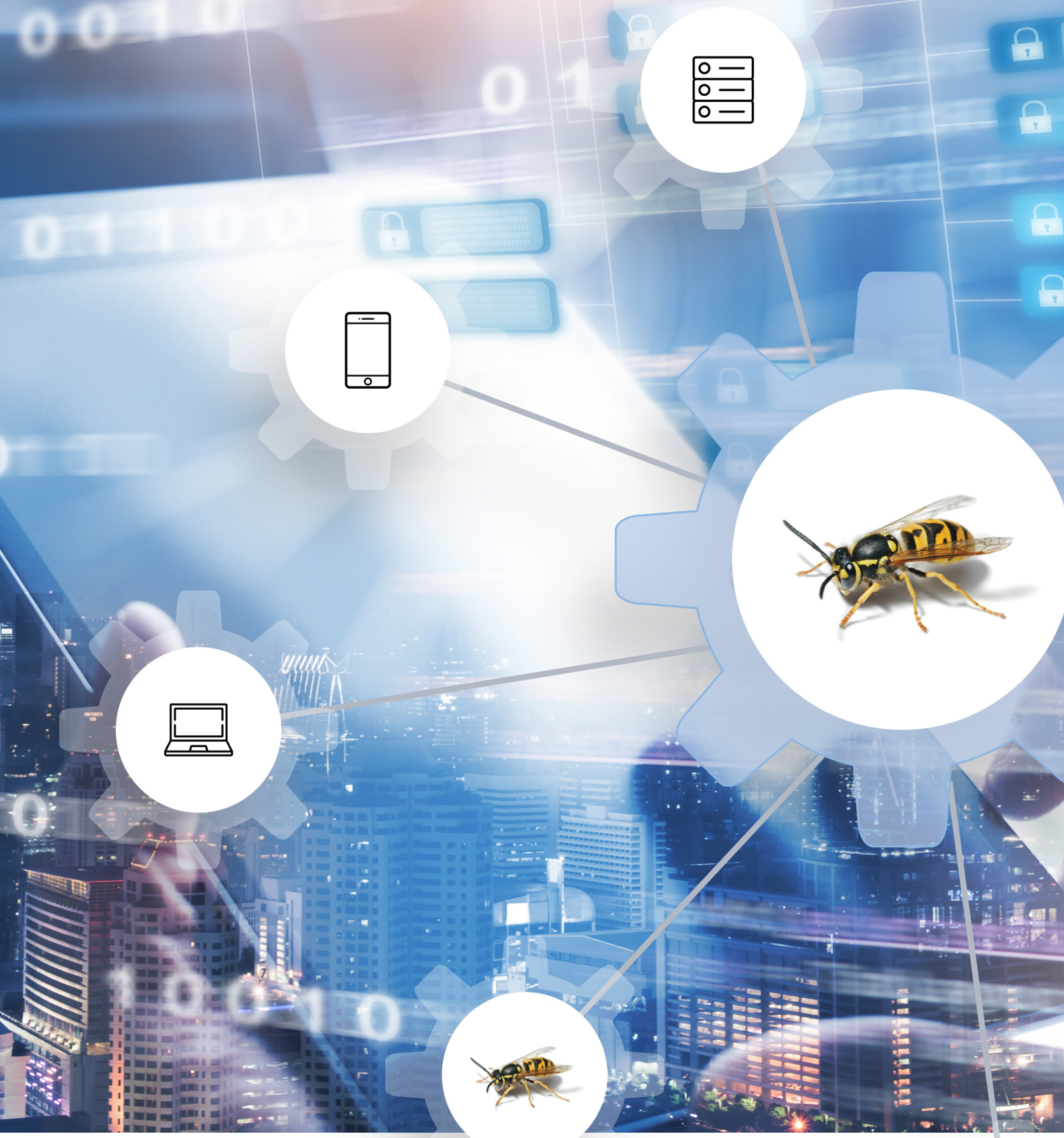


OWASP API Security Top 10 2019

أهم عشرة مخاطر امنية تستهدف واجهة برمجة التطبيقات (API)

جدول المحتويات

عن منظمة أواسب

هـو مشروع/مجتمـع لامن تطبيقـات الـويب وهـو مفتـوح المصـدر ويـهدف الى تمكين المؤسسات من تطوير او شراء او صيانة تطبيقاتها بشكل امن و موثوق.

في مجتمع OWASP ستجد :

- معايير و ادوات التطبيقات الامنة
- كتب ومراجع كاملة عن اختبار تطبيقات الويب و التطوير الامن ومراجعة الشفرة المصدرية
- العروض التقديمية
- ملخصات
- مكتبة المعايير الامنية والضوابط
- الفروع المحلية حول العالم
- البحوث
- المؤتمرات حول العالم
- القائمة البريدية

للاستزادة تفضل https://www.owasp.org

إن جميع الأدوات والوثائق والمنتديات والمنظمات الفرعية لمنظمة (أواسب) هي مجانيـــة ومفتوحـــة لجميـــع المهتمين بتطـــوير أمن التطبيقات . نقدم أمن التطبيقات كمشكلة تتضمن العامل البشري، والإجـراءات، والتقنيـة؛ وذلـك لأن أفضـل الأسـاليب فعاليـة 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 نـدعم الإسـتخدام الـواعي للتقنيات الأمنية التجارية . على غـرار الكثير من مشاريع البرمجيات مفتوحة المصدر، فإن)أواسب(تقدم أنواع كثيرة من المواد بشكل تعاوني ومفتوح

مؤسسـة (أواسب) هي منشـأة غـير ربحيـة تضـمن النجـاح المسـتمر للمشروع . تقريباً ، جميع المنتسبين إلى (أواسب) هم من المتطوعين، بمن فيهم أعضـاء المجلس، واللجـان العالميـة، وقـادة المنظمـات الفرعيـة، وقـادة المشـاريع وأعضـائها . نحن نـدعم الأبحـاث الأمنيـة الإبداعية بالمنح وتوفير البنية التحتية

إنضم إلينا

جدول المحتويات

مقدمة

تعتبر واجهة برمجة التطبيقات (API) أحد العناصر الأساسية للابتكار في عالم التطبيقات. حيث نجدها في التطبيقات البنكية وتجارة التجزئة والنقل وصولاً الى إنترنت الأشياء والمركبات ذاتية القيادة والمدن الذكية، تعد واجهات برمجة التطبيقات API جزءًا مهمًا من التطبيقات وخصوصاً الخاصة بالهاتف المحمول الحديثة والبرمجيات كخدمة SaaS وتطبيقات الويب ويمكن استخدمها في تطبيقات المستخدمين والشركاء والتطبيقات الداخلية.

بطبيعة استخدام واجهة التطبيقات API باستعراض بعض المعلومات الحساسة و المعلومات الشخصية لهذا السبب نجد ان API هو هدف أساسي للمهاجمين بشكل متزايد لذلك بدون تأمين البيئة الخاصة بواجهة برمجة التطبيقات سيصبح التطوير السريع مستحيل.

على الرغم من وجود مخاطر متعددة على تطبيقات الويب ومنها أعلى عشر مخاطر تستهدف تطبيقات الويب، وبالإضافة لذلك المخاطر التي تستهدف واجهة برمجة تطبيقات الويب والتي يستوجب علينا التركيز عليها لإيجاد حلول استراتيجية من شانها تخفيف المخاطر ونقاط الضعف المرتبطة مع واجهة برمجة التطبيقات.

إذا كنت معتادًا على مشروع OWASP Top 10 ، فستلاحظ أوجه التشابه بين كلا المستندين: إنهما مخصصان للقراءة والاعتماد. إذا كنت جديدًا في سلسلة OWASP Top 10 ، فقد يكون من الأفضل لك قراءة أقسام مخاطر الأمان والمنهجية والبيانات الخاصة بواجهة برمجة التطبيقات (API) قبل الانتقال إلى قائمة المخاطر 10 هنا.

يمكنك المساهمة في OWASP API Security Top 10 بأسئلتك وتعليقاتك وأفكارك في مستودع مشروع GitHub:

- https://github.com/OWASP/API-Security/issues
- https://github.com/OWASP/API-Security/blob/master/CONTRIBUTING.md

تستطيع الوصول الى الوثيقة OWASP API Security Top 10 من هنا :

- https://www.owasp.org/index.php/OWASP_API_Security_Project
- https://github.com/OWASP/API-Security

نود أن نشكر جميع المساهمين الذين جعلوا هذا المشروع متوفر لكم بجهودهم ومساهماتهم. تم سردها جميعًا في قسم الشكر والتقدير.
شكرا لك!

مدخل

مرحباً بك في أهم عشرة مخاطر امنية تستهدف واجهة برمجة التطبيقات (API)

مرحبًا بك في الإصدار الأول من OWASP API Security Top 10. إذا كنت على دراية بسلسلة OWASP Top 10 ، ستلاحظ أوجه التشابه بينهم: حيث نوصي بقراءة OWASP Top 10 قبل الشروع وقراءة هذا المحتوى.

تلعب واجهات برمجة التطبيقات دورًا مهمًا جدًا في هندسة التطبيقات الحديثة. نظرًا لأن إنشاء الوعي الأمني في البرمجة الامنة والابتكار لهما خطوات مهمة ومختلفة، ومع ذلك فمن المهم التركيز على نقاط الضعف الأمنية الشائعة لواجهة برمجة التطبيقات API.

الهدف الأساسي من OWASP API Security Top 10 هو تثقيف المشاركين في تطوير وصيانة واجهة برمجة التطبيقات API.، على سبيل المثال ، المطورين أو المصممين أو مهندسين البنية التحتية أو المديرين أو المؤسسات.

في قسم المنهجية والبيانات ، يمكنك قراءة المزيد حول كيفية تم إنشاء الإصدار الأول. وما هو المتوقع من الإصدارات المستقبلية ، حيث نريد تمكين صناعة الأمن في برمجة واجهة التطبيقات API ، كما نشجع الجميع على المساهمة في طرح الأسئلة والتعليقات والأفكار من خلال القائمة البريدية.

ملاحظات عن الاصدار

هذا هو الإصدار الأول من OWASP API Security Top 10 ، والذي نخطط لتحديثه بشكل دوري ، كل ثلاث أو أربع سنوات.

