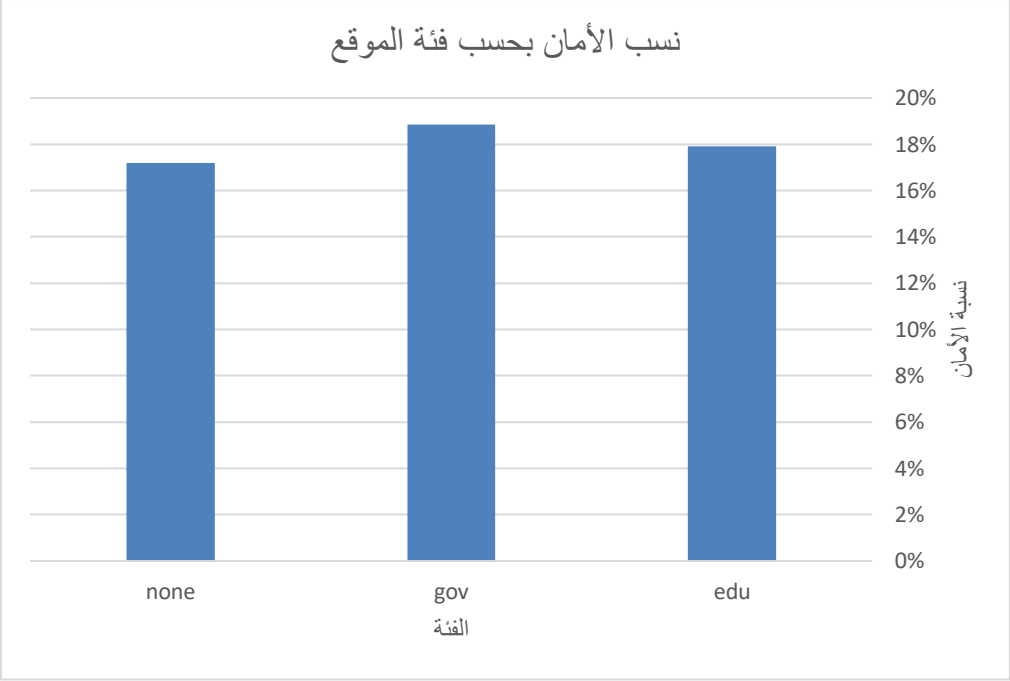
شكل 8-5 توزع الدرجات الايجابية للمواقع

نلاحظ أن معظم المواقع لا تتجاوز درجتها أمانها الـ 25% وجميع المواقع لا تتجاوز الـ 50% ما يشير إلى قلة الوعي الأمني.

8.7.3 متفرقات:

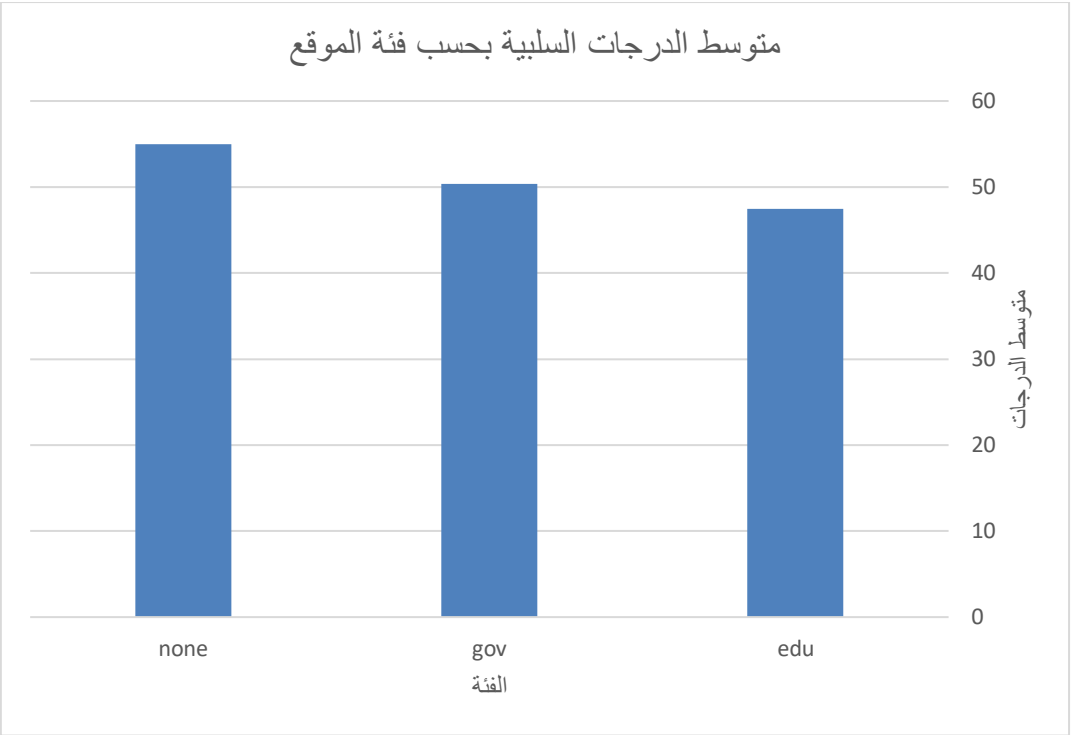
8.7.3.1 الدرجات بحسب فئة الموقع:

نلاحظ من النتائج التقارب والضعف العام لأمان المواقع بغض النظر عن الفئة التي ينتمي إليها الموقع:



شكل 8-6 نسب أمان المواقع بحسب الفئة

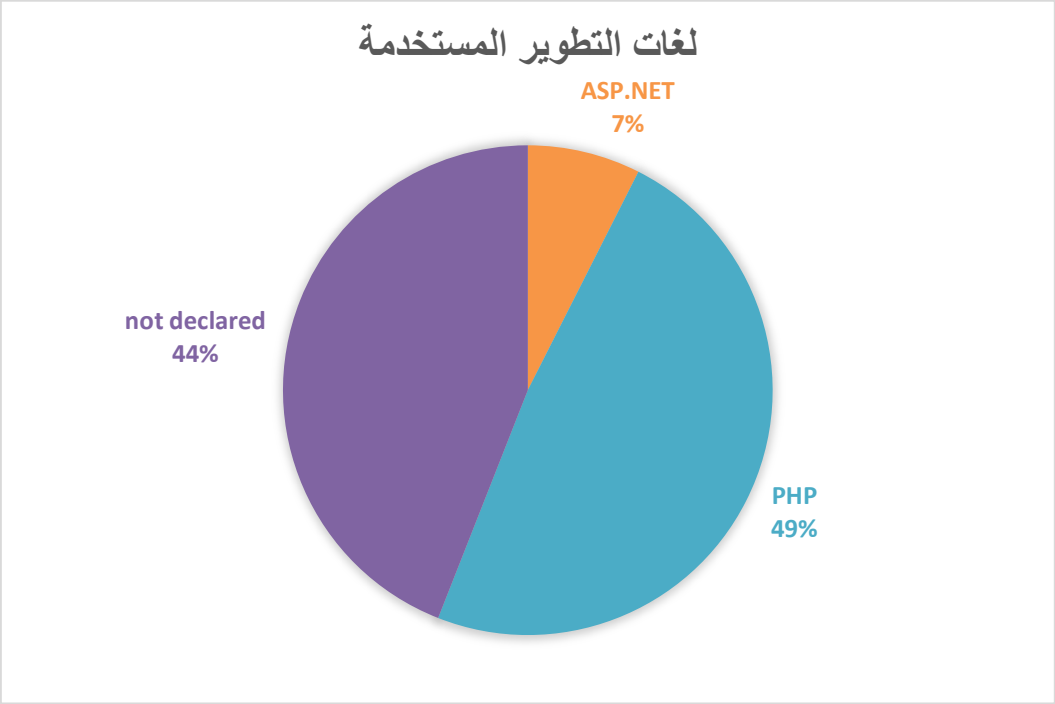
تتقارب معدلات نسب الأمان للمواقع بين الفئات المختلفة ولكنها جميعاً لا تتجاوز 20% مع تفوّق بسيط للمواقع الحكومية والتعليمية على المواقع العادية. أي أن الوعي الأمني ضعيف بغض النظر عن طبيعة الموقع. كما نلاحظ أن متوسط الدرجات السلبية للمواقع التعليمية والحكومية أقل من المواقع العادية.



