Czy klikanie w linki jest bezpieczne?

Jan Góra, *12 listopada 2017*

1 Wstęp

Często w internecie, a szczególnie w mediach społecznościowych, spotyka się wypowiedzi typu: *"Nie klikajcie w tego linka, bo to wirus"*. Ale czy oby na pewno samo odwiedzenie odnośnika może zainfekować nasz komputer? I czy wirus to rzeczywiście główne zagrożenie jakie na nas czyha po odwiedzeniu feralnego linka?

W tekście skupimy się jedynie na bezpieczeństwie trzech narzędzi niezbędnych do swobobnego poruszania się po internecie jakimi są *HTML*, *CSS* i *JavaScript*. Nie będziemy więc oceniać bezpieczeństwa wszelakich dodatków takich jak *PDF-viewer* czy *FlashPlayer*, które znane są z dużej ilości podatności i luk w bezpieczeństwie. Poruszę takie techniki jak *Phishing* czy *XSS Injection* oraz zaprezentuję autorskie konstrukcje wybranych ataków.

2 Phishing

Bardzo znaną i rozpowszechnioną techniką przejmowania wrażliwych danych użytkowników (takich jak login, hasło czy dane karty kredytowej) jest tzw. *Phising*. Polega on na tym, że ofiara zostaje przekierowana na fałszywą stronę, np stronę logowania, gdzie proszona jest o podanie uwierzytelniających danych. Tutaj kreatywność przestępców nie zna granic, a łamanym obiektem nie są zabezpieczenia, tylko człowiek. Świetnie wpasowuje się tutaj cytat: *"Łamałem ludzi, nie hasła", Kevin Mitnick*.

2.1 PunyCode

Pasek URL jest to pierwsza rzecz jaką powinniśmy sprawdzać podczas wpisywania wrażliwych danych. Bardziej lub mniej zaawansowany użytkownik internetu powinien od razu odróżnić od siebie dwa linki `google.services.com` i `services.google.com`. (Gdzie pierwszy link przynależy do domeny `services.com`, a drugi do `google.com`). Odrobina rozsądku połączona z tą podstawoą wiedzą wystarcza, aby uchronić się przed większością ataków phishingowych. Co jednak gdy adres URL wygląda niemal identycznie jak docelowy adres? Niewątpliwie im większe podobieństwo adresu tym większa szansa na "złowienie" ofiary. Tutaj pomocnym dla atakujących może okazać się *PunyCode*.

> *[7] Punycode, zdefiniowany w RFC 3492 ↓, jest konwersją łańcuchów Unicode do ograniczonej strony kodowej obsługiwanej przez serwery DNS. Adresy URL w kodowaniu Punycode mają postać: xn–znakiascii-kodznakowunicode, na przykład:*
>
> - *Unicode - www.kraków.pl*
> - *Punycode - www.xn–krakw-3ta.pl*

Instytut Informatyki Uniwersytetu Wrocławskiego, 2017–2018
Esej przygotowany w ramach zajęć *Kultura bezpieczeństwa komputerowego*

Atak z wykorzystaniem wyżej wymienionego kodowania nazywany jest *Atakiem homograficznym (ang. homograph attack)*. Chociaż pomysł na atak jest bardzo stary, to zyskał on popularność dopiero niedawno, za sprawą rozgłośnionego bloga opublikowanego *14 kwietnia br.* przez studenta matematyki - *Xudong Zheng'a* [1]. Stworzył on domenę z certyfikatem SSL `apple.com` (`xn--pple-43d.com`), złudnie przypominającą `apple.com`. Wszystkie zastosowane tam znaki pochodzą z cyrylicy. Dzięki tej popularności, przeglądarki intensywnie zajęły się ochroną użytkowników przed tego typu atakami, a fałszywa domena `apple.com` błyskawicznie została "załatana" i w większości przeglądarek, w pasku *URL* zobaczymy adres zaczynający się od *xn–*. Jako, że w dniu pisania tego tekstu podatność ta jest wciąż niwelowana to domeny, które jeszcze niedawno pokazywały się jako znaki *Unicode* zastępowane są adresami *ASCII*. Przykładowo domena `sieci.pl` (`xn--e1ayzdec.pl`), której znaki również pochodzą z cyrlicy, w najnowszej wersji *Chrome* wyświetla się jako `xn--e1ayzdec.pl`, zaś w *Firefox* zobaczymy: `sieci.pl`