على عكس هذا الإصدار ، في الإصدارات المستقبلية ، سنقوم بدعوة عامة للمشاركة لتمكين صناعة تطبيقات امنة ويكون جهد مشترك . في قسم المنهجية والبيانات ستجد المزيد من التفاصيل حول كيفية تم إنشاء هذا الإصدار. لمزيد من التفاصيل حول مخاطر الأمان ، يرجى الرجوع إلى قسم مخاطر أمان واجهة برمجة التطبيقات API.

من المهم أن ندرك أنه على مدى السنوات القليلة الماضية ، تغيرت بنية التطبيقات بشكل كبير. حيث تلعب واجهات برمجة التطبيقات API وتقوم حاليًا بدور مهم للغاية في هذه البنية الجديدة للخدمات المصغرة وتطبيقات الدخول ذات الصفحة الواحدة (SPA) وتطبيقات الأجهزة المحمولة وإنترنت الأشياء وما إلى ذلك.

ان ايجاد OWASP API Security Top 10 يحتاج الى جهدًا كبير بهدف خلق الوعي حول مشكلات أمان API الحديثة. وكما نكرر الشكر لجميع المتطوعين في انشاء هذه الوثيقة ، وجميعهم مدرجون في قسم الشكر والتقدير.

شكرا لك!

مخاطر برمجة واجهة التطبيقات

تم استخدام نموذج تقييم المخاطر الخاص بـ OWASP وذلك بهدف تحليل المخاطر

يلخص الجدول أدناه المصطلحات المرتبطة بدرجة المخاطر.

عوامل التهديد	الاستغلال	نقاط الضعف الامنية	اكتشاف الضعف الامني	التأثيرات التقنية	التأثيرات على الاعمال
خصائص API	بسيط 3	منتشر 4	بسيط 3	حرج 3	تحديد الاعمال
خصائص API	متوسط 2	عام 2	متوسط 2	متوسط 2	تحديد الاعمال
خصائص API	صعب 1	صعب 1	صعب 1	منخفض	تحديد الاعمال

ملاحظة:

هذا النهج لا يأخذ في الاعتبار احتمال وجود عامل التهديد. كما أنه لا يأخذ في الحسبان أيًا من التفاصيل الفنية المختلفة المرتبطة بتطبيقك يمكن لأي من هذه العوامل أن تؤثر بشكل كبير على الاحتمالية الإجمالية للمهاجم للعثور على ثغرة أمنية معينة واستغلالها. لا يأخذ هذا التصنيف في الاعتبار التأثير الفعلي على عملك. سيتعين على مؤسستك تحديد مقدار المخاطر الأمنية من التطبيقات وواجهات برمجة التطبيقات التي ترغب المؤسسة في قبولها في ضوء بيئتك التنظيمية. الغرض من OWASP API Security Top 10 ليس القيام بتحليل المخاطر هذا نيابة عنك.

المراجع :

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

مصادر خارجية:

- ISO 31000: Risk Management Std
- ISO 27001: ISMS
- NIST Cyber Framework (US)
- ASD Strategic Mitigations (AU)
- NIST CVSS 3.0
- Microsoft Threat Modeling Tool

أهم عشرة مخاطر امنية تستهدف واجهة برمجة التطبيقات (API)

API1:2019 خلل التفويض والصلاحيات	تقوم واجهة برمجة التطبيقات API الى كشف بعض المعلومات عن مصدر التعامل مع لطلبات وهو بالعادة يكون (endpoints). التي تتعامل مع الطلبات الناشئة مما قد يؤدي الى مشكلة في التحكم في مستوى صلاحيات الوصول ، حيث يجب ان يكون هناك مستوى صلاحيات محدد ومعرف ومحدود لكل طلب يتم إرساله الى مصدر البيانات بواسطة المستخدمين.
API2:2019 خلل في صلاحيات المستخدم	غالبًا ما يتم تنفيذ آليات المصادقة بشكل غير صحيح ، مما يسمح للمهاجمين باختراق معايير المصادقة أو استغلال الثغرات المنطقية في آلية عمل التطبيق مما تسمح له بانتحال هويات المستخدمين الاخرين بشكل مؤقت أو دائم. حيث ان اختراق النظام او آلية تحديد هوية المستخدم هو خطر على API بشكل عام.
API3:2019 خلل في استعراض البيانات	ان تنصيب التقنيات بدون مراعات تغير الإعدادات الافتراضية التي قد تودي الى الكشف عن خصائص ومعلومات وبيانات هامة ولا يجب الاعتماد على عوامل التصفية لدى المستخدم قبل عرضها.
API4:2019 ضعف في البنية التحتية و حد محاولات الطلبات	في كثير من الأحيان ، لا تفرض واجهات برمجة التطبيقات أي قيود على حجم أو عدد الموارد التي يمكن أن يطلبها العميل / المستخدم. ليس فقط يمكن لهذا تأثير على أداء الخادم API، بل قد يؤدي إلى هجمات حجب الخدمة (DoS)، وكذلك يمكن المهاجم من استخدام هجمة كسر كلمات المرور.
API5:2019 خلل في مستوى الصلاحيات والتفويض	تميل سياسات التحكم في الوصول المعقدة ذات المجموعات والأدوار المختلفة ، والفصل غير الواضح بينهم في الصلاحيات الإدارية والعادية ، إلى عيوب في التفويض والصلاحيات. والتي تمكن المهاجم من استغلال هذا الضعف في الوصول إلى المستخدمين الآخرين و تصعيد الصلاحيات الى صلاحيات الإدارية.
API6:2019 خلل في التعين او التعديل	يؤدي ادخال البيانات المقدمة من العميل على سبيل المثال ادخال البيانات في ملف (Json) دون عوامل تصفية او قوائم فلترة خاصة مبنية على قوائم بيضاء الى خلل في التعديل او التعين والذي يسمح للمهاجمين بقراءة بيانات او طلب معلومات غير مصرح بها.
API7:2019 الاعداد الخاطئ	عادة ما يكون الإعدادات الخاطئة او الاعتماد على الاعدادات الافتراضية او الاعدادات و التغيرات الغير مخطط لها مسبقاً او البيانات السحابية الغير مؤمنه او الاخطاء في اعدادات طلبات بروتوكول HTTP او مشاركة الموارد (CORS)ـ او رسائل الخطأ التفصيلية التي تحتوي على معلومات حساسة.
API8:2019 الحقن	تحدث عمليات استغلال الحقن NoSQL ،SQL و Command Injection.. الخ عند ارسال معلومات او بيانات او طلبات او اوامر الى المفسر حيث يتم خداع المفسر لطلب وتنفيذ تعليمات او الحصول على بيانات غير مصرح باستخدامها.
API9:2019 خلل في ادارة الاصول	تميل واجهات برمجة التطبيقات API الى الكشف عن مصادر البيانات (Endpoints). مما يجعل عمليات التوثيق في المستندات لجميع التغيرات في غاية الاهمية ويجب الحذر عند اجراءها، حيث ان اعدادات وتنصيب الخوادم بشكل صحيح عند تثبيت API مهم جداً في تقليل الاخطاء التي قد تؤدي الى الكشف عن البيانات على سبيل المثال الاصدار الخاص بـ API او وجهة معالج الاخطاء الخاصة به.
API10:2019 خلل في طريقة تسجيل الاحداث والمراقبة	ان التسجيل الغير صحيح للأحداث و المراقبة لها يؤدي الى ضعف عملية الاستجابة للحوادث، مما يسمح للمهاجم بالعودة مره اخرى او حتى البقاء داخل الشبكة او التنقل داخل الشبكة او الاطلاع و التلاعب و تسريب البيانات حيث تُظهر معظم الدراسات ان الوقت اللازم لاكتشاف الاختراقات يزيد عن 200 يوم وعادة ما يتم اكتشاف تلك الاختراقات من اطراف خارجية بدلاً من المراقبة بسبب ضعفها.

API1:2019 خلل التفويض والصلاحيات

خصائص API	قابلية الاستغلال: 3	الانتشار : 3	قابلية الاكتشاف : 2	التأثر التقني و تأثر الاعمال: 3	خصائص API
	يستطيع المهاجم استغلال نقاط الضعف في مصادر البيانات Endpoint المتأثرة بخلل في الصلاحيات من خلال التلاعب او التغير بالمعرف الفريد عند ارسال الطلبات. قد يؤدي كذلك الى الوصول غير المصرح به الى البيانات الحساسة. وتعتبر خلل تفويض الصلاحيات مشكلة شائعة جداً في التطبيقات بسبب عدم إمكانية الخوادم من تتبع العمليات التي يقوم بها المستخدم بشكل كامل. وحيث انه يعتمد بشكل كامل على إيصال كل معرف فريد لمصدر البيانات الذي يطلب منه.	يعتبر هذا الهجوم هو الأكثر شيوعاً على واجهات برمجة التطبيقات API. حيث ان استخدام مثل هذه الاليات شائعة جداً في التطبيقات الحديثة وواسعة الانتشار. حتى وان كانت صلاحيات الوصول في البنية التحتية للتطبيق مبنيه بشكل سليم، فقد ينسى المطورون استخدام تلك الصلاحيات في الوصول الى البيانات الحساسة وقد لا يتم اكتشاف نقاط الضعف المبنية على خلل صلاحيات الوصول من خلال عمليات المسح للبحث عن الثغرات بشكل آلي.	يمكن أن يؤدي الوصول غير المصرح به إلى الكشف عن البيانات لأطراف غير مصرح لها أو فقدان البيانات أو التلاعب بها وكذلك يمكن أن يؤدي الوصول غير المصرح به إلى إلى الاستيلاء الكامل على الحساب.	

هل واجهة برمجة التطبيقات (API) مصابة ؟

إن عمليات إدارة صلاحيات الوصول والتحكم بها عادة يبنى من خلال كتابة الاكواد البرمجية في المقام الأول بشكل سليم بحيث يستطيع المستخدم الوصول إلى البيانات المسموح له بالوصول لها. إن جميع مصادر البيانات الخاصة بـ API لها معرف وكائن وصلاحيات خاص ومرتبطة بها، وعند وجود أي إجراء على تلك المصادر أو الكائنات يجب أن يتم إستخدام تلك التصاريح. حيث يتم التحقق من صلاحيات المستخدم الذي قام بعملية تسجيل الدخول ومعرفة إذا كان لدية حق الوصول لأجراء أو إستعراض او تعديل البيانات. وعادة ما يؤدي الفشل في التحقق من هذه الالية إلى الكشف والتعديل عن معلومات وبيانات الغير مصرح به.

