ClickJacking

تعريفها هي بإختصار إنك بتقدر تسرق ضغطة زر مهمة وتخلي الضحية يدوس عليها بدون ما يعرف، زي صفحة تغيير الباسورد مثلا أنت ممكن تحطها في iframe وتخفيها بشوية اكواد css وتخلي الضحية يدوس على زرار "مبروك كسبت مثلا" في حين إنه لما يدوس هيحصل إنه يغير كلمة السر بتاعته زي ما عايز المهاجم - باعتبار مفيش ادخل كلمة السر الحالية - يعني اليوزر بيعمل action في موقع مخفي في الموقع الي هو قاعد عليه حالياً.

أماكن تواجدها

- الثغرة بتكون في أي مكان/صفحة حساس/ة يكون في زرار submit بيعمل من خلاله action معين زي إضافة إميل/تغيير الباسورد/ إضافة رقم تليفون/ الإشتراك في النسخة الـ Pro/ تحويل فلوس وهكذا.

Frame busting scripts

- دي عبارة عن جافا سكريبتس وظيفتها إنها حماية ضد الـ Clickjacking attacks بحيث إن بتجبر التطبيق الي أنت موجود عليه يكون هو التطبيق الرئيسي وبتظهر أي frames مخفية وبتمنع الضغط على frames مخفية وأحيانا بتقول لليوزر إن في خطر clickjacing هنا، ودول عبارة عن Add-ons في المتصفح وبالتالي لو حد مفعل الأداة دي على المصتفح وأنا حاولت أعمل Clickjacking attack هيظهرله الـ iframe المخفي وبالتالي فشل الهجوم. (من الآخر هي بتمنع إن الصفحة يحصلها(frame

سيناريوهات

- أول سيناريو هو الـbasic ، يعني بتنفذ الهجوم بدون أي إضافات مجرد الـ click الأصلي بيكون تحت الـ click الوهمي وتكون مظبط الصفحة بحيث لما يدوس الـ click الوهمي يتنفذ الـ click الأصلي وده بينفع مع حذف الأميل أو حذف الكوكي مثلا، المهم أي زرار مفهوش إنك تدخل قيمة من عندك هو مجرد click بس.
- تاني سيناريو بيكون الـ input محتاج قيمة جواه وهنا بيكون الـ URL parameters بتتكتب في الـ input يعني لو الـ URL كده https://example.com/change-password في لو حاولت تكتب قيمة الباسورد في الـ URL هتلاقيه ظهر في الـ Input لكن مش بيتنفذ، يعني بيكون كده

https://example.com/change-password?password=123456 وبالتالي الـ Input بيكون مكتوب فيه 123456 وكده بقي زي الـ basic attack مجرد بس تحط الرابط وتظبط الصفحتين بالـ.CSS

- السيناريو التالت خاص بالـ: Frame busting scripts وهو إنك لما تكتب تاج الـ iframe بيتم منعه بواسطة الـ frame busting scripts وهنا بنلجأ لإستخدام الـ sandbox attribute ونديله قيمة allow-forms or allow-scriptsودول وظيفتهم إنهم بيمنعوا الـ Frame busting scripts من إنهم يدخلوا الـ iframe وبالتالي بيعتبر من الـ main page.
- السيناريو الرابع خلينا نقول هو تحويل الـ XSS من متوسطة الخطورة لخطيرة جداً، في بعض الأماكن بتكون مصابة بالـ XSS ولكن مش بتكون لا Reflected ولا Stored لا هي بتظهر عندك أنت بس زي لما تيجي تبعت أميل أو feedback أو تعمل جروب وهكذا وبالتالي أنت لو محتاج توصلها ليوزر تاني هتضطر تلجأ للـ Clickjacking.

الحماية من الثغرة

- السيرفر بينفذ حليين عشان يمنع ثغرة الـ Clickjacking وهما الـ:

1-الـ X-Frame-Options: هنا ممكن نستخدم 3 أوامر وهما

deny - وده بيمنع إن الصفحة دي يتعملها frame في أي مكان حتى الموقع نفسه.

same origin - ودي بتسمح إن الصفحة يتعملها frame داخل الموقع نفسه لكن موقع تاني لا.

allow-from: - وده بياخد قيمة أو أكتر لـ URL موقع معين وبالتالي بيسمح للموقع ده إنه يعمل frame للصفحة عنده.

- الـ X-Frame-Options مش دايما بيدعمه المصتفح.

2- الـ Content-Security-Policy: ودي مشهورة إنها آلية دفاع ضد هجمات زي الـXSS,Clickjacking ، وبيكون عبارة عن Response header بيبعته الـ Server.

- الـ CSP وظيفته بيقدم معلومات لمتصفح المستخدم عن المصادر المسموح بيها والموارد الي يمكن تطبيقها(الـ actions الي اليوزر يقدر يعملها).
- أفضل حل لمنع الثغرة دي إستخدام الـ frame-ancestors داخل الـ CSP وده ليها أكتر من قيمة زي none, self, WebsiteURL

- Content-Security-Policy: frame-ancestors 'self';

- الـ CSP أفضل من الـ X-Frame-Options لإنك بتقدر تديله أكتر من أمر زي مثلا

Frame-ancestors: 'self' https://normal-website.com https://*.example.comوهكذا.

- -الأفضل إستخدام الاتنين مع بعض وتكوين multi-layer defense عشان تتأكد من منع الثغرة كلياً.