شكل 8-7 متوسط الدرجات السلبية وفقاً لفئة الموقع

8.7.3.2 الدرجات وفقاً للغة التطوير المعلنة:

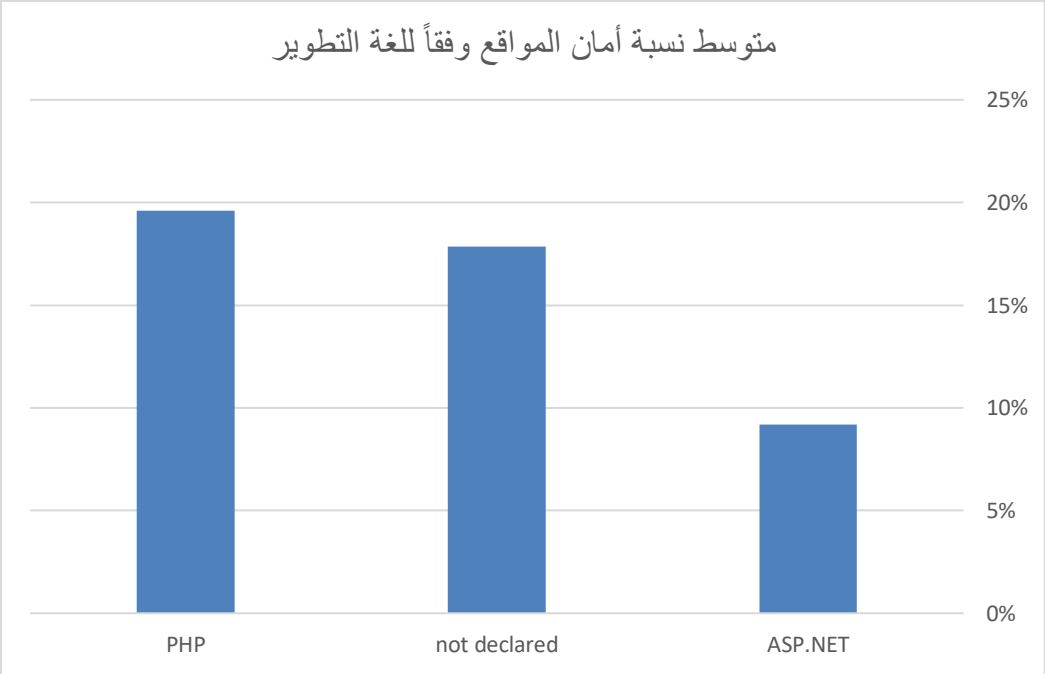
لاحظنا خلال دراستنا أن لغة التطوير الأكثر استخداماً هي **PHP** بنسبة 49%:



شكل 8-8 لغات التطوير المستخدمة

شكل 8-9 توزع درجات الضعف بحسب لغة التطوير المستخدمة

70%من المواقع المعتمدة على **ASP.NET** مجموع نقاط ضعفه أقل من 50 بينما تقريبا 69% من المواقع المعتمدة على **PHP** مجموع نقاط ضعفها أقل من 50 أي أن النسبة متقاربة جداً.



شكل 8-10 متوسط نسبة أمان المواقع وفقاً للغة التطوير

مجموع نقاط الضعف متقارب إلا أن معدل أمان المواقع المطورة باستخدام **PHP** ضعف نسبة أمان المواقع المطورة باستخدام **ASP.NET**.

8.8 التحديات:

8.8.1 دقة الدراسة الساكنة للخصائص الأمنية

نظرًا للإعتبارات القانونية والأخلاقية، اقتصر تحليلنا لمواطن الضعف في المواقع الإلكترونية على التحليل الساكن غير الفاعل، مع وجود استثناءات قليلة. وبالتالي، فإن النتائج الموضحة سابقاً هي فقط تقدير نسبي لحالة أمان المواقع السورية. على الرغم من أنه لا يمكن تقييم سوى جزء ضئيل من حالة أمان موقع الويب، إلا أننا ما زلنا قادرين على التمييز بين المواقع الضعيفة والمواقع القوية أمنياً. في الوقت نفسه، نحن على دراية بعدم دقة تحليلنا وألقينا الضوء على محدودية تقييم الأمان الخارجي من دون تفاعل مع مسؤولي المواقع. قد يكون فعالا التحقق مما إذا كان مسؤولو المواقع الإلكترونية على استعداد للموافقة على تقييم أمني أكثر فاعلية، في مقابل الحصول على النتائج مجانًا.

8.8.2 نظام التقييم

لتقييم الحالة العامة للأمان لموقع الويب، قمنا بتطوير نظام تقييم على أساس **CWSS**، كما وضحنا سابقاً. ومع ذلك، يخضع نظام التقييم هذا لنوعين من القيود:

- أولاً، ينشأ إجمالي الدرجات الإيجابية المعينة لموقع الويب عن نتيجة معينة بشكل فردي من 12 ميزة أمان، بينما يتم الحصول على مجموع النقاط السلبية من نتيجة تُعزى إلى أربع نقاط ضعف. نتيجة لذلك، تكون النتيجة الإيجابية لموقع الويب على نطاق مختلف عن النتيجة السلبية. هذا يمنعنا من أن نكون قادرين على مقارنة النتيجة الإيجابية بالنتيجة السلبية.

- نظرًا لأن إجمالي النقاط المعيّنة في موقع ويب ينبع من مجموعة محدودة من العوامل، فقد لا تعكس النتيجة الإجمالية دائمًا الحالة الأمنية الفعلية لموقع الويب.

ومع ذلك، نظرًا لأننا نقيم الجوانب المتنوعة المرتبطة بشكل كبير بأمان موقع الويب، فإننا نعتقد أن نظام التقييم لدينا يوفر تقديرًا جيدًا للحالة العامة للأمان لموقع الويب. كما قد يختلف تأثير استغلال بعض الثغرات الأمنية حسب نوع موقع الويب. من أجل حساب هذه الاختلافات، تم حساب كل مقياس لموقع

ويب عام. وبالتالي، فإن المقاييس التي تم تعيينها ليست خاصة بموقع الويب ودرجة ميزة واحدة مرتبطة بالدرجات الأخرى. يتيح لنا ذلك تعيين درجة قابلة للمقارنة مما يعطي تقديرًا لحالة الأمان لكل موقع ويب تم اختباره.

8.9 الاستنتاجات والتوصيات:

لاحظنا من النتائج وجود عدد لا بأس به من المواقع السورية يحوي نقاط ضعف وثغرات. وبعضها يحوي ثغرات خطيرة. كما لوحظ غياب كبير لاستخدام تقنيات التخفيف من المخاطر الأمنية ومعالجتها.

في عصرنا الحالي الذي أصبح فيه كل العالم يعتمد على تطبيقات الويب في كافة مجالات الحياة، لم يعد ممكنا تجاهل أهمية أمن التطبيقات لأنه العقبة الأساسية التي تقف دون استخدام هذه التطبيقات في المجالات الحياتية في بلدنا. فمثلاً لا يمكن تطبيق الحوكمة الالكترونية بشكل جدي في ظل غياب الوعي الأمني. هذا عدا عن التعاملات البنكية والدفع الالكتروني. وحتى في أبسط الاشكال مثل التراسل الرسمي عبر المواقع الالكترونية.

تم إحداث الهيئة الوطنية لخدمات الشبكة [14] بموجب القانون رقم /4/ تاريخ 2009/2/25، بهدف تنظيم وتنسيق وتسهيل العمل على الشبكة المعلوماتية. وإحدى مهامها تقديم الخدمات الأمنية من خلال مركز أمن المعلومات. لكنها تركز في عملها على أمن المخدم وليس على أمن المستخدم. حتى أنه وللأسف الموقع التابع للهيئة لا يستخدم طبقة النقل الآمنة **https** حتى لصفحات تسجيل المستخدم وتسجيل الدخول. والذي يعتبر المعيار الأكثر ثقلاً في تقييم أمان موقع ما.

على ضوء ما سبق نوصي بالسعي لزيادة الوعي الأمني من خلال:

- توجيه المزيد من الدراسات والأبحاث للتعمق في مجال أمن التطبيقات محليا.
- تفعيل دور الهيئة الوطنية لإدارة الشبكة وتطوير وتدريب كوادرها.

ملحق A

خاصية SameSite لملفات تعريف الارتباط

ملفات تعريف الارتباط هي إحدى الطرق المتاحة لإضافة حالة ثابتة إلى مواقع الويب. على مر السنين نمت قدراتها وتطورت، ولكن بقي النظام الأساسي مع بعض الإشكاليات القديمة. لمعالجة هذا، تقوم المتصفحات (بما في ذلك **Chrome** و**Firefox** و**Edge**) بتغيير سلوكها لفرض المزيد من الإعدادات الافتراضية للحفاظ على الخصوصية.