امثلة على سيناريوهات الهجوم:

السيناريو الاول:

توفر منصة التجارة الالكترونية مواقع عبر الانترنت (عبارة عن متاجر الالكترونية) خدمة مصادر الربح الخاصة بالمتاجر المستضاف على المنصة، حيث يستطيع المهاجم من خلال عرض مصدر الصفحة معرفة API الذي قام بجلب تلك المعلومات ومعرفة مصدرها على سبيل المثال : `shops/{shopName}/revenue_data.json/` ومن خلال تلك الطريقة يستطيع المهاجم من الحصول على بيانات الربح لجميع المتاجر المتسضافة في المنصة من خلال تغير {shopName} في عنوان URL بطريقة غير مصرح بها.

السيناريو الثاني :

اثناء فحص حركة مرور البيانات من قبل المهاجم، قام بإرسال طلب من نوع PATCH من خلال بروتوكول HTTP لاختبار وفحص جميع الردود من قبل الخادم، وبعد عمليات متعددة قام المهاجم بإرسال طلب من نوع PATCH وهو احد الطلبات المتعارف عليها في برتوكول HTTP. تتضمن الترويسة الافتراضية التي يستخدمها الطلب هي header X-User-Id: 54796 مما لفت انتباه المهاجم الى تغيرها لي header X-User-Id: 54795 مما سمح للمهاجم بالوصول/و التعديل الغير مصرح به لبيانات مستخدمين اخرين.

API1:2019 خلل التفويض والصلاحيات

كيف أمنع هذه الثغرة؟

- الاعتماد على سياسة و آلية تخويل لصلاحيات تعتمد على سياسة الاستخدام المقبول والتسلسل الهرمي السهل الواضح.
- استخدام آلية لتحقق من صلاحيات المستخدم الذي قام بتسجيل الدخول وهل لديه الحق في تنفيذ الإجراءات على السجلات في كل سجل على حدة وبشكل مستقل.
- يفضل استخدام قيم عشوائية وغير قابلة لتخمين في استخدام GUIDs في السجلات
- يفضل كتابة معايير لاختبار مدى نضج التفويض والصلاحيات وفي حال وجود أي ثغرة يفضل عدم استخدمها حتى تتخطى الاختبارات والمعايير المتفق عليها.

المراجع :

مصادر الخارجية :

- CWE-284: Improper Access Control
- CWE-285: Improper Authorization
- CWE-639: Authorization Bypass Through User-Controlled Key

API2:2019 خلل في صلاحيات المستخدم

خصائص API	قابلية الاستغلال: 3	الانتشار : 2	قابلية الاكتشاف : 2	التأثر التقني و تأثر الاعمال: 3	خصائص API
	المصادقة في واجهات برمجة التطبيقات API هي آلية معقدة وصعبة الفهم وقد يكون لدى مهندسي البرمجيات ومهندس امن المعلومات بعض المفاهيم الخاطئة حول حدود المصادقة وكيفية تنفيذها بشكل صحيح. بالإضافة إلى ذلك، تعد آلية المصادقة هدفًا سهلاً للمهاجمين ، نظرًا لأنها متاحة للجميع. تجعل هاتان النقطتان مكون المصادقة عرضة للعديد من عمليات الاستغلال.		هناك مسألتان فرعيتان: 1. محدودية آليات الحماية: يجب التعامل مع مصادر البيانات الخاصة بواجهات برمجة التطبيقات API والمسؤولة عن المصادقة بشكل مختلف عن المصادر الاخرى وتأمين طبقات إضافية من الحماية 2. سوء تنفيذ الآلية: يتم استخدام / تنفيذ الآلية دون مراعاة طرق الاستغلال الهجوم، أو أنها تبني بشكل غير صحيح (على سبيل المثال، قد لا تتناسب آلية المصادقة المصممة لأجهزة إنترنت الأشياء مع تطبيقات الويب)		يمكن للمهاجمين التحكم في حسابات المستخدمين الآخرين في النظام ، وقراءة بياناتهم الشخصية ، وتنفيذ إجراءات حساسة نيابة عنهم ، مثل المعاملات المالية وإرسال الرسائل الشخصية.	

هل أنا معرض لهذه الثغرة؟

مصادر البيانات وآلية عملها والاصول الخاصة بها تحتاج إلى الحماية. حيث يجب معاملة "نسيت كلمة المرور / إعادة تعيين كلمة المرور" بنفس طريقة آليات المصادقة.

يكون API معرض للخطر اذا كان:

- اذا كان لدى المهاجم قائمة متكاملة من اسماء المستخدمين وكلمات المرور تم الحصول عليها من اختراق او تسريب سابق
- عند قيام المهاجم بهجمات كسر كلمة المرور وعدم استخدام آلية تحقق اخرى من المستخدم مثل Captcha.
- كلمات المرور الضعيفة
- ارسال المعلومات الحساسة او كلمات المرور من خلال URL.
- عدم التحقق بالشكل الصحيح من عمليات المصادقة
- الموافقة على استخدام المصادقة الغير موقعه او الموقع بشكل غير امن (`"alg":"none"`) او عدم التحقق من تاريخ انتهاء المصادقة.
- استخدام البيانات غير المشفرة في عمليات تسجيل الدخول او عدم حفظ الارقام السرية بشكل مشفر
- استخدام مفاتيح تشفير ضعيفة.

امثلة على سيناريوهات الهجوم :

السيناريو الاول :

في حال قام المهاجم باستخدام بمحاولة الدخول بحسابات متعددة والتي تم الحصول عليها من تسريب للبيانات والتي يجب ان نقوم بوضع آلية للحماية من هجمات الدخول المتعدد بحسابات صحيح في وقت قصير ومحدود

السيناريو الثاني :

في حال قام المهاجم بمحاولة استعاد كلمة المرور من خلال ارسال طلب `POST` الى `` `api/system/verification-codes/` `` وذلك باستخدام اسم المستخدم فقط لتحقق من استعادة كلمة المرور. حيث يقوم التطبيق بإرسال رسالة نصية لهاتف الضحية مع آلية المصادقة الجديدة والمكونة من 6 ارقام. وحيث ان API لم يقم بوضع حد اعلى لطلبات المصادقة سيقوم المهاجم بتنفيذ جميع الاحتماليات وذلك بالتخمين على آلية المصادقة التي تم ارسالها الى هاتف الضحية وذلك بإرسال طلبات متعددة الى `` `api/system/verification-codes/{smsToken}` `` لتحقق من مصدر البيانات في حال كان احد عمليات التخمين كانت صحيحة.

API2:2019 خلل في صلاحيات المستخدم

كيف أمنع هذه الثغرة؟

▪ يجب ان تكون على دراية بجميع طرق و آليات المصادقة التي تتم من خلال (الهواتف /تطبيقات الويب /المصادقة الواحدة/إلخ)

▪ قم بالتعاون مع مهندس التطبيقات لمعرفة ماهي الآليات المفقودة عند عمليات المصادقة

▪ اقرأ عن آليات المصادقة الخاصة بك. تأكد من أنك تفهم ماذا وكيف يتم استخدامها ويجب التنويه على ان برتوكول OAuth ليس للمصادقة ، ولا مفاتيح واجهة برمجة التطبيقات API تستخدم للمصادقة.

▪ لا تقم بختراع واعادة صناعة آليات مصادقة جديدة بل اتبع افضل الامتثالات والمعايير المتعارف عليها.

▪ يجب التعامل مع مصادر البيانات لاستعادة كلمة المرور ونسيت كلمة المرور بشكل صحيح وذلك من خلال وضع ضوابط و آليات للحد من هجمات كسر كلمات المرور والاستفادة من وسائل الحماية كتعطيل الحساب بعد عدد محاولات غير ناجحة من عمليات تسجيل الدخول.

▪ قم باستخدام نموذج OWASP Authentication Cheatsheet

▪ في حال توفر التحقق الثنائي قم باستخدامه.

▪ قم بتنصيب التقنيات والطرق والاليات لرصد هجمات كسر كلمات المرور او محاولة استغلال الحسابات المسربة وقم بوضع آلية محددة لتقليل معدل المصادقة المستخدمة على API.

▪ قم باستخدام آلية ايقاف الحسابات او Captcha وذلك لتقليل ومنع هجمات كسر كلمات المرور وقم بتنصيب تقنية عدم اتاحة استخدام كلمات المرور الضعيفة.

▪ لا ينبغي استخدام API كوسيلة للمصادقة للمستخدم بل يستخدم على سبيل المثال لتطبيقات والمشاريع.

المراجع :

- OWASP Key Management Cheat Sheet
- OWASP Authentication Cheatsheet
- Credential Stuffing

مصادر الخارجية :

- CWE-798: Use of Hard-coded Credentials

API3:2019 خلل في استعراض البيانات

خصائص API	قابلية الاستغلال: 3	الانتشار : 2	قابلية الاكتشاف : 2	التأثر التقني و تأثر الاعمال: 2	خصائص API
عادة ما يكون الكشف الغير مصرح به عن المعلومات او البيانات من خلال مراقبة حركة مرور البيانات او الطلبات وتحليل جميع الردود القادمة من API، وذلك للبحث عن أي بيانات حساسة يتم استعادتها وغير مصرح للمستخدم بالاطلاع عليها.	تعتمد واجهات التطبيقات API على ان تكون عوامل التصفية من جانب المستخدم حيث ان API عادة ما يتم استخدامه كمصدر للبيانات وفي بعض الأحيان يقوم المطورون بتنصيب API بشكل عام وافتراضي من غير التفكير في طرق التعامل مع البيانات الحساسة. حيث ان أدوات الفحص واكتشاف الثغرات الأمنية تستطيع رصد مثل تلك الثغرات والتي يصعب على API معرفة إذا كان هذا الطلب لأغراض الاستخدام الصحيح والقانوني او لأغراض الاطلاع وتسريب البيانات الحساسة. لذلك يجب ان نقوم بتصنيف البيانات الحساسة والفهم العميق لألية الطلب لها.			عادة ما يودي الكشف عن البيانات الى الاطلاع غير المصرح به او تسريب البيانات	

هل أنا معرض لهذه الثغرة؟

تقوم واجهة برمجة التطبيقات بإرجاع البيانات الحساسة إلى العميل حسب التصميم والطلب . عادة ما يتم تصفية هذه البيانات من جانب العميل قبل تقديمها للمستخدم. يمكن للمهاجم بسهولة اعتراض حركة المرور ورؤية البيانات الحساسة.