2.2 Target='_blank'

Czas na dużo ciekawszą oraz bardzo lekceważoną podatność, która potocznie nazywana jest *Target='_blank' vulnerability* [2]. Większość stron internetowych przy odnośnikach posiada atrybut: `target='_blank'`, który służy do otworzenia linku w nowej karcie. Ale mało kto zdaje sobie sprawę, z jakimi zagrożeniami wiąże się ta podatność. Okazuje się, że po kliknięciu w taki link, docelowa strona (*JavaScript*) ma dostęp do obiektu `window.opener`, który wskazuje na obiekt `window` na stronie, na której odwiedziliśmy łącze. Co prawda dzięki `Same-origin policy` dostęp ten jest bardzo ograniczony. Realny dostęp mamy tak naprawdę tylko do dwóch własności - `window.opener.location` i `window.opener.length`. Nie mamy wystarczających przywilejów do odczytania wartości `location`, ale możemy ją zmienić, co spowoduje przekierowanie strony na zmieniony adres. Poważnym zagrożeniem byłoby gdyby dało się zmienić adres na dowolny, czyli na przykład `window.opener.location = "javascript:something"` co by oznaczało, że możemy wstrzyknąć kod *JavasScript* do dowolnej strony, ale o tym później. Na szczęście czegoś takiego zrobić nie możemy. Jednak sam fakt, że ktoś może manipulować adresami *URL* moich kart przyprawia mnie o dreszcze. Ciekawym i bardzo niezrozumiałym dla mnie jest dostęp do wartości `window.opener.length`. Ale co to właściwie jest? W dokumentacji znajdziemy:

> `window.length` returns the number of frames (either `<frame>` or `<iframe>` elements) in the window.

Co oznacza, że wartość ta to nic innego jak ilość ramek w danym dokumencie. Od razu nasuwa się pomysł, że wartość tę można wykorzystać do wykrywania akcji na stronie docelowej. W połączeniu z `document.refferer` możemy stworzyć bardzo efektywną stronę phishingową, która oprócz manipulowania wartością `location` może stwierdzić z jakiej domeny użytkownik przybył i na tej podstawie podmienić adres *URL* "tematycznie".

Łatwo zauważyć, że komunikacja w tym przypadku jest jednostronna – strona z której odwiedziliśmy link nie ma dostępu do odwiedzonej przez nas strony. Co nie znaczy, że nie jest możliwe zrobienie komunikacji dwustronnej. Do tego celu zamiast otwierać odnośnik poprzez `<a target=_blank>` możemy go otworzyć poprzez użycie kodu `var _window = window.open(...)` dzięki czemu uzyskujemy dostęp dwustronny. Teraz zaczyna dziać się jeszcze ciekawiej. Na pewno nie raz spotkałeś się komunikatem *"PDF Viewer, Focusing, please wait..."*, najczęściej równolegle otwierane są okienka w tle, które zawierają jakąś reklamę. *JavaScript* w przeciwieństwie do wtyczki *PDF'a* nie ma możliwość przełączenia się pomiędzy oknami, ale za to ma możliwość otworzenia nowej karty i poprzez funkcję `alert()` może powrócić do pierwotnej karty, jednak nie jest to tak efektywne jak otworzenie nowego okna, które prawdopodobnie zostanie zauważone po bardzo długim czasie. Co to oznacza w praktyce? Wykorzystując `window.open` i fakt, że okno zostanie otworzone w tle, mamy "kontrolę" nad kartą tak długo aż nie zostanie ona zamknięta.