كل ملف تعريف ارتباط هو زوج (مفتاح = قيمة) مع عدد من السمات التي تتحكم في متى وأين يتم استخدام ملف تعريف الارتباط هذا. تستخدم هذه السمات لتعيين أشياء مثل تواريخ انتهاء الصلاحية أو الإشارة إلى أنه يجب إرسال ملف تعريف الارتباط فقط عبر **HTTPS**. تقوم الخوادم بتعيين ملفات تعريف الارتباط عن طريق إرسال ترويسة **Set-Cookie** بشكل مناسب في استجابتها. للحصول على جميع التفاصيل يمكن مراجعة [139].

لنفترض أن لديك مدونة حيث تريد الإعلان عن عرض ترويجي "ما جديدنا" للمستخدمين. يمكن للمستخدمين استبعاد العرض الترويجي ثم لن يروه مرة أخرى لفترة من الوقت. يمكنك تخزين هذا التفضيل في ملف تعريف ارتباط، وتعيينه بحيث تنتهي صلاحيته خلال شهر (2،600،000 ثانية)، وإرساله فقط عبر **HTTPS**. ستبدو هذه الترويسة كما يلي:

```
Set-Cookie: promo_shown=1; Max-Age=2600000; Secure
```

عندما يقوم القارئ بعرض صفحة تلبي هذه المتطلبات، أي أنها على اتصال آمن وكان عمر ملف تعريف الارتباط أقل من شهر، عندها سيرسل المتصفح هذا العنوان في طلبه:

```
Cookie: promo_shown=1
```

يمكنك أيضًا إضافة وقراءة ملفات تعريف الارتباط المتاحة لذلك الموقع في **JavaScript** باستخدام تعليمة **document.cookie**. يؤدي تنفيذ تعليمة **document.cookie** إلى إنشاء ملف تعريف ارتباط أو إعادة كتابته باستخدام هذا المفتاح. على سبيل المثال، يمكنك تجربة ما يلي في وحدة تحكم **JavaScript** بالمتصفح:

```
> document.cookie = "promo_shown=1; Max-Age=2600000; Secure"
< "promo_shown=1; Max-Age=2600000; Secure"
```

ستؤدي قراءة **document.cookie** إلى إخراج جميع ملفات تعريف الارتباط التي يمكن الوصول إليها في السياق الحالي ، مع فصل كل ملف تعريف ارتباط بفاصلة منقوطة:

```
> document.cookie;
< "promo_shown=1; color_theme=peachpuff; sidebar_loc=left"
```

إذا جربت ذلك على مجموعة مختارة من المواقع الشهيرة، فستلاحظ أن معظمها يقوم بتعيين أكثر بكثير من ثلاثة ملفات تعريف ارتباط فقط. في معظم الحالات، يتم إرسال ملفات تعريف الارتباط هذه مع كل طلب إلى النطاق، والذي له عدد من الآثار. غالبًا ما يكون عرض الحزمة لرفع البيانات أكثر تقييدًا من التنزيل للمستخدمين، بحيث يؤدي الحمل الزائد على جميع الطلبات الصادرة إلى إضافة تأخير في زمن البايت الأول. يجب أن تكون متحفظًا في عدد وحجم ملفات تعريف الارتباط التي تحددها. استفد من ميزة **Max-Age** للمساعدة في ضمان عدم بقاء ملفات تعريف الارتباط لفترة أطول من اللازم.

ما هي ملفات تعريف ارتباط الطرف الأول first-party والطرف الثالثthird-party ؟

إذا عدت إلى المواقع التي زرتها سابقاً، فمن المحتمل أنك لاحظت وجود ملفات تعريف ارتباط موجودة لمجموعة متنوعة من النطاقات، وليس فقط النطاق الذي كنت تزوره حاليًا. يشار إلى ملفات تعريف الارتباط المطابقة لنطاق الموقع الحالي، أي ما يتم عرضه في شريط عناوين المتصفح، باسم ملفات تعريف ارتباط الطرف الأول. وبالمثل، يشار إلى ملفات تعريف الارتباط من المجالات الأخرى غير الموقع الحالي باسم ملفات تعريف ارتباط الطرف الثالث. هذه ليست تسمية مطلقة ولكنها مرتبطة بسياق المستخدم، نفس ملف تعريف الارتباط يمكن أن يكون طرف أول أو طرف ثالث اعتمادا على موقع المستخدم في ذلك الوقت.

بمتابعة المثال الوارد أعلاه، دعنا نقول إن إحدى مشاركات مدونتك تحتوي على صورة لقط مدهش بشكل خاص موجودة في العنوان التالي (**blog/img/amazing-cat.png/**). نظرًا لأنها صورة مذهلة، يستخدمها شخص

آخر مباشرة في موقعه. إذا كان زائر ما قد زار مدونتك سابقاً وكان لديه ملف تعريف الارتباط المسمى **promo_shown**، فعندما يشاهد ملف **amazing-cat.png** على موقع الشخص الآخر، سيتم إرسال ملف تعريف الارتباط في هذا الطلب للصورة. وهذا لا فائدة له نظرًا لأن **promo_shown** لا يستخدم لأي شيء على موقع هذا الشخص الآخر، إنه يضيف فقط حمل زائد إلى الطلب.

إذا كان هذا غير مقصود، فلماذا تريد القيام بذلك؟ إنها الآلية التي تسمح للمواقع بالحفاظ على حالتها عند استخدامها في سياق جهة خارجية. على سبيل المثال، إذا قمت بتضمين مقطع فيديو **YouTube** على موقعك، فسيشاهد الزوار خيار "المشاهدة لاحقًا" في المشغل. إذا كان قد تم تسجيل دخول الزائر مسبقاً إلى **YouTube**، فسيتم توفير تلك الجلسة في المشغل المضمن بواسطة ملف تعريف ارتباط طرف ثالث مما يعني أن زر "المشاهدة لاحقًا" سيؤدي إلى حفظ الفيديو مباشرة بدلاً من مطالبته بتسجيل الدخول أو الاضطرار إلى الانتقال بعيدًا عن صفحتك والعودة إلى **YouTube**.

تعتمد هجمات تزوير الطلبات العابر للموقع (**CSRF**) على حقيقة أن ملفات تعريف الارتباط مرفقة بأي طلب من أصل معين، بغض النظر عمن يرسل الطلب. على سبيل المثال، إذا قمت بزيارة **evil.example**، فيمكنه إرسال الطلبات إلى **your-blog.example**، وسيقوم المتصفح الخاص بك بإرفاق ملفات تعريف الارتباط المرتبطة. إذا لم تكن مدونتك حريصة على كيفية التحقق من صحة تلك الطلبات، فيمكن أن يقوم **evil.example** بتفعيل إجراءات مثل حذف المنشورات أو إضافة محتواها.

أصبح المستخدمون أكثر وعياً بكيفية استخدام ملفات تعريف الارتباط لتتبع نشاطهم عبر مواقع متعددة. لكن حتى الآن لم تكن هناك طريقة لتوضيح نيتك صراحة من ملف تعريف الارتباط. يجب إرسال ملف تعريف الارتباط **promo_shown** الخاص بك فقط في سياق الطرف الأول، في حين أن ملف تعريف الارتباط الخاص بالجلسة لعنصر ما مراد تضمينه في مواقع أخرى فوجوده مقصود لتوفير حالة تسجيل الدخول ضمن سياق طرف ثالث.

التصريح عن القصد من ملف تعريف الارتباط باستخدام خاصية SameSite:

يتيح لك إدخال سمة **SameSite** (المعرفة في **RFC6265bis** [64]) أن تصرح للمستعرض ما إذا كان ملف تعريف الارتباط الخاص بك يجب أن يقتصر على سياق الطرف الأول أي نفس الموقع. من المفيد أن نفهم

بالضبط معنى "الموقع" هنا. الموقع هو مزيج من لاحقة النطاق واسم النطاق قبله مباشرة. على سبيل المثال، يعد النطاق **www.web.dev** ضمن موقع **web.dev**.

مثال: إذا كان المستخدم موجودًا ضمن **www.web.dev** ويطلب صورة من **static.web.dev**، فهذا طلب ضمن موقع واحد.

قائمة اللواحق العامة [140] "**The public suffix list**" تحدد ذلك، فهي لا تحتوي فقط نطاقات المستوى الأعلى مثل **.com** ولكن تشمل أيضًا خدمات مثل **github.io**. وهذا ما يمكّن من اعتبار **your-project.github.io** و **my-project.github.io** موقعين منفصلين. أي إذا كان المستخدم على **your-project.github.io** ويطلب صورة من **my-project.github.io**، فهذا طلب عبر المواقع.