امثلة على سيناريوهات الهجوم:

السيناريو الاول:

يقوم مطورين تطبيق الهواتف الذكية باستخدام `api/articles/{articleId}/comments/{commentId}/` كمصدر للبيانات وذلك بهدف عرض المقالات وبعض البيانات الوصفية الخاصة بها. وهنا يقوم المهاجم باعتراض حركة مرور البيانات الصادرة من هذه التطبيق وقراءة تلك البيانات الوصفية والتي قد تقوم بتسريب بعض البيانات الحساسة مثل بيانات كاتبين التعليقات وبعض بيانات تحديد الشخصية ك PII، حيث ان مصدر البيانات تم تنصيبه بشكل افتراضي على هيئة (JSON) ومبنية على عامل التصفية لدى المستخدم.

السيناريو الثاني :

يسمح نظام المراقبة المبني على أنظمة IOT او انترنت الأشياء لمدير النظام بانشاء حسابات للمستخدمين بمختلف الصلاحيات، حيث قام مدير النظام بانشاء حساب لاحد حراس الامن والذي مصرح له بالاطلاع على بعض المباني و المواقع. وعندما قام الحارس باستخدام هاتفه للاطلاع على النظام يقوم نظام API باستدعاء لوحة أنظمة المراقبة المتاحة له من خلال `api/sites/111/cameras/` والتي تسمح له بمعرفة عدد الكاميرات المتاحة الاطلاع عليها من قبل حارس الامن حيث ان بعد عملية الطلب تم استقبال الرد من الخادم ببعض المعلومات التفصيلية على سبيل المثال `{"id":"xxx","live_access_token":"xxxx-bbbbb","building_id":"yyy"}` والتي لا تظهر على لوحة المراقبة الخاصة بالحارس (الواجهة الرسومية) بل في تفاصيل الطلب فقط والتي تحتوي على جميع الكاميرات والمباني.

API3:2019 خلل في استعراض البيانات

كيف أمنع هذه الثغرة؟

- لا تثق ابداً في عوامل التصفية لدى العميل او المستخدم في حال كانت هناك بيانات حساسة
- دائماً قم بمراجعة الطلبات والردود من مصادر البيانات للتاكد من ان جميع البيانات المتوفرة هي بيانات غير حساسة ومنطقية
- يجب على مهندسي التطبيقات الداخلية و مسؤولي الانظمة السؤال بشكل دائم من هم مستخدمي تلك البيانات قبل البدء بتنصيب API جديدة على النظام.
- تجنب استخدام الإعدادات العامة مثل to_json() و To_string() واستبدلها بخصائص معينة ومحددة مطلوب استرجاعها.
- قم بتصنيف المعلومات الحساسة و المعلومات المرتبطة بالهوية الشخصية (PII) التي يخزنها تطبيقك ويعمل معها ، مع مراجعة جميع الطلبات الخاصة بواجهة برمجة التطبيقاتAPI والردود المتوقعة منها ومعرفة الاشكاليات الامنية التي قد يتم رصدها بتلك الردود
- قم باستخدام آليات التحقق مثل (schema-based response validation mechanism) وحدد ماهي البيانات التي يتم ارجاعها مع الطلبات بما في ذلك الاخطاء والمعلومات المتوفرة بها.

المراجع :

مصادر خارجية :

- CWE-213: Intentional Information Exposure

API4:2019 ضعف في البنية التحتية و حد محاولات الطلبات

خصائص API	قابلية الاستغلال: 2	الانتشار : 3	قابلية الاكتشاف : 3	التأثر التقني و تأثر الاعمال: 2	خصائص API
	ان عملية الاختراق في بعض الاحيان عملية غير معقدة حيث لا يستلزم الا طلب بسيط للAPI ومن غير عملية مصادقة كذلك وقد يتم ارسال طلب من خلال جهاز واحد او أجهزة متعددة او أجهزة الخدمات السحابية.		من الاخطاء الشائعة والمنتشرة هو عدم وضع معدل لطلبات او لم يتم اختيار الموصفات المناسبة عند تنصيب واجهات برمجة التطبيقات API.		قد يؤدي الاستغلال الى هجمات حجب الخدمة DOS مما يجعل واجهة برمجة التطبيقات API غير مستجيبة او خارج الخدمة.

هل أنا معرض لهذه الثغرة؟

تستهلك واجهة برمجة التطبيقات API المصادر والأصول من شبكات ووحدات المعالجة وكذلك وسائط التخزين حيث يعتمد بشكل كبير مقدرة تعامل البنية التحتية حسب طلبات ومدخلات المستخدم لمصادر البيانات. وضع في الاعتبار ان طلبات واجهة برمجة التطبيقات API التي تفوق قدرات البنية التحتية تعرضها للخطر بشكل كبير اذا لم يتم تداركها و وضع معدل لمستوى ومحتوى تلك الطلبات ومنها:

مدة حياة الطلب

- اعلى حد من استخدام الذاكرة العشوائية لكل طلب
- عدد الملفات وطرق وصفها وحفظها وعرضها
- عدد العمليات
- عدد وحجم البيانات عند رفعها
- عدد الطلبات لكل مستخدم
- عدد الصفحات التي يتم عرضها في كل طلب و استجابة لصفحة الواحدة.

امثلة على سيناريوهات الهجوم :

السيناريو الاول :

يقوم المهاجم برفع صورة كبيرة الحجم والابعاد عن طريق طلب POST الى `api/v1/images/` وعند اكتمال عملية الرفع يقوم الخادم باستعراض الصور المتبقية على هيئة ايقونات مصغرة بسبب الابعاد والحجم الذي قد يستغرق الموارد وقد يؤدي الى عدم واجهة برمجة التطبيقات API.

السيناريو الثاني :

يقوم التطبيق بعرض المستخدمين بحد اقصى 100 مستخدم في كل صفحة من خلال ارسال طلب الى `api/users?/page=1&size=100`، مما قد يمكن المهاجم من تغير القيمة الى 200000 في عدد أسماء المستخدمين المعروضة في صفحة واحد مما يسبب في حدوث مشكلات في أداة قاعدة البيانات وفي الوقت نفسة تصبح واجهة برمجة التطبيقات غير متاحة وغير قادرة على التعامل مع الطلبات الأخرى (هجمة حجب الخدمة DOS) ويمكن استخدام نفس السيناريو لاستعراض الأخطاء او لاستغلال بعض عمليات Integer Overflow او Buffer Overflow.

API4:2019 ضعف في البنية التحتية و حد محاولات الطلبات

كيف أمنع هذه الثغرة؟

- يجعل منصة Docker الامر في غاية البساطة في التحكم في الذاكرة العشوائية او وحدات المعالجة و التخزين
- ضع معدل محدد لعدد الطلبات التي يقوم بطلبها المستخدم خلال اطار زمني معين
- اخطار المستخدم عند تجازو المعدل المحدد في الاطار الزمني المعين
- قم باضافة بعض آليات التحقق من جانب الخادم في عمليات الطلبات او حتى التحقق من النصوص او العمليات او الطلبات وتحديداً في تلك العمليات التي تتطلب عدد من السجلات يتم استرجاعها من العميل.
- تحديد وفرض الحد الاعلى لحجم وابعاد الطلبات المرفوعة مثل الحد الاقصى لعدد الجمل او الحد الاعلى لعدد الاسطر

المراجع :

- Docker Cheat Sheet - Limit resources (memory, CPU, file descriptors, processes, restarts
- Blocking Brute Force Attacks
- REST Assessment Cheat Sheet

مصادر خارجية :

- CWE-307: Improper Restriction of Excessive Authentication Attempts
- CWE-770: Allocation of Resources Without Limits or Throttling
- "*Rate Limiting (Throttling)*" - Security Strategies for Microservices-based Application Systems

API5:2019 ضعف في التحقق من الهوية وادارة التفويض والصلاحيات

خصائص API	قابلية الاستغلال: 3	الانتشار : 2	قابلية الاكتشاف :1	التأثر التقني و تأثر الاعمال: 3	خصائص API
	ان عملية استغلال هذه الثغرة بسيط نسبياً بحيث يستطيع المهاجم ارسال طلب غير ضار من خلال واجهة برمجة التطبيقات API الى مصادر البيانات التي من الغير مصرح له بالاطلاع عليها. وقد تكون تلك المصادر متاحة للمستخدمين المجهولين او المستخدم الذي لا يملك صلاحيات عالية. وحيث ان من السهل اكتشاف مثل تلك الثغرات من خلال معرفة سلوك الطلبات والياتها المستخدمة وطرق طلبها والردود المتوقعة من كل طلب وقد يتم استغلالها ببساطة من خلال استبدال طلب GET بـ PUT او تغير صلاحيات المستخدم من " User " الى "Admin".		عادة ما يتم تحديد صلاحيات الوصول للموارد من خلال الاعدادات. وفي بعض الأحيان على مستوى الاكواد البرمجية، ان عملية تنصيب طرق التحقق بالشكل الصحيح قد تكون في بعض الأحيان عملية معقدة. حيث ان معظم التطبيقات الحديثة تحتوي على مختلف الصلاحيات والمجموعات بتسلسل هرمي معقد بعض الشئ.		بعض آليات العمل قد تسمح للمهاجم في الاستفادة والوصول والاطلاع الغير مصرح به، او حصوله على صلاحيات إدارية تمكنه من التحكم والسيطرة.

هل أنا معرض لهذه الثغرة؟

أفضل طريقة للعثور على مشكلات وخلل تفويض مستوى الصلاحيات والمصادقة هي إجراء تحليل عميق لآلية التفويض ، مع مراعاة التسلسل الهرمي للمستخدم ، والأدوار أو المجموعات المختلفة في التطبيق ، وطرح الأسئلة التالية:

- هل يستطيع المستخدم العادي الوصول الى مصادر صلاحيات المدراء ؟
- هل يستطيع المستخدم تعديل او تعيين او مسح مصادر البيانات عند تغير طريقة الطلب للبروتوكول على سبيل المثال من GET الى DELETE ؟
- هل يستطيع المستخدم في مجموعة أ من الوصول الى مصادر المجموعة ب من خلال تخمين مصدر تلك المجموعة / api/v1/users/export_all

لا تقم بوضع وتقسيم الصلاحيات ما بين الصلاحيات المعتادة والصلاحيات الادارية من خلال مسار URL.