Na cele tego tekstu przygotowałem prostą stronę *proof of concept* – `http://terjanq.github.io/window.opener` wykrywającą akcję zmiany strony używając `window.length`, która po kliknięciu w dowolne miejsce przenosi nas na stronę *facebook'a*, gdzie po ok 20 sekundach zostajemy wylogowani, a przy próbie ponownego zalogowania się zostajemy przekierowani na podmienią stronę.

Rozwiązaniem, aby uchronić nasz serwis przed tego typu atakiem jest dołączenie do każdego odnośnika atrybutu `rel=ńoopener noreferrer"`, oraz wyzerowanie referencji do obiektu `window.opener` poprzez `window.opener = null`

3 XSS Injection

Cross-site Scripting (XSS) Injection Jest chyba jedną z najbardziej niebezpiecznych podatności. Polega ona na wstrzyknięciu kawałka kodu *JavaScript* do strony ofiary. Istnieje mnóstwo technik w jaki sposób można próbować to zrobić, ale my skupimy się tylko na wstrzyknięciu kawałka kodu do odnośnika.

Wyobraźmy sobie serwis, który w przypadku odwiedzenia strony *example.com/aaaa* wyświetla stronę:

> *Error 404*
> *Strona 'aaaa' nie została odnaleziona*

Jeśli strona nie zamienia znaków specjalnych na odpowiedniki html, możemy wstrzyknąć kawałek kodu przykładowo poprzez `example.com/<script>...</scrip>`, który zamiast wyświetlić nazwę strony jako ciąg znaków, wykona kod zawarty w znacznikach `script`. Aby adres wyglądał bardziej "niewinnie" możemy użyć *URIendcode*, który po zamaskowaniu może wyglądać następująco: `http://example.com/ssid=\%3c\%73\%63\%72\%69\%70\%74\%3e\%61\%6c\%65\%72\%74\%28\%27\%78\%73\%73\%27\%29\%3c\%2f\%73\%63\%72\%69\%70\%74\%3e`, który jest niczym innym jak `http://example.com/ssid=<script>alert('XSS')</script>`.

Zagrożeń wynikających z *XSS Injection* jest mnóstwo. Dzięki *JavaScript* można uzyskać dostęp do części *ciasteczek*, które mogą zawierać takie dane jak id sesji, pozwalające zalogować się jako dany użytkownik bez znania jego loginu i hasła. Można również przekształcić stronę w taki sposób, żeby namówić użytkownika do podania loginu i hasła czyli zastosować wcześniej omawiany *phishing*. Oprócz tych dwóch oczywistych rzeczy, można jeszcze na przykład przesyłać informacje o wszystkich podstronach w danej domenie, oraz wszystkich domenach z którymi dana strona może się połączyć. Na przykład, wstrzykując kod *XSS* na stronie *facebook'a* moglibyśmy zobaczyć wszystkie wiadomości jakie dany użytkownik przesłał. Niestety w takiej sytuacji użytkownik nie ma możliwości obronienia się przed tego typu atakiem. Winnym jest tutaj serwis, który nie zabezpieczył serwisu wystarczająco dobrze. Również przeglądarki starają się niwelować szkody w przypadku strony podatnej, poprzez blokowanie takiego kodu zanim zdąży się wykonać. Jednak wykrycie kiedy powinny to robić nie jest łatwym zadaniem, więc jest to jedynie "rzucanie przestępcom, kłód pod nogi".

Z tego też powodu za znalezienie takiej podatności na większości dużych, ale nie tylko, serwisach oferowane są bardzo przyzwoite nagrody pieniężne. Między innymi poprzez programy *Bug Bounty*.