خاصية **SameSite** لملف تعريف الارتباط تقدم ثلاث طرق مختلفة للتحكم في هذا السلوك. يمكنك اختيار عدم تحديد الخاصية، أو يمكنك استخدام **Strict** أو **Lax** لتقييد ملف تعريف الارتباط على طلبات الموقع نفسه.

إذا استخدمت القيمة **Strict**، فلن يتم إرسال ملف تعريف الارتباط الخاص بك إلا في سياق الطرف الأول. من ناحية المستخدم، لن يتم إرسال ملف تعريف الارتباط إلا إذا كان موقع ملف تعريف الارتباط يطابق الموقع المعروض حاليًا في شريط **URL** الخاص بالمتصفح. لذلك، إذا تم تعريف ملف تعريف الارتباط **promo_shown** على النحو التالي:

```
Set-Cookie: promo_shown=1; SameSite=Strict
```

عندما يكون المستخدم ضمن موقعك، سيتم إرسال ملف تعريف الارتباط مع الطلب كما هو متوقع. ومع ذلك، عند اتباع رابط إلى موقعك، مثلاً من موقع آخر أو عبر بريد إلكتروني من صديق، لن يتم إرسال ملف تعريف الارتباط بناءً على هذا الطلب الأولي. يعد هذا أمرًا جيدًا عندما يكون لديك ملفات تعريف ارتباط متعلقة بمهمة مثل تغيير كلمة المرور أو إجراء عملية شراء، ولكنها تكون مقيدة للغاية بالنسبة إلى **promo_shown**. إذا اتبع القارئ رابط ما إلى الموقع، فإنه يريد أن يتم إرسال ملف تعريف الارتباط حتى يمكن تطبيق تفضيلاته.

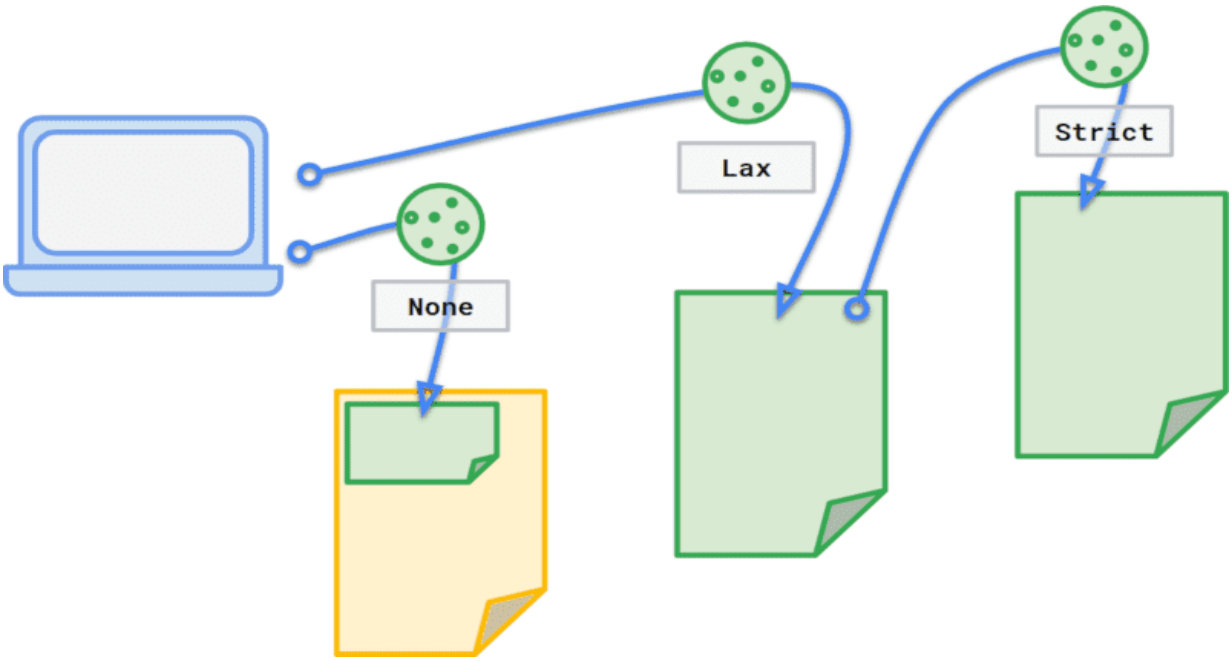
هنا يأتي دور **SameSite = Lax** من خلال السماح بإرسال ملف تعريف الارتباط مع هذه التنقلات ذات المستوى الأعلى. دعنا نعيد النظر في مثال منشور القط من الأعلى حيث يشير موقع آخر إلى المحتوى الخاص بك. فهم يستخدمون صورتك الخاصة بالقط مباشرةً ويقدمون رابطًا لمقالك الأصلي.

```
<p>Look at this amazing cat!</p>
<img src="https://blog.example/blog/img/amazing-cat.png" />
<p>Read the <a
href="https://blog.example/blog/cat.html">article</a>.</p>
```

وملف تعريف الارتباط تم تعريفه بالشكل التالي:

```
Set-Cookie: promo_shown=1; SameSite=Lax
```

عندما يكون القارئ في مدونة الشخص الآخر، لن يتم إرسال ملف تعريف الارتباط عندما يطلب المتصفح **amazing-cat.png**. ومع ذلك، عندما يتبع القارئ الرابط إلى **cat.html** على مدونتك، سيشمل هذا الطلب ملف تعريف الارتباط. هذا يجعل **Lax** اختيارًا جيدًا لملفات تعريف الارتباط التي تؤثر على عرض الموقع مع كون **Strict** مفيدًا لملفات تعريف الارتباط المتعلقة بالعمليات التي يقوم بها المستخدم.



أخيرًا، هناك خيار عدم تحديد القيمة، الذي كان من قبل وسيلة للقول ضمنيًا أنك تريد إرسال ملف تعريف الارتباط في جميع السياقات. في المسودة الأخيرة لـ **RFC6265bis**، يتم توضيح ذلك بإدخال قيمة جديدة من **SameSite= None** . هذا يعني أنه يمكنك التصريح أنك تريد عن قصد إرسال ملف تعريف الارتباط في سياق جهة خارجية.

التغييرات في السلوك الافتراضي للمستعرضات للتعامل مع ملفات تعريف الارتباط:

سيقوم **Chrome** و **Firefox** و **Edge** وغيرهم بتغيير سلوكهم الافتراضي بما يتماشى مع مقترح **IETF**، **Incrementally Better Cookies** [70] بحيث:

سيتم التعامل مع ملفات تعريف الارتباط التي لا تحتوي على سمة **SameSite** على أنها **SameSite = Lax**، مما يعني أن السلوك الافتراضي سيكون لتقييد ملفات تعريف الارتباط على سياقات الطرف الأول فقط.

يجب أن تحدد ملفات تعريف الارتباط للاستخدام العابر للموقع `SameSite=None; Secure` لتمكين التضمين في سياق طرف ثالث.

أصبح هذا هو السلوك الافتراضي في **Chrome 80**، والذي كان في مرحلة تجريبية اعتبارًا من 19 كانون الثاني 2019، وتم إصداره في شباط 2020. إذا كنت تقدم حاليًا ملفات تعريف الارتباط المخصصة للاستخدام عبر المواقع، فستحتاج إلى إجراء التغييرات.

هناك عدد من حالات الاستخدام الشائعة والأنماط التي تحتاج إلى إرسال ملفات تعريف الارتباط في سياق طرف ثالث. إذا قمت بالاعتماد على إحدى حالات الاستخدام هذه، فتأكد من قيامك أنت أو المزود بتحديث ملفات تعريف الارتباط الخاصة بهم لضمان استمرار الخدمة في العمل بشكل صحيح.

المصدر : **https://web.dev/samesite-cookies-explained/**

ملحق B

سياسة أمان المحتوىCSP

تعد سياسة أمان المحتوى (**CSP**) طبقة إضافية من الأمان تساعد على اكتشاف أنواع معينة من الهجمات وتخفيفها، بما في ذلك هجمات البرمجة العابرة للموقع (**XSS**) وهجمات حقن البيانات. يتم استخدام هذه الهجمات لكل شيء بدءًا من سرقة البيانات إلى تشويه الموقع وحتى نشر البرامج الضارة.