و من الشائع لدى المطورين عرض مصادر البيانات الإدارية ضمن مسار محدد مثل API/Admin ومن الشائع كذلك استخدام مصادر واحدة للمستخدم العادي وكذلك للمدراء مثل api/users.

امثلة على سيناريوهات الهجوم :

السيناريو الاول :

يقوم التطبيق فقط بالسماح للمستخدمين المدعوين بالتسجيل، حيث يقوم التطبيق بطلب API الخاص من خلال طلب GET على سبيل المثال المسار التالي " /api/invites/{invite_guid}" ويأتي الرد من الخادم والذي يحتوي على ملف JSON مع تفاصيل الدعوة، وكذلك تفاصيل المستخدمين و الصلاحيات والبريد الالكتروني.

يقوم المهاجم بتكرار الطلبات ومحاولة التلاعب والتعديل في طريقة الطلب من مصدر البيانات من GET الى POST مع المسار التالي " / api/invites/new" حيث ان هذا المسار مسموح بالوصول له فقط لأصحاب الصلاحيات الإدارية بواسطة صفحة الإدارة والتي من الوضح عدم تطبيق مستوى المصادقة والتفويض على مستوى الصلاحية.

المهاجم قام باستغلال الخطأ من خلال ارسال طلب دعوة لنفسة ومن ثم قام بإنشاء حساب بصلاحيات مرتفعة.

```
POST /api/invites/new
```

```
{“email”:”hugo@malicious.com”,”role”:”admin”}
```

السيناريو الثاني :

تحتوي واجهة برمجة التطبيقات API على صلاحيات وصول الى مصادر البيانات والمحددة فقط لمدراء النظام من خلال الطلب باستخدام GET للمسار التالي api/admin/v1/users/all/ حيث ان مصدر البيانات عند ارجاع البيانات لا بتأكد من صلاحيات من قام بطلبها او الصلاحيات المخولة له مما يمكن المهاجم من تخمين المسارات الخاصة بمصادر البيانات لاستعراض بيانات حساسة غير مصرح له بالوصول لها.

كيف أمنع هذه الثغرة؟

يجب أن يحتوي التطبيق الخاص بك على وحدة تفويض متسقة وسهلة التحليل يتم استدعاؤها وظائف تطبيقك. في كثير من الأحيان يتم توفير هذه الحماية بواسطة مكون أو أكثر خارج الاكواد البرمجية الخاصة بالتطبيق.

- يجب منع الوصول لجميع المصادر بشكل افتراضي وبعد ذلك يتم السماح والاستثناء للمصادر لكل مصدر على حدة ولكل صلاحية بشكل مستقل.
- قم بمراجعة صلاحيات المصادقة والتفويض الخاص بالآليات العمل، مع مراعاة منطق التسلسل الهرمي وصلاحيات المجموعات ولصلاحيات على مستوى المستخدمين.
- التأكد من ان صلاحيات التحكم الادارية مبنية بشكل سليم ومرتبطة بصلاحيات المصادقة والتفويض لكل مجموعة او مستخدم او صلاحية.
- التأكد من ان الاوامر والصلاحيات الادارية مبنية بشكل محوكم وهناك وحدة تحكم تقوم بفحص الصلاحيات والتفويض لكل مستخدم بناء على المجموعة التي تم تعينه بداخلها.

المراجع :

- OWASP Article on Forced Browsing
- OWASP Top 10 2013-A7-Missing Function Level Access Control
- OWASP Development Guide: Chapter on Authorization

مصادر خارجية :

- CWE-285: Improper Authorization

API6:2019 خلل في التعيين او التعديل

خصائص API	قابلية الاستغلال: 2	الانتشار : 2	قابلية الاكتشاف : 2	التأثر التقني و تأثر الاعمال: 2	خصائص API
	يتطلب الاستغلال عادةً فهم منطق آلية العمل وعلاقة الكائنات ببعضها وهيكل واجهة برمجة التطبيقات API حيث يعد استغلال الخلل في التعين او التعديل أسهل في واجهات برمجة التطبيقاتAPI ، حيث انها في بعض الاحيان عند عرض بعض الخصائص الخاصة بـ API يقوم كذلك بعرض الإعدادات والخواص الخاصة بها.		تشجع الاطر الحديثة في البرمجة المطورين على استخدام الطرق الأتوماتيكية التي تسمح للمستخدم بإدخال المتغيرات داخل الكائن. وهذا يسمح للمهاجمين باستخدامها لتحديث بعض المعلومات الحساسة او الكتابة فوق تلك الخصائص او الكائنات التي قام المطورين بإخفائها		قد يؤدي هذا الاستغلال الى تصعيد الصلاحيات والتلاعب بالامتيازات وتجاوز آليات الامان وغير ذلك

هل أنا معرض لهذه الثغرة؟

تحتوي بعض التطبيقات الحديثة على العديد من الخصائص وبعض تلك الخصائص يجب تحديثها بواسطة المستخدمين على سبيل المثال `user.first_name` او `user.address` وبعض الخصائص لا يسمح للمستخدمين بتعديلها على سبيل المثال `user.is_vip`.

تكون واجهة برمجة التطبيقات API ومصادر البيانات عرضة للاختراق اذا تم استخدام مدخلات المستخدم ككائنات داخلية، من دون مراعات لمستوى حساسية وخطورة تلك الكائنات. وها قد يسمح للمهاجم بتحديث خصائص الكائنات التي لا يجب او غير مصرح له بالوصول اليها.

امثلة على بعض الخصائص ذات الحساسية:

- التعديل في بعض الخواص: مثل `user.is_admin, user.is_vip` يجب ان تكون فقط لاصحاب الصلاحيات الإدارية.
- الخواص المعتدة على العمليات: مثل `user.cash` يجب ان يتم التحقق داخلياً بعد التأكد من عملية الدفع.
- الخواص الداخلية: على سبيل المثال `article.created_time` يجب ان يكون داخلياً وبواسطة التطبيق فقط.

امثلة على سيناريوهات الهجوم :

السيناريو الاول :

تطبيق مخصص لرحلات يوفر للمستخدم خيار تعديل البيانات والمعلومات الأساسية للملف الشخصي من خلال ارسال طلب بواسطة برمجة واجهة التطبيقات api التالي `api/v1/users/me/` بواسطة طلب PUT باستخدامJSON بالشكل التالي:

```
{"user_name":"inons","age":24}
```

يتضمن الطلب GET للمسار التالي `api/v1/users/me/` مع خاصية معرفة الرصيد الائتمانية:

```
{"user_name":"inons","age":24,"credit_balance":10}.
```

حيث قام المهاجم باعتراض الطلب وتغيره الى التالي:

```
{"user_name":"attacker","age":60,"credit_balance":99999}
```

ونظراً لان مصادر البيانات مصابة بخلل في التعيين والتعديل قام المهاجم بالحصول على مبالغ مالية من دون دفع أي مبلغ حقيقي.

API6:2019 خلل في التعيين او التعديل

السيناريو الثاني :

تتيح منصة مشاركة ملفات الفيديو تحميل ورفع وتنزيل الملفات بتنسيقات وامتدادات مختلفة. حيث لاحظ المهاجم ان واجهة برمجة التطبيقات والتي تستطيع الوصول لها من خلال طلب GET على المسار التالي api/v1/videos/{video_id}/meta_data/ انه يستطيع الحصول على ملف JSON يحتوي على خصائص ملفات الفيديو. على سبيل المثال "mp4_conversion_params":"-v codec h264" مما يوضح ان التطبيق يستخدم أوامر Shell لعملية تحويل الفيديو.
وجد المهاجم احد مصادر البيانات مصابة بالثغرة التي تسمح له بالتعديل والتعين فقام بإرسال تعليمات برمجية ضارة باستخدام واجهة برمجة التطبيقات API مع طلب POST من خلال المسار التالي api/v1/videos/new/ حيث قام بتعين القيمة التالية مع العملية "
"/:mp4_conversion_params":"-v codec h264 && format C والتي سمحت للمهاجم بتنفيذ التعليمات من خلال أوامر Shell بعد ارساله لطلب تنزيل ملف الفيديو.

كيف أمنع هذه الثغرة؟

- تجنب بقدر ما يمكن استخدام الوظائف التي تتطلب من المستخدم ادخل بعض المتغيرات في الاكواد الداخلية.
- أضف الخصائص التي يتوجب على المستخدم إدخالها الى قائمة بيضاء محددة.
- استخدام الطرق والاساليب التي تمنع المستخدم من الاطلاع او الوصول غير المصرح به الى المصادر او الخصائص.
- إذا كان من الممكن فرض سياسة استخدام مدخلات محددة في البيانات عند عمليات الرفع او التنزيل.

المراجع :

مصادر خارجية :

- CWE-915: Improperly Controlled Modification of Dynamically-Determined Object Attributes

API7:2019 الاعداد الخاطئ

خصائص API	قابلية الاستغلال: 3	الانتشار : 3	قابلية الاكتشاف : 3	التأثر التقني و تأثر الاعمال: 2	خصائص API
	يحاول المهاجمون غالباً البحث عن الثغرات الأمنية على مستوى الأنظمة او اليات العمل اول على مصادر بيانات غير محمية وذلك بغاية الوصول الغير مصرح به للمعلومات.	يمكن ان يحدث الاعداد الخاطئ في أي مستوى من مستويات واجهة برمجة التطبيقات API، ابتداءًا من مستوى الشبكة الى مستوى التطبيقات، حيث تتوفر الأدوات للقيام بالفحص واكتشاف الأخطاء بشكل آلي وذلك بهدف البحث عن مواطن الإعدادات الخاطئة او الخدمات الفعالة والغير ضرورية او الخيارات القديمة والمصابة بثغرات.		قد يؤدي عملية الإعدادات الخاطئة الى تسريب البيانات وكذلك اختراق الأنظمة والخوادم.	

هل أنا معرض لهذه الثغرة؟

قد يكون واجهة التطبيقات API معرضة لثغرات في حال :

- اذا لم يكن هناك أي آلية متبعة لعملية تعزيز حماية النظام في جميع مراحله او اذا كان هناك تهيئة غير صحيحة على الخدمات السحابية.
- اذا لم يكن هناك آلية لسد الثغرات الأمنية او في حال كانت الأنظمة المستخدمة غير محدثة او خارجة عن الخدمة.
- اذا كان هناك تفعيل لبعض الطلبات الغير مطلوبة مثل بعض طلبات HTTP الغير مستخدمة TREAC او DELETE على سبيل المثال.
- اذا لم يتم استخدام التشفير بواسطة TLS.
- إذا لم يتم تعين سياسة مشاركة المواد بطريقة صحيحة او كان هناك خطا في الإعدادات الخاصة بها
- إذا كانت رسائل الخطأ تحتوي على معلومات حساسة ويمكن تتبعها.