4 Iframe attack

Iframe attack jest to jeden z bardzo niebezpiecznych ataków. Polega on na wysyłaniu żądań do adresów *IP* wykorzystując do tego celu między innymi elementy takie jak `iframe` czy też zwykłe obrazki `img`. Tak jak firmy bardzo dbają o bezpieczeństwo sieci wewnętrznych, tak w

gospodarstwach domowych domyślne hasło do routera bardzo często nie zostaje w ogóle zmienione. Co daje pewne możliwości na skonstruowanie ataku siłowego, próbującego uwierzytelnić się do routera, a następnie zmienić adresy serwerów *DNS* na własne.

> *[7] Domain Name System (DNS, pol. „system nazw domenowych") – system serwerów, protokół komunikacyjny oraz usługa obsługująca rozproszoną bazę danych adresów sieciowych. Pozwala na zamianę adresów znanych użytkownikom Internetu na adresy zrozumiałe dla urządzeń tworzących sieć komputerową. Dzięki DNS nazwa mnemoniczna, np. pl.wikipedia.org jest tłumaczona na odpowiadający jej adres* IP*, czyli 91.198.174.192.*

Widać zagrożenie jest ogromne. Po sukcesywnej zmianie takich adresów, atakujący jest w stanie przechwycić większość ruchu sieciowego, a jakie konsekwencje mogą z tego wyniknąć pozostawiam czytelnikowi do własnej oceny.

Prześledźmy konstrukcję takiego ataku na przykładzie bardzo popularnej brazylijskiej strony – `politica.estadao.com.br` [3], która w 2014 roku została zainfekowana kodem wykonującym taki atak. Atakujący zaczyna od pobrania od użytkownika publicznego adresu *IP*, nie jest to trudne zadanie, ponieważ jak sama nazwa wskazuje, jest to adres publiczny. Następnie poprzez analizę otrzymanego adresu *IP* ustala dodatkowe informacje takie jak: dostawcę internetu, ta informacja jest również dostępna publicznie, można to sprawdzić używając takich narzędzi jak *WHOIS IP Lookup Tool* i na tej podstawie może próbować odgadnąć adres wewnętrzny *IP* routera, możliwe hasła do konta administratora oraz podatności routerów jakie dany dostawca oferuje, np bardzo popularnym adresem jest *192.168.0.1* czy *192.168.1.1* oraz login/hasło *admin/admin*. Kiedy zbiór adresów jest już ograniczony czas przystąpić do ataku siłowego czyli do sprawdzania popularnych loginów i haseł oraz wykorzystania wszystkich możliwych podatności, aby następnie móc dobrać się do konfiguracji serwerów DNS. Prześledźmy kilka pakietów *GET* jakie atakujący wysyłał do routera:

- *GET http://root:root@192.168.32.10/dnsProxy.cmd*
- *GET http://root:root@192.168.32.10/dnscfg.cgi*
- *GET http://admin@192.168.32.100/dnsProxy.cmd*
- *GET http://admin:admin@192.168.32.100/dnscfg.cmd*

Jak widać wykonanie takiego ataku nie jest bardzo skomplikowane oraz opiera się on na prostej idei. Warto zmienić domyślne hasło routera, nawet jeśli wygląda bezpiecznie. Bardzo ciekawym przykładem są routery od *TP-link*, w których wydawałoby się, że domyślne hasła to losowy ciąg liter i cyfr, a w rzeczywistości to pewna kombinacja liter i cyfr adresu *MAC*, a dokładniej kombinacja liczb w systemie dwójkowym, skrócona do zapisu szesnastkowego. Jednak jako, że nie jest to przedmiotem tego tekstu, zaciekawionych czytelników odsyłam do źródła [4].