تم تصميم **CSP** ليكون متوافقًا تمامًا مع الإصدارات السابقة (باستثناء الإصدار 2 من **CSP** حيث توجد بعض التناقضات المذكورة صراحة في التوافق مع الإصدارات السابقة) . المتصفحات التي لا تدعمها تعمل مع المخدمات التي تقوم بتطبيقها، والعكس بالعكس: المتصفحات التي لا تدعم **CSP** تتجاهلها ببساطة، وتعمل كالمعتاد، بحيث تعود لاستخدام سياسة الأصل الموحد لمحتوى الويب. إذا كان الموقع لا يقدم ترويسة **CSP**، فإن المتصفحات التي تدعمها أيضًا تعود لاستخدام سياسة الأصل الموحد.

لتمكين **CSP**، تحتاج إلى إعداد مخدم الويب الخاص بك لإعادة ترويسة **HTTP Content-Security-Policy** ضمن الردود (في بعض الأحيان تستخدم ترويسة **X-Content-Security-Policy**، لكنها إصدار قديم يعود إلى المرحلة التجريبية).

كخيار بديل لاستخدام الترويسة، يمكن استخدام عنصر **<meta>** للتصريح عن السياسة، على سبيل المثال:

```
<meta http-equiv="Content-Security-Policy" content="default-
src 'self'; img-src https://*; child-src 'none';">
```

معالجة هجوم البرمجة العابرة لموقع:

الهدف الأساسي لـ **CSP** هو تخفيف هجمات **XSS** والإبلاغ عنها. تستغل هجمات **XSS** ثقة المستعرض بالمحتوى المستلم من المخدم. يتم تنفيذ البرامج النصية الضارة من قبل متصفح الضحية لأن المتصفح يثق في مصدر المحتوى، حتى عندما لا يأتي من حيث يبدو أنه قادم.

تتيح **CSP** لمسؤولي التطبيق تقليل أو إزالة الثغرات التي يمكن أن يحدث بها **XSS** عن طريق تحديد النطاقات التي يعتبرها المستعرض مصادر صالحة للبرامج النصية القابلة للتنفيذ. عندئذٍ، سيقوم متصفح متوافق مع **CSP** بتنفيذ البرامج النصية المستلمة من تلك النطاقات المسموح بها، مع تجاهل كل البرامج النصية الأخرى (بما في ذلك البرامج النصية السطريّة المضمنة وسمات **HTML** الخاصة بمعالجة الأحداث).

كنوع نهائي من الحماية، يمكن للمواقع التي لا تسمح مطلقًا بتنفيذ البرامج النصية أن تختار عدم السماح بتنفيذ الجافا سكريبت نهائيا.

تخفيف هجمات التنصت:

بالإضافة إلى تقييد النطاقات التي يمكن تحميل المحتوى منها، يمكن للمخدم تحديد البروتوكولات المسموح باستخدامها؛ على سبيل المثال (والأفضل من وجهة نظر الأمان)، يمكن أن يحدد المخدم أنه يجب تحميل جميع المحتويات باستخدام **HTTPS**. لا تتضمن استراتيجية أمان نقل البيانات الكاملة فقط تطبيق **HTTPS** لنقل البيانات، ولكن أيضًا تفعيل خاصية **secure** لجميع ملفات تعريف الارتباط وإعادة التوجيه التلقائية من صفحات **HTTP** إلى **HTTPS**. يمكن للمواقع أيضًا استخدام ترويسة **Strict-Transport-Security** للتأكيد من أن المتصفحات تتصل بهم فقط عبر قناة مشفرة.

استخدام وكتابة CSP:

لإعداد سياسة أمان المحتوى يتوجب إضافة ترويسة **Content-Security-Policy** إلى صفحة الويب ومنحها قيمًا للتحكم في الموارد التي يُسمح للزبون بتحميلها لتلك الصفحة. تساعد سياسة أمان المحتوى المصممة بشكل صحيح على حماية صفحة من هجوم البرمجة النصية للمواقع. سنشرح كيفية إنشاء مثل هذه الترويسات بشكل صحيح، ونقدم أمثلة عليها.

يتم تحديد السياسة باستخدام عدد من التعليمات، يصف كل منها السياسة لنوع مورد معين أو مجال معين. يجب أن تتضمن سياستك التعليمة **default-src**، والتي تحدد المصدر الاحتياطي لأنواع الموارد الأخرى عندما لا يكون لديهم سياسات خاصة بهم. يجب أن تتضمن السياسة التعليمة **default-src** أو **script-src** لمنع تشغيل البرامج النصية السطريّة المضمّنة، وكذلك حظر استخدام تابع **eval**. يجب أن تتضمن السياسة التعليمة **default-src** أو **style-src** لتقييد تطبيق الأنماط المضمنة في عنصر **<style>** أو خاصية **style**.

أمثلة:

مثال 1:

مدير التطبيق يريد لجميع الموارد أن تأتي من أصل الموقع ذاته (هذا لا يتضمن النطاقات الجزئية)

```
Content-Security-Policy: default-src 'self'
```

مثال 2:

مدير التطبيق يريد السماح بالمحتوى من مجال موثوق به وجميع نطاقاته الفرعية

```
Content-Security-Policy: default-src 'self' *.trusted.com
```

مثال 3:

يريد مسؤول موقع الويب السماح لمستخدمي تطبيق الويب بتضمين صور من أي أصل في المحتوى الخاص بهم، ولكن تقييد مصدر وسائط الصوت أو الفيديو بمزود موثوق، وجميع البرامج النصية فقط من مخدم معين يستضيف شيفرات موثوقة.

```
Content-Security-Policy: default-src 'self'; img-src *; media-src
media1.com media2.com; script-src userscripts.example.com
```

هنا، افتراضيًا، يُسمح بالمحتوى فقط من أصل المستند، مع الاستثناءات التالية:

قد يتم تحميل الصور من أي مكان (لاحظ الإشارة "*").

لا يُسمح بالوسائط إلا من **media1.com** و **media2.com** (وليس من النطاقات الفرعية لتلك المواقع).

لا يُسمح بالسكريبت إلا من **userscripts.example.com**

مثال 4:

يريد مسؤول موقع ويب لأحد المواقع المصرفية عبر الإنترنت التأكد من تحميل جميع محتوياته باستخدام **TLS**، لمنع المهاجمين من التنصت على الطلبات.

```
Content-Security-Policy: default-src
https://onlinebanking.jumbobank.com
```

يسمح المخدم فقط بالوصول إلى المستندات التي يتم تحميلها بشكل خاص عبر **HTTPS** من خلال موقع **onlinebanking.jumbobank.com**

مثال 5:

بشكل افتراضي، لا يتم إرسال تقارير الانتهاك للسياسة المحددة. لتمكين الإبلاغ عن الانتهاكات، تحتاج إلى استخدام المعرف "**report-uri**"، مع تحديد عنوان واحد على الأقل لتسليم التقارير إليه:

```
Content-Security-Policy: default-src 'self'; report-uri
http://reportcollector.example.com/collector.cgi
```

ثم تحتاج إلى إعداد الخادم الخاص بك لتلقي التقارير؛ يمكنك تخزينها أو معالجتها بأي طريقة تشعر أنها مناسبة.

كمثال على تقرير الانتهاك المرسل:

```
{
  "csp-report": {
    "document-uri": "http://example.com/signup.html",
    "referrer": "",
    "blocked-uri": "http://example.com/css/style.css",
    "violated-directive": "style-src cdn.example.com",
    "original-policy": "default-src 'none'; style-src
cdn.example.com; report-uri /_/csp-reports"
  }
}
```

المصدر:

https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP

قائمة الاختصارات

1	**URI**	Uniform Resource Identifier
2	**HTTP**	Hyper Text Transfer Protocol
3	**DOM**	Document Object Model
4	**AJAX**	Asynchronous JavaScript + XML
5	**XML**	Extensible Markup Language
6	**JSON**	JavaScript Object Notation
7	**SVG**	Scalable Vector Graphics
8	**SOP**	Same-Origin Policy
9	**SQL**	Structured Query Language
10	**OWASP**	Open Web Application Security Project
11	**CWE/SANS**	Common Weakness Enumeration
12	**CSP**	Content Security Policy
13	**MitM**	Man in the Middle
14	**HTTPS**	Hypertext Transfer Protocol Secure
15	**WPA**	Wi-Fi Protected Access
16	**EAP**	Extensible Authentication Protocol
17	**TLS**	Transport Layer Security
18	**RC4**	Rivest Cipher 4
19	**IETF**	Internet Engineering Task Force
20	**BCP**	Best Current Practice
21	**CA**	Certificate Authority
22	**PKI**	Public Key Infrastructure
23	**OCSP**	Online Certificate Status Protocol
24	**HSTS**	HTTP Strict Transport Security
25	**CT**	Certificate Transparency
26	**HPKP**	HTTP Public Key Pinning
27	**TOFU**	Trust on First Use

28	**CSRF**	Cross Site Request Forgery
29	**CORS**	Cross Origin Resource Sharing
30	**XFO**	X-Frame-Options
31	**IE**	Internet Explorer
32	**CSP**	Content Security Policy
33	**XSS**	Cross Site Scripting
34	**IP**	Internet Protocol
35	**HMAC**	Hash-based Message Authentication Code
36	**SRI**	Sub Resource Integrity
37	**CDN**	Content Delivery Network
38	**API**	Application Programming Interface
39	**NSA**	National Security Agency
40	**CWSS**	Common Weakness Scoring System

المراجع

[1] "FLYING PIG: The NSA Is Running Man In The Middle Attacks Imitating Google's Servers | Techdirt." https://www.techdirt.com/articles/20130910/10470024468/flying-pig-nsa-is-running-man-middle-attacks-imitating-googles-servers.shtml.