امثلة على سيناريوهات الهجوم :

السيناريو الاول :

يعثر المهاجم على ملف .bash_history في احد المسارات الرئيسية في الخادم والذي يحتوي على الأوامر التي يستخدمها المطورين في الوصول الى واجهة برمجية التطبيقات API.

```
$ curl -X GET 'https://api.server/endpoint/' -H 'authorization: Basic Zm9vOmJhcg=='
```

يمكن للمهاجم ايضاً معرفة مصادر البيانات من خلال الأوامر التي يستخدمها المطورين من خلال تكرار عملية الوصول للملف أعلاه وما حدث ذلك الا بسبب عد توثيق الإجراءات بالشكل الصحيح.

السيناريو الثاني :

يقوم المهاجمون في معظم الأحيان في استخدام محركات البحث بهدف الحصول على خوادم يستطيع من خلالها الوصول الى مصدر البيانات بشكل مباشر. او من خلال البحث عن أحد المنافذ المشهورة في قواعد البيانات او في إدارة الأنظمة والخوادم. وفي حال كان الخادم او النظام المستهدف يقوم باستخدام الأعدادت الافتراضية وغير محمي باستخدام مصادقة صحيحة قد يمكن المهاجم من الوصول للبيانات الشخصية PII والذي قد يؤدي الى تسريب بيانات المستخدمين لتلك الخدمة.

السيناريو الثالث :

عند اعتراض حركة المرور للبيانات الخاصة بأحد تطبيقات الهواتف المحمولة والتي تستخدم بروتوكول TLS في حركة البيانات ولكن لا تعتمد على التشفير باستخدام TLS عند استخدام واجهة برمجة التطبيقات API وبعد البحث من قبل المهاجم استطاع معرفة ان عملية تحميل ورفع الصور يتم بشكل غير مشفر، فقد وجد المهاجم نمط وطريقة لمعرفة الاستجابة الواردة من قبل الخادم او من قبل مصدر البيانات والتي قد تمكنه بطريقة او بأخرى من تتبع تفضيلات المستخدمين عند تنزيل او عرض تلك الصور.

API7:2019 الاعداد الخاطئ

كيف أمنع هذه الثغرة؟

دورة حياة واجهة برمجة التطبيقات API لابد ان تشتمل على :

- عملية تعزيز حماية الأنظمة تساهم بشكل كبير في بناء بيئة امنة و موثوقة
- إيجاد آلية لمراجعة الإعدادات و التحديثات بأكملها ويجب ان تتضمن مراجعة كل من ملفات الحفظ و المزامنة مكونات واجهة برمجة التطبيقات API التطبيقات و الخدمات السحابية.
- توفير اتصال امن و مشفر لجميع الاتصالات في التعامل مع التطبيق او رفع وتحميل الصور.
- عملية تقييم امني مستمر لمعرفة مستوى نضج الاعدادات في جميع انحاء البنية التحتية.

علاوة على ذلك:

- لمنع تتبع الأخطاء التي قد يتم الرد بها بعد عمليات الطلب والتي قد تمكن المهاجم من استعراض البيانات الحساسة يجب ان تكون جميع الرود محدودة ومحصورة بما في ذلك عمليات الاستجابة للأخطاء.
- تأكد انه لا يمكن الوصول الى واجهة برمجة التطبيقات API الا من خلال احد الطلبات المحددة وعد السماح بجميع الطلبات الخاصة ببروتوكول HTTP بالعمل بل ويجب تعطيلها مثال (HEAD , TRACE).
- يجب على واجهات برمجة التطبيقات API التي تتوقع أن يتم الوصول إليها من عملاء يستندون إلى المتصفح على سبيل المثال (الواجهة الامامية لخدمات الويب) يجب تنفيذ سياسة سليمة وموثوقة لمشاركة الموارد عبر (CORS).

المراجع :

- OWASP Secure Headers Project
- OWASP Testing Guide: Configuration Management
- OWASP Testing Guide: Testing for Error Codes
- OWASP Testing Guide: Test Cross Origin Resource Sharing

مصادر خارجية :

- CWE-2: Environmental Security Flaws
- CWE-16: Configuration
- CWE-388: Error Handling
- Guide to General Server Security, NIST
- Let’s Encrypt: a free, automated, and open Certificate Authority

API8:2019 الحقن

خصائص API	قابلية الاستغلال: 3	الانتشار : 2	قابلية الاكتشاف : 3	التأثر التقني و تأثر الاعمال: 3	خصائص API
	يقوم المهاجمون بإرسال بيانات ضارة من خلال واجهة برمجة التطبيقات API وذلك من خلال حقنها بأي طريقة ممكنة (مثل الإدخال المباشر و التعليمات البرمجية و طرق ربط الخدمات ...الخ) على أمل إرسالها إلى المفسر.	تعتبر ثغرات الحقن منشرة وشائعة في استعلامات SQL و LDAP و NoSQL وأوامر أنظمة التشغيل و XML parsers و ORM. واكتشاف هذه الثغرات يعتبر سهل عن مراجعة الشفرة المصدرية. بإمكان المهاجمين استخدام أدوات الفحص.		يمكن أن يؤدي الحقن إلى إفشاء المعلومات أو مسح وفقدان البيانات. كما يمكن أن يودي إلى حجب الخدمة DoS، أو اختراق كامل للنظام.	

هل أنا معرض لهذه الثغرة؟

قد تكون واجهة برمجة التطبيقات API معرضة للاستغلال بمثل هذه الهجمات عندما :

- لا يتم تصفية البيانات أو التحقق منها في حال كانت مقدمة من المستخدمين من طريق واجهة برمجة التطبيقات.
- يتم استخدام البيانات بشكل مباشر مع SQL/NoSQL/LDAP queries, OS commands, XML parsers.
- لا يتم التحقق من صحة البيانات الواردة من أنظمة خارجية مثل (الأنظمة المرتبطة بالخادم) أو تصفيتها أو التحقق منها من قبل واجهة برمجة التطبيقات API قبل عملية استخدامها.

امثلة على سيناريوهات الهجوم :

السيناريو الاول :

يقوم نظام جهاز التحكم الأبوي باستخدام المسار api/CONFIG/restore/ والذي يتوقع أن يستقبل معرف التطبيق appId في أجزاء متعددة. فباستخدام برنامج فك وتحويل الشفرات البرمجية(decompile)، يجد المهاجم أن المعرف appId يتم تمريره مباشرة للنظام ومن غير عوامل التصفية المقترحة:

```
snprintf(cmd, 128, "%srestore_backup.sh /tmp/postfile.bin %s %d",
     "/mnt/shares/usr/bin/scripts/", appid, 66);
system(cmd);
```

يسمح الأمر التالي للمهاجم بإغلاق أي جهاز مصاب بتلك الثغرة البرمجية

```
$ curl -k "https://${deviceIP}:4567/api/CONFIG/restore" -F 'appid=$(/etc/pod/power_down.sh)'
```

السيناريو الثاني :

لدينا تطبيق قائم على وظائف CRUD للتعامل مع الحجوزات، تمكن مهاجم من التعرف على إمكانية حقن NoSQL من خلال الاستعلام بالمعرف الفريد للحجوزات bookingId وطلب الحذف بأمر كالتالي: DELETE /api/bookings?bookingId=678

خادم واجهة برمجة التطبيقات (API Server) يستخدم الدالة التالية للتعامل مع طلبات الحذف:

```
router.delete('/bookings', async function (req, res, next) {
  try {
    const deletedBooking = await Bookings.findOneAndRemove({_id' : req.query.bookingId});
    res.status(200);
  } catch (err) {
    res.status(400).json({
      error: 'Unexpected error occured while processing a request'
    });
  }
});
```

API8:2019 الحقن

قام المهاجم باعتراض الطلبات الخاصة بالمعرف الفريد bookingId وقام بتغير أمر الاستعلام كما هو معروض بالأسفل مما أدى إلى حذف حجز يعود لمستخدم آخر:

```
DELETE /api/bookings?bookingId[$ne]=678
```

كيف أمنع هذه الثغرة؟

لمنع عمليات الحقن انت بحاجة إلى فصل الأوامر والتعليمات البرمجية عن الاستعلامات بشكل صحيح و امن.

- قم بإجراء التحقق من صحة البيانات المدخلة باستخدام مكتبة موحدة وامنه وموثوقة ويتم صيانتها بشكل دوري.
- تحقق من صحة جميع البيانات المقدمة من المستخدم أو غيرها من البيانات الواردة من الأنظمة المتكاملة وتصفيتها.
- يجب التعامل مع الأحرف والرموز الخاصة باستخدام الصيغة المحددة للمفسر المستهدف.
- استخدم واجهة برمجة تطبيقات آمنة (safe API) ذات استعلامات واضحة.
- ضع حداً لعدد السجلات التي يتم إرجاعها لمنع تسريب البيانات بشكل كبير في حالة نجاح عملية الحقن.
- تحقق من صحة البيانات الواردة باستخدام عوامل تصفية كافية للسماح فقط بالقيم الصالحة لكل استعلام تم إدخاله.
- عرف بشكل واضح ومحدد الانماط و أنواع البيانات المستخدمة في الاستعلامات

المراجع :

- OWASP Injection Flaws
- SQL Injection
- NoSQL Injection Fun with Objects and Arrays
- Command Injection

مصادر خارجية :

- CWE-77: Command Injection
- CWE-89: SQL Injection

API9:2019 خلل في ادارة الاصول

خصائص API	قابلية الاستغلال: 3	الانتشار : 3	قابلية الاكتشاف : 2	التأثر التقني و تأثر الاعمال: 2	خصائص API
	ان استخدام واجهة برمجة التطبيقات Api القديمة و الغير محدثة هو اسهل طريقة تسبب اختراق الأنظمة لديك دون الحاجة والجهد الذي قد يبذلها المهاجم حتى وان كانت أدوات ومكونات الأمان تم ايجادها بشكل صحيح وسليم، ولكن جميع تلك الأدوات والمكونات وجدت للأنظمة الحديثة و المتطورة من واجهة برمجة التطبيقات API.	ان عمليات التوثيق الغير محدثة تجعل من الصعب تتبع واصلاح الثغرات. وكذلك عدم جرد الأصول التقنية يؤدي بشكل مباشر الى عدم ترقيع الأنظمة من الثغرات الأمنية، والذي قد يؤدي الى تسريب للبيانات. وكما انه من الشائع رصد واجهة برمجة التطبيقات API متاحة على الانترنت دون الحاجة لها بسبب الأنظمة والمفاهيم الحديثة في الخدمات المصفرة والتي تجعل من التطبيقات سهلة النشر ومستقلة على سبيل المثال (الحوسبة السحابية)		قد يتمكن المهاجمون من الوصول الى بيانات حساسة غير مصرح لهم بالوصول لها، او حتى اختراق الخادم من خلال استغلال احد الثغرات الغير مرقعه لخدمات واجهة برمجة التطبيقات API والتي تستخدم للاتصال بقاعدة البيانات.	

