Maciej Rzymski
Dariusz Sęk
Radosław Dobrzeniecki
Maciej Oliwa

Gdańsk, dn. 05.05.2016 r.

Technologie Społeczeństwa Informacyjnego - projekt
Temat: *Bezpieczeństwo w HTTP*

Spis treści

1. Content Security Policy

1.1. Historia

Oryginalnie standard został nazwany Content Restriction, przedstawiony w roku 2004 przez Roberta Hansena. Został pierwszy raz zaimplementowany w przeglądarce Firefox 4 i szybko wprowadzony również w innych przeglądarkach. Wersja 1 tego standardu została opublikowana w 2012 roku jako kandydat do rekomendacji przez W3C i szybko została opublikowana kolejna wersja (Poziom 2) w 2014 roku. Na rok 2015 draft Poziomu 3 jest w fazie rozwojowej i nowe możliwości szybko są adaptowane przez przeglądarki.

Eksperymentalna wersja CSP posiadała następujące nazwy nagłówków:

- Content-Security-Policy - standardowa nazwa nagłówka zaproponowana przez dokument W3C. Google Chrome wspiera je od wersji 25, Firefox od wersji 23, wydanej 6 Sierpnia 2013.
- X-WebKit-CSP - wycofany, eksperymentalny nagłówek zaimplementowany do Google Chrome i innych bazowanych na WebKit'cie przeglądarek (Safari) w 2011 roku.
- X-Content-Security-Policy - wyfoany, eksperymentalny nagłówek dodany w przeglądarkach bazujących na silniku Gecko 2.

Strona może deklarować wiele różnych nagłówków CSP, jak również mieszać wymuszenie i nagłówki tylko do raportowania. Każdy nagłówek przetwarzany jest oddzielnie przez przeglądarkę. CSP może być też dostarczany bezpośrednio w kodzie HTML korzystając z tagu HTML META, jednak jego efektywność w tym przypadku będzie ograniczona.

Wiele frameworków webowych wspiera CSP, dla przykładu AngularJS (natywnie) oraz Django (jako middleware). Instrukcje dla Ruby on Rails są zamieszczone na GitHubie. Wsparcie dla frameworka webowego jest wymagane tylko jeżeli zawartość CSP w jakiś sposób zależy od stanu aplikacji sieciowej - jak np. użycie stanu pochodzenia nonce. W przeciwnym wypadku CSP jest raczej statyczne i może być dostarczone z poziomu aplikacji webowej powyżej aplikacji, np. na load balancerze albo serwerze webowym.

Stan na rok 2015 nowych standardów bezpieczeństwa przeglądarek zaproponowanych przez W3C, większość komplementarna z CSP:

- Sub-Resource Integrity (SRI) aby zabezpieczyć tylko znane, zaufane źródłowe pliki (typowo JavaScript, CSS) są załadowywane z serwerów osób trzecich (głównie content delivery networks (CDNs))
- Mixed Content aby zapewnić jasność intencji polityki przeglądarki dla stron ładowanych przez HTTPS do linkowania zawartości przesyłanych w czystym HTTP
- Upgrade Insecure Requests podpowiadający przeglądarce jak postępować z linkami dziedzicznymi na stronach migrowanych do HTTPS'a
- Credential Management jako jednolite API JavaScriptu do udostępniania wiadomości o użytkowniku, aby zapewnić skomplikowane schematy logowania
- Regerrer Policy - dodatek do CSP aby podpowiedzieć przeglądarce jak generować nagłówki Refererowe (nagłówek odpowiedzialny za informacje na temat gdzie strona zgłasza się po swoją zawartość albo informacje na temat skąd przybywa na daną stronę użytkownik).