[2] "LinkedIn Breach Exposes Light Security Even at Data Companies - The New York Times." https://www.nytimes.com/2012/06/11/technology/linkedin-breach-exposes-light-security-even-at-data-companies.html?pagewanted=all.

[3] "Adobe Hacked – Customers' Card Details and Adobe Source Code Stolen - Infosecurity Magazine." https://www.infosecurity-magazine.com/news/adobe-hacked-customers-card-details-and-adobe/.

[4] "Yahoo Investigates Password Breach - WSJ." https://www.wsj.com/articles/SB10001424052702304373804577522613740363638.

[5] "CSRF Vulnerability in eBay Allows Hackers to Hijack User Accounts – Video." https://news.softpedia.com/news/CSRF-Vulnerability-in-eBay-Allows-Hackers-to-Hijack-User-Accounts-Video-383316.shtml.

[6] "How Reuters got compromised by the Syrian Electronic Army." https://medium.com/@FredericJacobs/the-reuters-compromise-by-the-syrian-electronic-army-6bf570e1a85b.

[7] "2018 Norton Report." https://www.nortonlifelock.com/content/dam/nortonlifelock/docs/about/2018-norton-lifelock-cyber-safety-insights-report-global-results-en.pdf.

[8] "W3C DAS WG » Home." https://www.w3.org/das/.

[9] "Threat Reports Archives | WhiteHat Security." https://www.whitehatsec.com/resources-category/threat-reports/.

[10] T. Van Goethem, P. Chen, N. Nikiforakis, L. Desmet, and W. Joosen, 2014-, "Large-scale security analysis of the web: Challenges and findings," vol. 8564 LNCS, doi: 10.1007/978-3-319-08593-7_8.

[11] A. Alarifi, M. Alsaleh, and A. M. Al-Salman, 2013-, "Security analysis of top visited Arabic Web sites," *Int. Conf. Adv. Commun. Technol. ICACT*, no. ii, pp. 173–178.

[12] N. Nikiforakis *et al.*, 2012-, "You Are What You Include: Large-scale Evaluation of Remote JavaScript Inclusions," *Proc. 2012 ACM Conf. Comput. Commun. Secur. - CCS '12*, p. 736, doi: 10.1145/2382196.2382274.

[13] M. Vasek and T. Moore, "Identifying Risk Factors for Webserver Compromise."

[14] "National Agency for Network Services." http://www.nans.gov.sy/ar/page/security_awareness_and_studies.

[15] P. De Ryck, L. Desmet, F. (Frank) Piessens, and M. Johns, *Primer on client-side web security*. .

[16] "Web Storage (Second Edition)." https://www.w3.org/TR/webstorage/.

[17] "Indexed Database API." https://www.w3.org/TR/2015/REC-IndexedDB-20150108/.

[18] "File API." https://www.w3.org/TR/FileAPI/.

[19] P. J. Leach, T. Berners-Lee, J. C. Mogul, L. Masinter, R. T. Fielding, and J. Gettys, "Hypertext Transfer Protocol -- HTTP/1.1."

[20] "RFC 6265 - HTTP State Management Mechanism." https://tools.ietf.org/html/rfc6265.

[21] "HTML Standard." https://html.spec.whatwg.org/multipage/web-messaging.html.

[22] "Cross-Origin Resource Sharing." https://www.w3.org/TR/cors/.

[23] "Category:OWASP Top Ten Project - OWASP." https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project.

[24] "CWE - 2019 CWE Top 25 Most Dangerous Software Errors." http://cwe.mitre.org/top25/archive/2019/2019_cwe_top25.html.

[25] "Content Security Policy Level 2." https://www.w3.org/TR/CSP2/.

[26] "NoScript - JavaScript/Java/Flash blocker for a safer Firefox experience! - what is it? - InformAction." https://noscript.net/.

[27] "IEEE 802.11i-2004 - IEEE Standard for information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements-Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer

(PHY) specifications: Amendment 6: Medium Access Control (MAC) Security Enhancements.” https://standards.ieee.org/standard/802_11i-2004.html.

[28] “RFC 5247 - Extensible Authentication Protocol (EAP) Key Management Framework.” https://datatracker.ietf.org/doc/rfc5247/.

[29] “RFC 8446 - The Transport Layer Security (TLS) Protocol Version 1.3.” https://tools.ietf.org/html/rfc8446.

[30] N. J. Alfardan and K. G. Paterson, 2013-, “Lucky Thirteen: Breaking the TLS and DTLS Record Protocols,” doi: 10.1109/SP.2013.42.

[31] N. J. Alfardan, D. J. Bernstein, K. G. Paterson, B. Poettering, and J. C. N. Schuldt, “On the Security of RC4 in TLS and WPA *,” 2013-.

[32] “ChaCha20 and Poly1305 based Cipher Suites for TLS.” https://tools.ietf.org/id/draft-agl-tls-chacha20poly1305-03.html.

[33] “Alexa Top 1 Million Analysis - February 2019.” https://scotthelme.co.uk/alexa-top-1-million-analysis-february-2019/.

[34] “Official Google Webmaster Central Blog [EN]: HTTPS as a ranking signal.” https://webmasters.googleblog.com/2014/08/https-as-ranking-signal.html.

[35] “Recommendations for secure use of TLS and DTLS.” https://tools.ietf.org/html/rfc7525.

[36] “Qualys SSL Labs.” https://www.ssllabs.com/.

[37] M. Marlinspike, “New Tricks For Defeating SSL In Practice.”

[38] M. Marlinspike, “Sslstrip.” https://moxie.org/software/sslstrip/.

[39] “RFC 5280 - Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile.” https://tools.ietf.org/html/rfc5280.

[40] “The EFF SSL Observatory | Electronic Frontier Foundation.” https://www.eff.org/observatory.

[41] “ImperialViolet - Revocation checking and Chrome’s CRL.” https://www.imperialviolet.org/2012/02/05/crlsets.html.

[42] R. Prins, H. Hoogstraaten, D. Niggebrugge, and D. Heppener, 2012-, “Black Tulip: Report of the investigation into the DigiNotar Certificate Authority breach,” pp. 1–101, doi: 10.13140/2.1.2456.7364.

[43] "HTTPS Everywhere | Electronic Frontier Foundation." https://www.eff.org/https-everywhere.

[44] "HProxy: Client-Side Detection of SSL Stripping Attacks | SpringerLink." https://link.springer.com/chapter/10.1007/978-3-642-14215-4_12.

[45] C. Jackson and A. Barth, "ForceHTTPS: protecting high-security web sites from network attacks," in *Proceeding of the 17th International Conference on World Wide Web 2008, WWW'08*, 2008, pp. 525–533, doi: 10.1145/1367497.1367569.

[46] "RFC 6797 - HTTP Strict Transport Security (HSTS)." https://tools.ietf.org/html/rfc6797.

[47] "HSTS Preload List Submission." https://hstspreload.org/.

[48] "RFC 6962 - Certificate Transparency." https://tools.ietf.org/html/rfc6962.

[49] "RFC 7469 - Public Key Pinning Extension for HTTP." https://tools.ietf.org/html/rfc7469#section-2.1.

[50] L.-S. Huang, A. Rice, E. Ellingsen, and C. Jackson, "Analyzing Forged SSL Certificates in the Wild."

[51] T. Duong and J. Rizzo, "Here Come The ⊕ Ninjas," 2011-.

[52] "CRIME - Google Slides." https://docs.google.com/presentation/d/11eBmGiHbYcHR9gL5nDyZChu_-lCa2GizeuOfaLU2HOU/edit?pli=1#slide=id.g1d134dff_1_10.