هل أنا معرض لهذه الثغرة؟

قد يكون واجهة برمجة التطبيقات معرض لمثل هذه الثغره في حالة :

- الغرض من استخدام واجهة برمجة التطبيقات غير واضح والذي قد يقود للأسئلة التالية:
  - ما هي البيئة التي تعمل فيها واجهة برمجة التطبيقات (على سبيل المثال ، الإنتاج ، التدريج ، الاختبار ، التطوير)؟
  - من المخول للوصول الى الشبكة الخاصة بواجهة برمجة التطبيقات (على سبيل المثال ، عام ، داخلي ، شركاء)؟
  - ما هو إصدار API المستخدم ؟
  - ماهي البيانات التي يتم جمعها بواسطة API؟ وهل هي بيانات شخصية؟
  - ماهي آلية وسير العمليات ؟
- لا توجد وثائق معتمدة او وثائق قديمة وغير محدثة.
- لا توجد خطة لإيقاف أي واجهة برمجة التطبيقات القديمة API
- لا توجد آلية لحصر الأصول او انها قديمة.
- لا توجد آلية لحصر الأصول المتصلة بالأنظمة سوء كانت طرف اول او طرف ثالث.
- إصدارات قديمة وغير محدثة ولا تزال مستخدمة

امثلة على سيناريوهات الهجوم :

السيناريو الاول :

بعد إعادة عملية تصميم التطبيقات، لم يتم الاهتمام بترقية الإصدار الخاص بواجهة برمجة التطبيقات API بل تم استخدام القديم وهو متوفر على المسار التالي api.someservice.com/v1. وهو المستخدم وغير محمي، مع إمكانية الوصول الى قاعدة البيانات بصلاحيات مستخدم. وبعد عمليات الفحص من قبل المهاجمين في التطبيقات المعاد تصميمها وهي على المسار التالي api.someservice.com/v2. بعد عملية تخمين بسيطة جداً تم تغير /v2 الى v1 في المسار للموقع والذي منح المهاجم إمكانية الوصول لواجهة برمجة التطبيقات القديمة والغير محدثة والتي أدت الى تسريب معلومات حساسة لأكثر من 100 مليون مستخدمة ومنه معلومات شخصية.

السيناريو الثاني :

تقوم منصات التواصل الاجتماعي باستخدام آلية مبتكرة لمنع هجمات كسر كلمات المرور من خلال تحديد معدل الطلبات وذلك بهدف تقليل محاولات الاختراق. ولكن آلية الأمان تلك لم يتم تطبيقها على الكود الخاص بواجهة برمجة التطبيقات API. بل قاموا بفصلها لكي تكون ما بين المستخدم و (www.socialnetwork.com) .API. و أحد الباحثين قام بإيجاد النطاق الخاص بـ API

API9:2019 خلل في ادارة الاصول

(ww.mbasic.beta.socialnetwork.com) والذي يستطيع من خلاله القيام بنفس المهام التي تقوم بها منصة التواصل الاجتماعي بما في ذلك إعادة تعين كلمات المرور من خلال استخدام أسلوب مصادقة يتم استخدامه من قبل المنصة وهو عبارة عن رمز مكون من 6 ارقام يتم إدخاله في حال طلبت استعادة كلمة المرور.

كيف أمنع هذه الثغرة؟

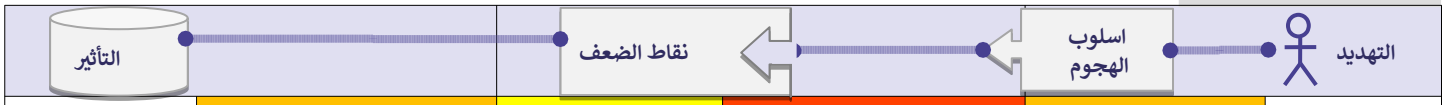
- جرد وحصر جميع المعرفات والأجهزة الخاصة بواجهة برمجة التطبيقات وتوثيقهم بمستند لكل كائن على حدة، والتركيز بشكل كبير على بيئة API (على سبيل المثال، الإنتاج ، التدريج ، الاختبار ، التطوير)، وماهي آليات الوصول لشبكة API وهل هي متاحة للعامة او داخلياً او لعملاء والشركاء.
- حصر جميع الأصول المتربطة بالأنظمة الخاصة بك وماهي البيانات التي يتم تبادلها ومدى حساسية تلك البيانات.
- قم بتوثيق جميع جوانب واجهة برمجة التطبيقات API خصوصاً ما يرتبط بعمليات المصادقة وآلية العمل وتحديد معدل واضح لمستوى مشاركة الموارد عبر CORS ومصادر البيانات، بما في ذلك الاستعلامات والطلبات والاستجابة لتلك الطلبات.
- قم باستخدام بعض الطرق والتقنيات الالية لأغراض التوثيق المبنية على معايير أساسية والتي تشمل وثائق CI/CD
- التأكد من ان الوثائق متاحة للأشخاص المصرح لهم فقط.
- التأكد من استخدام التدابير الوقائية اللازمة مثل جدران الحماية الخاصة بواجهة برمجة التطبيقات API لجميع واجهة برمجة التطبيقات المتصلة بالأنترنت وليس فقط المتوفر في بيئة التشغيل.
- تجنب استخدام مصادر البيانات على البيئة التشغيلية باستخدام واجهة برمجة التطبيقات Api غير جاهز للعمل في تلك البيئة، وفي حال توجب عليك استخدامه فيجب تطبيق عليه جميع المعايير الأمنية نفسها التي تم تطبيقها على بيئة التشغيل.
- في حال كانت الإصدارات الحديثة من واجهة برمجة التطبيقات تحتوي على معايير امان افضل، قم بأجراء تحليل للمخاطر لاتخاذ القرارات والإجراءات التي تخفف من الضرر على الإصدار الحالي. على سبيل المثال اذا كان من الممكن تعطيل الإصدار السابق من دون الاضرار بواجهة برمجة التطبيقات والانتقال للإصدار الحديث بشكل تدريجي او من الممكن اجبار المستخدمين على الانتقال الى الإصدار الحديث بشكل عاجل.

المراجع :

مصادر خارجية :

- CWE-1059: Incomplete Documentation
- OpenAPI Initiative

API10:2019 خلل في طريقة تسجيل الاحداث والمراقبة



خصائص API	قابلية الاستغلال: 2	الانتشار : 3	قابلية الاكتشاف : 1	التأثر التقني و تأثر الاعمال: 2	خصائص API
	يستغل المهاجمون عدم تسجيل الاحداث وحركة مرور البيانات للقيام بأنشطة ضارة.	من دون وجود آلية او نظام لتسجيل حركة مرور البيانات او الاحداث سيكون هناك نظام مراقبة غير كفء، بل قد يصل في بعض الأحيان الى أستحالة تتبع الأنشطة الضارة واتخاذ الاجراءات في الوقت المناسب.		من غير الحصول على تغطية شاملة لحركة المرور والسجلات سيكون لدى المهاجمين المقدرة على اختراق الأنظمة من غير وجود أي آلية لرصدهم او تتبع تحركاتهم داخل الأنظمة.	

هل أنا معرض لهذه الثغرة؟

سيكون النظام لديك معرض اذا كان:

- لا يتم استخراج أي سجلات او لم يتم تعين عمليات التسجيل بالشكل الصحيح او لم يتم جمع السجلات بشكل كافي وناضج.
- عند عدم ضمان السجلات (على سبيل المثال في حال حقن السجلات بسجلات غير صحيح)
- لا يتم مراقبة السجلات بشكل مستمر
- لا يتم مراقبة البنية التحتية لواجهة برمجة التطبيقات API بشكل مستمر.

امثلة على سيناريوهات الهجوم :

السيناريو الاول :

عن طريق الخطأ تم تسريب احد مفاتيح إدارة المستودعات في احد قواعد البيانات العامة، تم أخطار مالك المستودع عن طريق البريد الالكتروني بشأن التسريب المحتمل، ولكن لم يقم مالك المستودع من التجاوب خلال 48 ساعة والتصرف بشأن هذا التسريب، و من المحتمل استخدام هذه المفاتيح في عمليات تسريب البيانات، ولكن بسبب عدم كفاية موارد تسجيل السجلات والاحداث لا تستطيع الشركة تقييم ومعرفة الأصول والبيانات التي تم الوصول لها او في حال تم تسريبها.

السيناريو الثاني :

تم استهداف أحد منصات مشاركة ملفات الفيديو بهجمات كسر كلمات المرور المسرية مسبقاً من أحد الهجمات السابقة. على الرغم من عدد المحاولات تسجيل الدخول غير الصحيحة لم يتم تفعيل التنبيهات خلال فترة الهجوم، وكردة فعل قام المستخدمين بالشكوى من اغلاق الحسابات الخاصة بهم بسبب عدد المحاولات، وبعد عملية تحليل السجلات الخاصة بواجهات برمجة التطبيقات API تبين ان هناك فعلاً هجوم وكان على الشركة اصدار اعلان لجميع المستخدمين بتغير كلمات المرور الخاصة.