1.2. Bezpieczeństwo

Głównym zadaniem CSP jest zapewnianie bezpieczeństwa dla zasobów sieciowych - stron oraz usług. Jest to dodana warstwa która pomaga wykrywać i przeciwdziałać atakom takim jak Cross-Site Scripting (XSS) lub wstrzykiwaniu danych. Celem takich ataków jest wykradanie danych, podszywanie się pod strony lub dystrybucja oprogramowania malware'owego.
CSP 1.1 zawiera definicję dwóch nagłówków odpowiedzi HTTP, które serwer może ustawić. Są to:

Content-Security-Policy
Content-Security-Policy-Report-Only

W fazie eksperymentalnej znajdują się odpowiedniki tych nagłówków, które będzie można umieszczać w sekcji <head> dokumentu HTML, w postaci znaczników <meta> (http-equiv="content-security-policy" i http-equiv="content-security-policy-report-only"). W treści nagłówka definiujemy Security Policy dla naszej aplikacji. Do dyspozycji są tzw. dyrektywy, za pomocą których można bardzo szczegółowo określić, co przeglądarka może załadować i uruchomić w kontekście wykonywanego kodu.

script-src to dyrektywa odpowiedzialna za definiowanie reguł dla JavaScript. Może ona przyjąć postać ciągu zawierającego wskazanie konkretnych URI, z których skrypty mogą zostać uruchomione (jeśli dany URI nie znajdzie się na tej liście, dołączenie go znacznikiem <script src="http://URI/script.js"> nie spowoduje jego załadowania, a przeglądarka zgłosi błąd naruszenia polityki CSP).

Ciąg może zawierać dowolną ilość adresów. Mogą to być adresy jednoznacznie identyfikujące daną domenę, ale możemy używać także tzw. *wildcard*, czyli znaku "*" określającego dowolną wartość w miejscu, gdzie został on zastosowany. Może to być: protokół, nazwa domenowa, subdomena czy port.

Ostatnią możliwą wartością jest "self" (w cudzysłowie – to bardzo ważne, by o tym pamiętać, w innym przypadku słowo self zostanie potraktowane jako… host o takiej nazwie). Słowo to wskazuje na możliwość wykonywania wszystkich skryptów JavaScript pochodzących z dokładnie tej samej domeny, z której pochodzi wywołujący dokument.

Pora na przykład praktyczny. Wyobraźmy sobie sytuację, że nasz serwis wszystkie zewnętrzne pliki z kodem JavaScript zaczytuje z serwera CDN (Content Delivery Network) zlokalizowanego pod adresem http://cdn.greatstuff.serv. Dodatkowo kilka prostych skryptów znajduje się

bezpośrednio na naszym serwerze. W takiej sytuacji prawidłowo skonstruowany nagłówek CSP powinien mieć taką postać:

Content-Security-Policy: script-src 'self' http://cdn.greatstuff.serv
Content-Security-Policy: script-src 'self' http://cdn.greatstuff.serv

Oznacza to, że w przypadku udanego ataku Persistent/Stored XSS oraz osadzenia przez atakującego w payloadzie XSS znacznika <script> zawierającego atrybut ‚src' wskazujący na skrypt w domenie http://agresor.evil.com – kod nie załaduje i nie wykona się, gdyż adres nie jest dopuszczony przez CSP.

Dokładnie w taki sam sposób powstają reguły dla pozostałych dyrektyw, odpowiedzialne za różne elementy:
default-src jest brana pod uwagę w momencie, gdy nie jest zdefiniowana żadna dyrektywa dla konkretnego typu zasobów (przykładowo – gdy nie ma definicji script-src, ale jest default-src – to ona będzie brana pod uwagę w przypadku rozpatrywania CSP dla plików JavaScript). Wartość zdefiniowana w default-src nie podlega dziedziczeniu, to znaczy, że jawne zdefiniowanie dyrektywy dla innego typu zasobów nadpisuje reguły z default-src (ale tylko i wyłącznie dla tego konkretnego typu, dla pozostałych, nie zdefiniowanych jawnie, nadal ma zastosowanie reguła z default-src).

connect-src odpowiada za wszystkie połączenia, realizowane przez przeglądarkę poprzez:

- Web Sockets
- metodę open() obiektu XMLHttpRequest
- obsługę odbierania informacji o zdarzeniach od serwera (server sent events) przez mechanizm EventSource

font-src pozwala na określenie CSP dla fontów ładowanych poprzez reguły @font-face kaskadowych arkuszy stylów CSS

frame-src, bardzo ważna w kontekście ataków XSS, w których agresor próbuje osadzić element <iframe> (ramkę) w źródle strony i poprzez jej atrybut ‚src' załadować i wykonać złośliwy kod. frame-src pozwala nam na ograniczenie potencjalnych adresów, z których mogą być załadowane pliki JavaScript w taki sam sposób, jak dyrektywa script-src.

img-src definiuje reguły CSP dla obrazków. Reguły te zadziałają dla:

- atrybutu "src" znacznika <img>

- wartości url(…) lub image(…) w kaskadowych arkuszach styli CSS
- wartości atrybutu href znacznika <link rel>, jeśli odnosi się on do obrazka (np. ikony)
- media-src odpowiada za atrybut src elementów <video>, <audio>, <source> oraz <track>.

  object-src – jak wyżej, ale dla elementów <object> i <embed>

  style-src – dla zewnętrznych arkuszy styli CSS

upgrade-insecure-requests - służy do automatycznego konwertowania niezabezpieczonych żądań (HTTP) do poszczególnych zasobów na alternatywne żądania zabezpieczone (HTTPS). Konwersja (ang. rewriting) odbywa się przezroczyście dla użytkownika. Dyrektywa nie ma zastosowania w przypadku nagłówka Content-Security-Policy-Report-Only.

Ostatnią dyrektywą jest report-uri, nie będąca bezpośrednio odpowiedzialna za konkretną politykę. report-uri służy do zdefiniowania adresu URL, pod który przeglądarka ma wysyłać raporty o naruszeniu reguł Content Security Policy.

1.3. Analiza wykorzystania

Przygotowana została lista 989 najpopularniejszych stron internetowych z Polski (289), Wielkiej Brytanii (377) oraz USA (323) na bazie danych z portalu https://www.quantcast.com/top-sites/ [6][7][8]. Lista nie uwzględnia stron, których adres URL oraz popularność została ukryta (“Hidden profile”).

Przy użyciu skryptu na każdy wyszczególniony adres URL zostało wysłane żądanie HTTP HEAD (wraz z odpowiednimi nagłówkami symulującymi zwykłego użytkownika i przeglądarkę WWW) w celu pobrania odpowiedzi serwera i wyszukania nagłówka HTTP “Content-Security-Policy”. Obsługiwane są również przekierowania wskazywane za pomocą nagłówką “Location” wraz z uwzględnieniem przejścia pomiędzy protokołem HTTP i HTTPS, ponieważ celem jest pobranie odpowiedzi serwera dla strony, którą ostatecznie widzi użytkownik wchodząc na dany adres URL.

Analiza polega na zbadaniu ile stron korzysta z Content Security Policy (poprzez wysłanie nagłówka w odpowiedzi żądania HTTP) oraz jaki to stanowi procent ruchu, biorąc pod uwagę tylko wyszczególnione najpopularniejsze strony w danym kraju i ich liczbę miesięcznych użytkowników.

Tabela 1. Rezultaty analizy wykorzystania nagłówka Content-Security-Policy przez najpopularniejsze strony WWW w Polsce, Wielkiej Brytanii oraz USA [6][7][8].

	Liczba przeanalizowanych stron WWW	**Liczba stron stosujących nagłówek CSP**	**% ruchu stron stosujących nagłówek CSP do wszystkich przeanalizowanych stron**
Polska	289	4	3,48
Wielka Brytania	377	5	5,71
USA	323	14	10,65

Tabela 2. Wykaz najpopularniejszych stron WWW w Polsce wykorzystujących nagłówek Content-Security-Policy.

URL	**Zawartość nagłówka CSP**
buzzfeed.com	upgrade-insecure-requests
cnn.com	default-src 'self' http://*.cnn.com:* https://*.cnn.com:* *.cnn.net:* *.turner.com:* *.ugdturner.com:* *.vgtf.net:*; script-src 'unsafe-inline' 'unsafe-eval' 'self' *; style-src 'unsafe-inline' 'self' *; frame-src 'self' *; object-src 'self' *; img-src 'self' * data: blob:; media-src 'self' *; font-src 'self' *; connect-src 'self' *;
rottentomatoes.com	frame-ancestors 'self' rottentomatoes.com *.rottentomatoes.com ;
washingtonpost.com	upgrade-insecure-requests

Tabela 3. Wykaz najpopularniejszych stron WWW w Wielkiej Brytanii wykorzystujących nagłówek Content-Security-Policy.