[53] Y. Gluck, N. Harris, and A. Prado, "BREACH: REVIVING THE CRIME ATTACK."

[54] "Heartbleed - Schneier on Security." https://www.schneier.com/blog/archives/2014/04/heartbleed.html.

[55] "Google Online Security Blog: Modernizing Transport Security." https://security.googleblog.com/2018/10/modernizing-transport-security.html.

[56] "Google Online Security Blog: Chrome UI for Deprecating Legacy TLS Versions." https://security.googleblog.com/2019/10/chrome-ui-for-deprecating-legacy-tls.html.

[57] "Cross-site Request Forgery - Exploitation & Prevention | Netsparker." https://www.netsparker.com/blog/web-security/csrf-

cross-site-request-forgery/.

[58] "Gmail CSRF vulnerability explained | Daniel Hepper." https://daniel.hepper.net/blog/2008/11/gmail-csrf-vulnerability-explained/.

[59] A. Barth, C. Jackson, and J. C. Mitchell, "Robust defenses for cross-site request forgery," in *Proceedings of the ACM Conference on Computer and Communications Security*, 2008, pp. 75–87, doi: 10.1145/1455770.1455782.

[60] "RFC 6454 - The Web Origin Concept." https://tools.ietf.org/html/rfc6454.

[61] J. Burns, "Cross Site Request Forgery: An introduction to a common web application weakness."

[62] R. Pelizzi and R. Sekar, *A Server-and Browser-Transparent CSRF Defense for Web 2.0 Applications* *. 2011.

[63] S. Lekies, W. Tighzert, and M. Johns, "Towards stateless, client-side drivenCross-Site Request Forgery protection forW eb applications."

[64] "draft-west-first-party-cookies-07 - Same-site Cookies." https://tools.ietf.org/html/draft-west-first-party-cookies-07.

[65] M. Johns and J. Winter, "RequestRodeo: Client Side Protection against Session Riding."

[66] P. De Ryck, L. Desmet, W. Joosen, and F. Piessens, "Automatic and Precise Client-Side Protection against CSRF Attacks."

[67] J. Samuel and B. Zhang, "Request policy: Increasing web browsing privacy through control of cross-site requests," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2009, vol. 5672 LNCS, pp. 128–142, doi: 10.1007/978-3-642-03168-7_8.

[68] B. S. Y. Fung and P. P. C. Lee, "A privacy-preserving defense mechanism against request forgery attacks," in *Proc. 10th IEEE Int. Conf. on Trust, Security and Privacy in Computing and Communications, TrustCom 2011, 8th IEEE Int. Conf. on Embedded Software and Systems, ICESS 2011, 6th Int. Conf. on FCST 2011*, 2011, pp. 45–52, doi: 10.1109/TrustCom.2011.10.

[69] "What's ABE?" https://noscript.net/abe/.

[70] "draft-west-cookie-incrementalism-00 - Incrementally Better Cookies." https://tools.ietf.org/html/draft-west-cookie-incrementalism-00.

[71] "Explaining the 'Don't Click' Clickjacking Tweetbomb." http://softwareas.com/explaining-the-dont-click-clickjacking-tweetbomb/.

[72] "Clickjacking: Attacks and Defenses | USENIX." https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/huang.

[73] "Framing Attacks on Smart Phones and Dumb Routers: Tap-jacking and Geo-localization Attacks | Request PDF." https://www.researchgate.net/publication/228414657_Framing_Attacks_on_Smart_Phones_and_Dumb_Routers_Tap-jacking_and_Geo-localization_Attacks.

[74] "RFC 7034 - HTTP Header Field X-Frame-Options." https://tools.ietf.org/html/rfc7034.

[75] P. Eckersley, "How Unique Is Your Web Browser?"

[76] G. Acar *et al.*, "FPDetective: Dusting the Web for Fingerprinters," doi: 10.1145/2508859.2516674.

[77] B. Adida, *SessionLock: Securing Web Sessions against Eavesdropping*. .

[78] P. A. Hallgren, D. T. Mauritzson, and A. Sabelfeld, "GlassTube: A Lightweight Approach to Web Application Integrity."

[79] I. Dacosta, S. Chakradeo, M. Ahamad, and P. Traynor, "One-Time Cookies: Preventing Session Hijacking Attacks with Stateless Authentication Tokens."

[80] A. Birgisson, J. Gibbs Politz, U. Erlingsson, A. Taly, M. Vrable, and M. Lentczner, "Macaroons: Cookies with Contextual Caveats for Decentralized Authorization in the Cloud," 2014-.

[81] N. Nikiforakis, W. Meert, Y. Younan, M. Johns, and W. Joosen, "SessionShield: Lightweight protection against session hijacking," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2011, vol. 6542 LNCS, pp. 87–100, doi: 10.1007/978-3-642-19125-1_7.

[82] P. De Ryck, N. Nikiforakis, L. Desmet, F. Piessens, and W. Joosen, "SERENE: Self-reliant client-side protection against session fixation,"

in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2012, vol. 7272 LNCS, pp. 59–72, doi: 10.1007/978-3-642-30823-9_5.

[83] "Session Management · OWASP Cheat Sheet Series." https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html.

[84] M. Johns, B. Braun, M. Schrank, and J. Posegga, "Reliable protection against session fixation attacks," in *Proceedings of the ACM Symposium on Applied Computing*, 2011, pp. 1531–1537, doi: 10.1145/1982185.1982511.

[85] "How Apple and Amazon Security Flaws Led to My Epic Hacking | WIRED." https://www.wired.com/2012/08/apple-amazon-mat-honan-hacking/.

[86] "How I Lost My $50,000 Twitter Username - Naoki Hiroshima - Medium." https://medium.com/@N/how-i-lost-my-50-000-twitter-username-24eb09e026dd.

[87] "Tabnabbing: A New Type of Phishing Attack - Internet O'Clock." https://abain.typepad.com/internet_oclock/2010/06/tabnabbing-a-new-type-of-phishing-attack.html.

[88] "About Touch ID advanced security technology - Apple Support." https://support.apple.com/en-us/HT204587.

[89] D. Berg, "How To Use Your Laptop's Fingerprint Reader | Laptop Mag." https://www.laptopmag.com/articles/how-to-use-your-fingerprint-reader.

[90] "7 ways to beat fingerprint biometrics | ITworld." https://www.itworld.com/article/2823742/120606-10-ways-to-beat-fingerprint-biometrics.html.

[91] "Security Features and Tips | Facebook Help Center." https://www.facebook.com/help/285695718429403.

[92] "Add or remove trusted computers - Computer - Google Account Help." https://support.google.com/accounts/answer/2544838?hl=en.

[93] "Mercury Center Archive Search Results." https://simson.net/clips/1995/95.SJMN.AOL_Hackers.html.

[94] S. Egelman, L. F. Cranor, and J. Hong, "You've Been Warned: An Empirical Study of the Effectiveness of Web Browser Phishing

Warnings," 2008-.

[95] R. Dhamija, J. D. Tygar, and M. Hearst, "Why Phishing Works."

[96] R. Dhamija and J. D. Tygar, "The Battle Against Phishing: Dynamic Security Skins."

[97] M. Wu, R. C. Miller, and S. L. Garfinkel, "Do Security Toolbars Actually Prevent Phishing Attacks?," 2006-.

[98] N. Nikiforakis, A. Makridakis, E. Athanasopoulos, and E. P. Markatos, "Alice, what did you do last time? Fighting Phishing Using Past Activity Tests."

[99] N. Chou, R. Ledesma, Y. Teraguchi, D. Boneh, and J. C. Mitchell, "Client-side defense against web-based identity theft."

[100] P. De Ryck, N. Nikiforakis, L. Desmet, and W. Joosen, "TabShots: Client-side detection of tabnabbing attacks," in *ASIA CCS 2013 - Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, 2013, pp. 447–455, doi: 10.1145/2484313.2484371.

[101] "360 Million Stolen Credentials and 1.25 Billion Email Addresses Found on the Black Market - Infosecurity Magazine." https://www.infosecurity-magazine.com/news/360-million-stolen-credentials-and-125-billion/.

[102] "Google Safe Browsing | Google Developers." https://developers.google.com/safe-browsing/.

[103] L. Wenyin, G. Huang, L. Xiaoyue, Z. Min, and X. Deng, "Detection of phishing webpages based on visual similarity," in *14th International World Wide Web Conference, WWW2005*, 2005, pp. 1060–1061, doi: 10.1145/1062745.1062868.

[104] "OWASP Top 10-2017," 2017-.