API10:2019 خلل في طريقة تسجيل الاحداث والمراقبة

كيف أمنع هذه الثغرة؟

- قم بتسجيل جميع محاولات المصادقة الفاشلة او محاولات رفض الوصول للمجلدات او الامتدادات وكذلك جميع المدخلات المحجوبة.
- يجب كتابة السجلات بشكل متناسق لاستخدامه في عمليات إدارة السجلات ويجب ان تتضمن كافة التفاصيل التي تتيح للمحلل معرفة الأنشطة الضارة ومن قام بها.
- يجب التعامل مع السجلات باعتبارها بيانات حساسة ويجب ضمان سلامتها اثناء المرور و التخزين.
- قم بإعداد عمليات المراقبة واجعلها مستمرة ولتشمل البنية التحتية والشبكات و واجهة برمجة التطبيقات API.
- استخدام أنظمة SIEM لإدارة السجلات من جميع المصادر والأنظمة و واجهات برمجة التطبيقات.
- قم بإعداد لوحة مراقبة مخصصة للتنبيهات الأمنية وقم بتفعيل التواقيع الرقمية لرصد الأنشطة المشبوهة لرصدها في مراحلها الأولية.

المراجع :

- OWASP Logging Cheat Sheet
- OWASP Proactive Controls: Implement Logging and Intrusion Detection
- OWASP Application Security Verification Standard: V7: Error Handling and Logging Verification Requirements

مصادر خارجية :

- CWE-223: Omission of Security-relevant Information
- CWE-778: Insufficient Logging

ما التالي للمطورين؟

قد تكون مهمة إنشاء برامج آمنة وصيانتها ، أو إصلاح البرامج الموجودة ، صعبة. وكذلك هو الحال مع واجهات برمجة التطبيقات لا تختلف.

نعتقد أن التعليم والوعي من العوامل الرئيسية لكتابة برامج آمنة. كل شيء آخر من المتطلبات هو لتحقيق الأهداف المنشودة ، وهو بالأساس يعتمد على إنشاء واستخدام عمليات أمنية قابلة للتكرار وضوابط أمنية قياسية.

لدى OWASP العديد من الموارد المجانية والمفتوحة لمعالجة مشاكل الأمن منذ بداية هذه المشروع. يرجى زيارة صفحة مشاريع أواسب للحصول على قائمة شاملة بالمشاريع المتاحة.

تعليم أمن التطبيقات	يمكنك البدء في قراءة مواد مشروع OWASP التعليمي وفقًا لمهنتك واهتماماتك. للتعلم العملي ، أضفنا crAPI - Ridiculous API في خارطة الطريق الخاصة بنا. وفي الوقت نفسه ، يمكنك التدرب على WebAppSec باستخدام OWASP DevSlop Pixi Module ، وهو تطبيق ويب ضعيف وخدمة API تهدف إلى تعليم المستخدمين كيفية اختبار تطبيقات الويب الحديثة وواجهات برمجة التطبيقات للتعامل مع مشكلات الأمان ، وكيفية كتابة واجهات برمجة تكون أكثر أمانًا في المستقبل. كما يمكنك أيضًا حضور جلسات OWASP AppSec التدريبية أو الانضمام إلى لفرق OWASP المحلية .
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	يجب أن يظل الأمن مصدر للاهتمام خلال جميع مراحل المشروع. تعد ورقة المرجعية من OWASP (Cheat Sheet) نقطة انطلاق جيدة للإرشادات حول كيفية تصميم الأمان أثناء مرحلة البناء. من بين العديد من الاوراق الأخرى ، ستجد ورقة مراجع الأمان (Security Cheat Sheet) وورقة مراجع التقييم (Assessment Cheat Sheet.)
أدوات التحكم الأمنية المعيارية	إن عملية إنشاء أدوات تحكم أمنية قوية ومناسبة للاستخدام هي مهمة صعبة جدا . 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 للازمة لإنتاج تطبيقات ويب آمنة. أيضا يقدم بعض المكتبات والأدوات التي قد تجدها ذات قيمة ، مثل التحقق من صحة أدوات التحكم.
دورة حياة التطوير الآمنة	يمكنك استخدام OWASP Software Assurance Maturity Model (SAMM) لتحسين العملية عند إنشاء واجهات برمجة التطبيقات API . تتوفر العديد من مشاريع OWASP الأخرى لمساعدتك خلال مراحل تطوير API المختلفة ، على سبيل المثال ، مشروع مراجعة كود OWASP.

ما التالي لمطوري الممارسات الامنية في التطبيقات؟

نظرًا لأهميتها في بناء التطبيقات الحديثة ، فإن بناء واجهات برمجة آمنة أمر في غاية الأهمية ، ويجب أن يكون الأمن جزءًا من دورة حياة التطوير بأكملها. لم تعد اختبارات الاختراق السنوية كافية.

يجب أن تنضم DevSecOps إلى جهود التطوير ، مما يسهل اختبار الأمان المستمر عبر دورة حياة تطوير البرامج بأكملها. هدفهم هو تعزيز طريق التطوير بأتمتة الأمان ، ودون التأثير على سرعة التطوير.

في حالة تود الاطلاع والمراجعة ، راجع: https://www.devsecops.org

فهم نماذج التهديد	تأتي أولويات الاختبار من نماذج التهديد المتوقعة. إذا لم يكن لديك واحد ، ففكر في استخدام OWASP Application Security Verification Standard (ASVS) ، ودليل اختبار OWASP كمدخل. قد يساعد في رفع مستوى الوعي لفريق التطوير.
فهم دورة حياة التطبيقات	قم بالانضمام الى فريق تطوير البرمجيات لفهم دورة حياة البرامج. حيث ان مساهمتك في اختبار الامان بشكل مستمر ومتوافق مع الاوات والعمليات والاجرات التي يتفق عليها الجميع وبشكل سلسل.
استراتيجيات الاختبار	لا يجب ان تؤثر اعمالك على سرعة وتيرة التطوير بل يجب أن تختار بحكمة الأسلوب الأفضل (البسيط والأسرع والأكثر دقة) للتحقق من متطلبات الأمان. يمكن أن يكون إطار OWASP للمعرفة الأمنية ومعيار OWASP للتحقق من أمان التطبيقات مصادر جيدة لمتطلبات الأمان الوظيفية وغير الوظيفية. هناك مصادر أخرى للمشاريع والأدوات المشابهة لتلك التي يقدمها مجتمع DevSecOps
تحقيق التغطية والدقة المطلوبة	أنت حلقة الوصل بين المطورين وفرق العمليات. لتحقيق التغطية بالشكل المطلوب ، لا يجب أن تركز فقط على آلية عملها فقط ، ولكن أيضًا على التنسيق بشكل سليم. وذلك من خلال العمل بالقرب من فرق التطوير والعمليات من البداية حتى تتمكن من استغلال الجهود المبذولة. يجب أن تهدف إلى حالة دائمة من تحقيق معايير الأمان بشكل الأساسي ومستمر.
ايصال النتائج بشكل واضح	قم بالمشاركة في صنع قيمة مع اقل اختلاف مع فرق العمل. وقم بتسليم النتائج في الوقت باستخدام الأدوات المتاحة من قبل الفريق، انضم إلى فريق التطوير لمعالجة النتائج والمخرجات وقم بشرح ووصف نقاط الضعف بشكل واضح جداً وكيف سيتم إساءة استخدامها وقم بذكر بعض السيناريوهات الحقيقة لاستغلالها .

ما التالي لمطوري الممارسات الامنية في التطبيقات؟

نظرة عامة

نظرًا لأن صناعة برامج آمنة لم تركز بشكل خاص على أحدث بنية وهيكلة للتطبيقات، حيث تلعب واجهات برمجة التطبيقات دورًا مهمًا، فإن تجميع قائمة بأكثر عشرة مخاطرلواجهة برمجة التطبيقات (API)، استنادًا إلى استفتاء عام، كانت من أصعب المهام. على الرغم من عدم وجود مصادر عامة، إلا أن قائمة العشرة أخطار لا تزال تستند على الاستفتاء، ومساهمات خبراء الأمن المعلوماتي، والمناقشات المفتوحة مع مجتمع الأمن.

المنهجية

في المرحلة الأولى، تم جمع البيانات المتاحة من المصادر العامة وحول الحوادث الأمنية لواجهات برمجة التطبيقات API ومراجعتها وتصنيفها من قبل مجموعة من خبراء الأمن. وكما تم جمع هذه البيانات من منصات مكافآت الثغرات وقواعد بيانات الثغرات الأمنية، في إطار زمني مدته عام واحد. تم استخدام تلك البيانات لأغراض إحصائية.

في المرحلة التالية ، طُلب من الممارسين الأمنيين ذوي الخبرة في اختبار الاختراق حصر اكثر عشر مخاطر امنية خاصة بهم.

تم استخدام منهجية OWASP لتصنيف المخاطر لإجراء تحليل المخاطر. تمت مناقشة النتائج ومراجعتها بين الممارسين الأمنيين. للحصول على رأي OWASP حول هذه الامر ، يرجى الرجوع إلى قسم مخاطر أمان API.

نتجت المسودة الأولى من OWASP API Security Top 10 2019 عن توافق بين النتائج الإحصائية من المرحلة الأولى وقوائم الممارسين الأمنيين. ثم تم تقديم هذه المسودة لتقديرها ومراجعتها من قبل مجموعة أخرى من ممارسي الأمن ، من ذوي الخبرة ذات الصلة في مجالات أمان واجهة برمجة التطبيقات.

تم تقديم OWASP API Security Top 10 2019 لأول مرة في حدث OWASP Global AppSec في (مايو 2019).. منذ ذلك الحين ، كان متاحًا على GitHub للمناقشة العامة والمساهمات.

قائمة المساهمين متاحة في قسم الشكر والتقدير .

الإقرار

المساهمين في صناعة المحتوى

نشكر جميع المشاركين بشكل عام من خلال منصة GitHub وكذلك المشاركين من خلال وسائل ووسائط اخرى وهم :

- 007divyachawla
- Abid Khan
- Adam Fisher
- anotherik
- bkimminich
- caseysoftware
- Chris Westphal
- dsopas
- DSotnikov
- emilva
- ErezYalon
- flascelles
- Guillaume Benats
- IgorSasovets
- Inonshk
- JonnySchnittger
- jmanico
- jmdx
- Keith Casey
- kozmic
- LauraRosePorter
- Matthieu Estrade
- nathanawmk
- PauloASilva
- pentagramz
- philippederyck
- pleothaud
- r00ter
- Raj kumar
- Sagar Popat
- Stephen Gates
- thomaskonrad
- xycloops123

وكذلك المترجمين للغة العربية وهم :

- مالك الدوسري
- محمد السحيمي
- صبري صالح
- مصطفى الاقصم
- فهد الدريبي
- 0xMohammed