URL	**Zawartość nagłówka CSP**
myjobscotland.gov.uk	frame-ancestors 'self'

buzzfeed.com	upgrade-insecure-requests
cnn.com	default-src 'self' http://*.cnn.com:* https://*.cnn.com:* *.cnn.net:* *.turner.com:* *.ugdturner.com:* *.vgtf.net:*; script-src 'unsafe-inline' 'unsafe-eval' 'self' *; style-src 'unsafe-inline' 'self' *; frame-src 'self' *; object-src 'self' *; img-src 'self' * data: blob:; media-src 'self' *; font-src 'self' *; connect-src 'self' *;
rottentomatoes.com	frame-ancestors 'self' rottentomatoes.com *.rottentomatoes.com ;
washingtonpost.com	upgrade-insecure-requests

Tabela 4. Wykaz najpopularniejszych stron WWW w USA wykorzystujących nagłówek Content-Security-Policy.

URL	**Zawartość nagłówka CSP**
twitter.com	script-src https://connect.facebook.net https://cm.g.doubleclick.net https://ssl.google-analytics.com https://graph.facebook.com https://twitter.com 'unsafe-eval' https://*.twimg.com https://api.twitter.com https://analytics.twitter.com https://publish.twitter.com https://ton.twitter.com 'nonce-RLwv/D7HBBPmdFZ1PlyLpA==' https://syndication.twitter.com https://www.google.com https://t.tellapart.com https://platform.twitter.com https://www.google-analytics.com 'self'; frame-ancestors 'self'; font-src https://twitter.com https://*.twimg.com data: https://ton.twitter.com https://fonts.gstatic.com https://maxcdn.bootstrapcdn.com https://netdna.bootstrapcdn.com 'self'; media-src https://twitter.com https://*.twimg.com https://ton.twitter.com blob: 'self'; connect-src https://graph.facebook.com https://*.giphy.com https://pay.twitter.com https://analytics.twitter.com https://media.riffsy.com https://upload.twitter.com https://api.mapbox.com 'self'; style-src https://fonts.googleapis.com https://twitter.com https://*.twimg.com https://translate.googleapis.com https://ton.twitter.com 'unsafe-inline' https://platform.twitter.com https://maxcdn.bootstrapcdn.com https://netdna.bootstrapcdn.com 'self'; object-src

	https://twitter.com https://pbs.twimg.com; default-src 'self'; frame-src https://staticxx.facebook.com https://twitter.com https://*.twimg.com https://player.vimeo.com https://pay.twitter.com https://www.facebook.com https://ton.twitter.com https://syndication.twitter.com https://vine.co twitter: https://www.youtube.com https://platform.twitter.com https://upload.twitter.com https://s-static.ak.facebook.com 'self' https://donate.twitter.com; img-src https://graph.facebook.com https://*.giphy.com https://twitter.com https://*.twimg.com data: https://fbcdn-profile-a.akamaihd.net https://www.facebook.com https://ton.twitter.com https://*.fbcdn.net https://syndication.twitter.com https://media.riffsy.com https://www.google.com https://stats.g.doubleclick.net https://*.tiles.mapbox.com https://www.google-analytics.com blob: 'self'; report-uri https://twitter.com/i/csp_report?a=NVQWGYLXFVZXO2LGOQ%3D%3D%3D%3D%3D%3D%3D&ro=false;
facebook.com	default-src * data: blob:;script-src *.facebook.com *.fbcdn.net *.facebook.net *.google-analytics.com *.virtualearth.net *.google.com 127.0.0.1:* *.spotilocal.com:* 'unsafe-inline' 'unsafe-eval' fbstatic-a.akamaihd.net fbcdn-static-b-a.akamaihd.net *.atlassolutions.com blob: chrome-extension://lifbcibllhkdhoafpjfnlhfpfgnpldfl;style-src * 'unsafe-inline' data:;connect-src *.facebook.com *.fbcdn.net *.facebook.net *.spotilocal.com:* *.akamaihd.net wss://*.facebook.com:* https://fb.scanandcleanlocal.com:* *.atlassolutions.com attachment.fbsbx.com ws://localhost:* blob: 127.0.0.1:*;
pinterest.com	default-src 'self' https://*.pinterest.com https://*.pinimg.com *.pinterest.com *.pinimg.com *.google.com connect.facebook.net *.google-analytics.com https://*.googleapis.com *.gstatic.com https://*.facebook.com *.facebook.com www.googleadservices.com googleads.g.doubleclick.net platform.twitter.com *.tiles.mapbox.com *.online-metrix.net *.bnc.lt bnc.lt *.yozio.com 'unsafe-inline' 'unsafe-eval'; frame-src *; img-src * data:; report-uri /_/_/csp_report/
blogger.com	script-src 'self' *.google.com *.google-analytics.com 'unsafe-inline' 'unsafe-eval' *.gstatic.com