[105] "Security researcher finds critical XSS bug in Google's Invoice Submission Portal | ZDNet." https://www.zdnet.com/article/security-researcher-finds-critical-xss-bug-in-googles-invoice-submission-portal/.

[106] "OWASP Java Encoder Project - OWASP." https://www.owasp.org/index.php/OWASP_Java_Encoder_Project.

[107] "HTML Purifier - Filter your HTML the standards-compliant way!"

http://htmlpurifier.org/.

[108] "(PDF) ScriptGard: Automatic Context-Sensitive Sanitization for Large-Scale Legacy Web Applications." https://www.researchgate.net/publication/221668938_ScriptGard_Automatic_Context-Sensitive_Sanitization_for_Large-Scale_Legacy_Web_Applications.

[109] "Context-Sensitive Auto-Sanitization in Web Templating Languages Using Type Qualifiers | Request PDF." https://www.researchgate.net/publication/220269493_Context-Sensitive_Auto-Sanitization_in_Web_Templating_Languages_Using_Type_Qualifiers.

[110] "XSS Auditor - The Chromium Projects." https://www.chromium.org/developers/design-documents/xss-auditor.

[111] "Subresource Integrity." https://www.w3.org/TR/SRI/.

[112] "Postcards from the post-XSS world." http://lcamtuf.coredump.cx/postxss/.

[113] "Yahoo's malware-pushing ads linked to larger malware scheme | PCWorld." https://www.pcworld.com/article/2086700/yahoo-malvertising-attack-linked-to-larger-malware-scheme.html.

[114] J. Magazinius, P. H. Phung, and D. Sands, "Safe wrappers and sane policies for self protecting JavaScript," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2012, vol. 7127 LNCS, pp. 239–255, doi: 10.1007/978-3-642-27937-9_17.

[115] P. H. Phung, D. Sands, and A. Chudnov, "Lightweight self-protecting JavaScript," in *Proceedings of the 4th International Symposium on ACM Symposium on Information, Computer and Communications Security, ASIACCS'09*, 2009, pp. 47–60, doi: 10.1145/1533057.1533067.

[116] P. Agten, S. Van Acker, Y. Brondsema, P. H. Phung, L. Desmet, and F. Piessens, "JSand: Complete client-side sandboxing of third-party JavaScript without browser modifications," in *ACM International Conference Proceeding Series*, 2012, pp. 1–10, doi: 10.1145/2420950.2420952.

[117] L. Ingram and M. Walfish, "TreeHouse: JavaScript sandboxes to help Web developers help themselves."

[118] J. Mickens, "Pivot: Fast, synchronous mashup isolation using generator chains," in *Proceedings - IEEE Symposium on Security and Privacy*, 2014, pp. 261–275, doi: 10.1109/SP.2014.24.

[119] "How the NSA Attacks Tor/Firefox Users With QUANTUM and FOXACID - Schneier on Security." https://www.schneier.com/blog/archives/2013/10/how_the_nsa_att.html.

[120] "VU#189754 - Microsoft Internet Explorer buffer overflow in PNG image rendering component." https://www.kb.cert.org/vuls/id/189754/.

[121] "Oracle Java Contains Multiple Vulnerabilities | CISA." https://www.us-cert.gov/ncas/alerts/TA13-064A.

[122] Ú. Erlingsson, Y. Younan, and F. Piessens, "Low-Level Software Security by Example," in *Handbook of Information and Communication Security*, Springer Berlin Heidelberg, 2010, pp. 633–658.

[123] T. D. Google, S. Gmbh, and S. Frei, "Why Silent Updates Boost Security."

[124] C. Kolbitsch, B. Livshits, B. Zorn, and C. Seifert, "Rozzle: De-cloaking Internet malware," in *Proceedings - IEEE Symposium on Security and Privacy*, 2012, pp. 443–457, doi: 10.1109/SP.2012.48.

[125] P. Laskov and N. Šrndić, "Static detection of malicious JavaScript-bearing PDF documents," in *ACM International Conference Proceeding Series*, 2011, pp. 373–382, doi: 10.1145/2076732.2076785.

[126] "Measuring Pay-per-Install: The Commoditization of Malware Distribution | USENIX." https://www.usenix.org/conference/usenix-security-11/measuring-pay-install-commoditization-malware-distribution.

[127] B. Stone-Gross, R. Abman, R. A. Kemmerer, C. Kruegel, D. G. Steigerwald, and G. Vigna, "The Underground Economy of Fake Antivirus Software," in *Economics of Information Security and Privacy III*, Springer New York, 2013, pp. 55–78.

[128] "Please read: Security Issue on AMO | Mozilla Add-ons Blog." https://blog.mozilla.org/addons/2010/02/04/please-read-security-issue-on-amo/.

[129] "Adware vendors buy Chrome Extensions to send ad- and malware-filled updates | Ars Technica." https://arstechnica.com/information-

technology/2014/01/malware-vendors-buy-chrome-extensions-to-send-adware-filled-updates/.

[130] "(PDF) Protecting Browsers from Extension Vulnerabilities." https://www.researchgate.net/publication/221655526_Protecting_Browsers_from_Extension_Vulnerabilities.

[131] "An Evaluation of the Google Chrome Extension Security Architecture | USENIX." https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/carlini.

[132] S. Van Acker, N. Nikiforakis, L. Desmet, F. Piessens, and W. Joosen, "Monkey-in-the-browser: Malware and vulnerabilities in augmented browsing script markets," in *ASIA CCS 2014 - Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security*, 2014, pp. 525–530, doi: 10.1145/2590296.2590311.

[133] B. S. Lerner, L. Elberty, N. Poole, and S. Krishnamurthi, "Verifying web browser extensions' compliance with private-browsing mode," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2013, vol. 8134 LNCS, pp. 57–74, doi: 10.1007/978-3-642-40203-6_4.

[134] "Feature Policy." https://w3c.github.io/webappsec-feature-policy/.

[135] "Google Search Scraper (apify/google-search-scraper) · Apify." https://apify.com/apify/google-search-scraper.

[136] "Requests: HTTP for Humans™ — Requests 2.22.0 documentation." https://requests.readthedocs.io/en/latest/.

[137] "Beautiful Soup Documentation — Beautiful Soup 4.4.0 documentation." https://beautiful-soup-4.readthedocs.io/en/latest/.

[138] "SSLyze Python API — SSLyze 2.1.0 documentation." https://nabla-c0d3.github.io/sslyze/documentation/.

[139] "draft-ietf-httpbis-rfc6265bis-03 - Cookies: HTTP State Management Mechanism." https://tools.ietf.org/html/draft-ietf-httpbis-rfc6265bis-03#section-4.1.

[140] "Public Suffix List." https://publicsuffix.org/.

Abstract:

With the Internet occupying a permanent place in our lives, the security aspects have gained more importance, and this is confirmed by the large media coverage of serious security incidents.

Over recent years, the center of gravity has shifted from server to user, as the browser has become an integrated execution environment for dynamic and complex applications. Unfortunately, the attacker's focus has also shifted towards browser-based attacks and threatened to the customer's platform. It is only natural for this transition that countermeasures and security policies evolve in this direction.

Also, to assess a site's security, site administrators often turn to consulting firms for penetration testing and code review. However, it is difficult for an external party, such as the government or regulatory organizations, to assess the safety of a site externally, especially if this evaluation is carried out on a wide range, such as including a large number of sites affiliated with a country or a specific industrial sector. This type of evaluation is necessary as citizens increasingly rely day to day on specific applications. This is similar to, for example, the necessity of mandatory evaluation of the structural safety of buildings in order to protect citizens from future disasters that could have been avoided.

In this thesis, we will first highlight the evolution from server-side applications to contemporary client-side applications that provide a different user experience. We will explore the concepts involved under these applications and explain the importance of security from the client-side, and we will explain the risks and the latest technologies and research to address them

Then we will conduct a study on the extent to which the Syrian websites adopt security mechanisms from the client side, as well as provide a reference model for the current state of network security, and we will also introduce a security scoring system to estimate the level of security provided by each site.

Keywords:

Application security, web applications, client-side security, Syrian sites, empirical study, security evaluation.

Syrian Arab Republic
Al Baath University
Faculty of Informatics Engineering
Department of Software Engineering & Information Systems

Client-Side Web Application Security

A thesis submitted in partial fulfilment of the requirements for the of Master's Degree in Software Engineering and Informatics Engineering

Prepared by

Eng. Tuhama Ghassan Qlyshi

Supervised by

Prof. Dr. Naser Abu Saleh

Professor–Department of Software Engineering and Information Systems

1441 A.H - 2020 A.D