5 CSS attack

Wydawałoby się, że tak niewinny język jakim jest *CSS (Cascading Style Sheets)*, manipulujący jedynie wyglądem strony i elementów na niej zawartych, posiada bardzo małe możliwości skonstruowania ataku. A jednak jest to możliwe. Swojego czasu było możliwe zastosowanie ataku *CSS Injection*, który w skrócie wykorzystywał podatności przeglądarek pozwalające na zaimportowanie skryptu *JavaScript* używając funkcji `@import`, czyli *summa summarum* był to wciąż atak *XSS*. Dzięki różniącej się składni miał szansę "prześlizgnąć" się przez filtry uniemożliwiające bezpiecznie pośrednie załączenie kodu *JavaScript*. Zainteresowanych czytelników odsyłam do lektury [5].

Atak, który skonstruowałem, będzie polegał tym razem na deanonimizacji użytkownika. Wyobraźmy sobie, że korzystamy z pewnego czatu czy serwisu, gdzie chcielibyśmy pozostać

anonimowi. Nagle otrzymujemy ciekawy link od rozmówcy i nie możemy powstrzymać się od tego, aby zobaczyć co się w nim znajduje. Jak to mówią: "Ciekawość to pierwszy stopień do piekła". Zupełnie nieświadomi, odwiedzamy stronę, która zapełniona jest przyciskami *lubię to!* od *facebook'a* czy *zasubskrybuj* od *Youtube'a*, które po kliknięciu nie wymagają, żadnej dodatkowej autoryzacji. Kluczem jest tutaj to, że nie jesteśmy w stanie ich zobaczyć, ponieważ wszystkie elementy mają *opacity* (pol. nieprzezroczystość) ustawione na 0%, czyli są dla użytkownika niewidzialne. Jeden niefortunny klik i atakujący może odkryć naszą tożsamość, sprawdzając kto właśnie polubił stworzoną do tego ataku stronę na *Facebooku* czy zasubskrybował kanał na *Youtube*. Oczywiście metoda ta zadziała tylko wtedy kiedy jesteśmy aktualnie zalogowani do któregokolwiek z tych serwisów.

6 IP i DDoSing

Bardzo popularnym wśród społeczności graczy jest, banalny do przeprowadzenia, atak mający na celu zdobycie publicznego adresu *IP* innego gracza. Głównym celem takiego działania jest późniejsze wysłanie ataku *DDoS* na pozyskany adres w celu zatrzymania pracy routera, bądź spowodowania tzw. "lagów".

> *[7] DDoS (ang. distributed denial of service, rozproszona odmowa usługi) – atak na system komputerowy lub usługę sieciową w celu uniemożliwienia działania poprzez zajęcie wszystkich wolnych zasobów, przeprowadzany równocześnie z wielu komputerów (np. zombie).*

Powodów takiego działania może być wiele, zaczynając od wyładowania frustracji spowodowanych porażką, a kończąc na zdobyciu przewagi w rozgrywce poprzez opóźnienie przepustowości łącza przeciwnika. Najprostszym sposobem na skonstruowanie takiego ataku jest wysłanie konkretnej osobie unikatowy adres *URL*, który będzie w stanie pobrać adres *IP* ofiary. Do tego celu można przykładowo użyć skracacza linków, aby link nie był zbyt podejrzany, który będzie wskazywać na nasz serwer, który pobierze interesujące nas dane, a następnie przekieruje żądanie przykładowo do śmiesznego obrazka na popularnej stronie, tak aby "zatrzeć" ślady. Idea banalna, a zarazem bardzo skuteczna.

7 Bonus i Podsumowanie

Zupełnie innym przypadkiem, kiedy kliknięcie w linka może wiązać się z poważnymi problemami jest historia blogera i badacza bezpieczeństwa - *Christian'a Haschek'a* [6]. Pewnego razu postanowił dołączyć do małej rozkwitającej społeczności na kanale *IRC*, która związana była właśnie z bezpieczeństwem komputerowym, nowościami hakerskimi czy nawet zwyczajnie z polityką. Jak pisze: *"The atmosphere was nice and not at all hostile. The topics were mostly politics and the lastest news from hacks on the news."*. Spotkał tam chłopaka, z którym trochę porozmawiał, ale szybko zorientował się, że to tzw. "script kiddie", czyli kolokwialnie mówiąc – dzieciak, który używa na narzędzi do przełamywania zabezpieczeń stron niekoniecznie posiadając jakąkolwiek wiedzę. Jednak nie powstrzymało go to od tego, aby odwiedzić link, który otrzymał od wyżej wymienionego "kolesia" z dopiskiem – *"lol, check this out - http://political-party-website/stuff"*. Link nie wyglądał podejrzanie, a jako że w odwiedzeniu odnośnika nie ma nic nielegalnego, zaciekawiony postanowił sprawdzić co się tam znajdowało. Okazało się, że folder */stuff* posiadał włączone listowanie katalogów w taki sposób, że był w stanie zobaczyć wszystkie pliki znajdujące się na serwerze. Zrobił kilka kliknięć, a następnie wyłączył stronę i jako, że było późno – poszedł spać. Rano podczas rutynowego przeglądania wiadomości, wszystkie portale donosiły, że strona została zhackowana, a loginy oraz zahaszowane hasła zostały upublicznione – *""The website of a political party has been hacked last night"'*. Cztery miesiące później został powitany w domu przez prokuratora, funkcjonariuszy policji oraz agentów jednostki antyterrorystycznej.

Został poinformowany, że są w posiadaniu dowodów na to, iż jest odpowiedzialny za zhakowanie strony partii. Podczas składania zeznań dowiedział się również, że od tygodni był na podsłuchu oraz że jego adres *IP* znalazł się dokładnie w trakcie przeprowadzania ataku, co mogło być spowodowane usterką użytego *VPNa*. Cały sprzęt elektroniczny został skonfiskowany i odzyskany po upływie jednego roku. Na szczęście cała historia zakończyła się umorzeniem sprawy z powodu braku dowodów oraz tego, że kliknięcie w odnośnik nie było przestępstwem.

Jak widać na załączonym przykładzie, nawet spec od bezpieczeństwa może popaść w porządne tarapaty poprzez kliknięcie w jeden niewinny link.

Pozwolę sobie jeszcze raz zacytować *Kevin'a Mitnick'a - "Łamałem ludzi, nie hasła"* i myślę, że będzie to bardzo dobre zakończenie tego tekstu.
To od od nas, użytkowników, zależy jak bardzo nasze przeglądanie internetu będzie bezpieczne, a pomysłów na zaatakowanie jest tyle ile ludzi na świecie. Warto dbać o to, aby nasze oprogramowanie zawsze było jak najbardziej aktualne, w szczególności przeglądarki internetowe, w których podatności łatane są błyskawicznie, a hasła nie były trywialne.

Literatura

[1] Xudong Zheng *Phishing with Unicode Domains*, April 14, 2017
`https://www.xudongz.com/blog/2017/idn-phishing/`

[2] Alex *Target="_blank" – the most underestimated vulnerability ever*, May 4 2016
`https://www.jitbit.com/alexblog/256-targetblank---the-most-underestimated-vulnerability-ever/`

[3] Fioravante Souza *Compromised Website Used To Hack Home Routers*, September 11, 2014
`https://blog.sucuri.net/2014/09/website-security-compromised-website-used-to-hack-home-routers.html`

[4] Paul Ducklin *The Wi-Fi router with a password that takes just 70 guesses...*, January 27, 2016
`https://nakedsecurity.sophos.com/2016/01/27/the-wi-fi-router-with-a-password-that-takes-just-70-guesses/`

[5] James Kettle *Detecting and exploiting path-relative stylesheet import (PRSSI) vulnerabilities*, February 17, 2015
`http://blog.portswigger.net/2015/02/prssi.html`

[6] Christian Haschek *That (not so) awesome time the police raided my home*, November 8, 2015
`https://blog.haschek.at/2015-that-not-so-awesome-time-the-police`

[7] Wikipedia, `https://www.wikipedia.org/`