	*.googlesyndication.com *.blogger.com *.googleapis.com uds.googleusercontent.com https://s.ytimg.com www-onepick-opensocial.googleusercontent.com www-bloggervideo-opensocial.googleusercontent.com www-blogger-opensocial.googleusercontent.com https://www.blogblog.com; report-uri /cspreport
tumblr.com	script-src 'self' https://secure.assets.tumblr.com https://assets.tumblr.com https://sb.scorecardresearch.com https://*.google-analytics.com https://*.cedexis.com https://*.cedexis-test.com 'unsafe-eval' 'nonce-1FXyNDWQiUf6Z9Xdbq2YykagZ8' https://www.google.com/recaptcha/api.js https://www.gstatic.com; report-uri https://cspreports.srvcs.tumblr.com/;
imdb.com	frame-ancestors 'self' imdb.com *.imdb.com *.media-imdb.com withoutabox.com *.withoutabox.com amazon.com *.amazon.com amazon.co.uk *.amazon.co.uk amazon.de *.amazon.de translate.google.com images.google.com www.google.com www.google.co.uk search.aol.com bing.com www.bing.com
vine.co	default-src https: data: vine:;img-src 'self' data: https://vine.co https://vines.s3.amazonaws.com https://*.cdn.vine.co https://media.vineapp.com https://t.co https://analytics.twitter.com https://ssl.google-analytics.com https://stats.g.doubleclick.net https://twemoji.maxcdn.com;script-src 'self' 'unsafe-inline' 'unsafe-eval' https://vine.co https://*.twitter.com https://vines.s3.amazonaws.com https://*.cdn.vine.co https://platform.vine.co https://stats.g.doubleclick.net https://ssl.google-analytics.com https://ajax.googleapis.com https://connect.facebook.net;style-src 'self' 'unsafe-inline' https://vine.co https://vines.s3.amazonaws.com https://*.cdn.vine.co;media-src 'self' blob: https://vine.co https://vines.s3.amazonaws.com https://*.cdn.vine.co https://*.vncdn.co https://media.vineapp.com;object-src 'self' blob: https://vine.co https://vine.co https://vines.s3.amazonaws.com https://*.cdn.vine.co https://media.vineapp.com;connect-src 'self' https://vine.co https://vines.s3.amazonaws.com https://*.cdn.vine.co https://media.vineapp.com https://graph.facebook.com;font-src 'self' https://vine.co https://vines.s3.amazonaws.com

	https://*.cdn.vine.co;report-uri https://twitter.com/i/csp_report?a=OZUW4ZI=&ro=false
usaa.com	frame-ancestors 'self'
flickr.com	default-src 'unsafe-inline' https://*.flickr.com https://*.flickr.net https://*.yimg.com https://bs.serving-sys.com https://*.braintreegateway.com https://*.kaptcha.com https://*.paypal.com https://*.conviva.com http://api.flickr.com https://*.pinterest.com; img-src data: blob: https://*.flickr.com https://*.flickr.net http://*.flickr.net https://*.staticflickr.com https://*.yimg.com https://*.yahoo.com https://*.cedexis.com https://*.cedexis-test.com https://*.cedexis-radar.net https://sb.scorecardresearch.com https://image.maps.api.here.com https://csync.yahooapis.com https://*.paypal.com https://*.pinterest.com; script-src 'unsafe-eval' 'unsafe-inline' https://*.flickr.com http://*.flickr.net https://*.flickr.net https://*.yimg.com https://*.analytics.yahoo.com https://y-flickr.yahoo.com https://yep.video.yahoo.com https://video.media.yql.yahoo.com https://*.yahooapis.com https://fc.yahoo.com https://*.braintreegateway.com https://*.paypalobjects.com https://*.cedexis.com https://*.cedexis-radar.net https://*.cedexis-test.com; connect-src https://*.flickr.com https://*.flickr.net http://*.flickr.net https://*.yimg.com https://geo.query.yahoo.com https://*.yahooapis.com https://*.conviva.com http://api.flickr.com https://*.pinterest.com http://*.yahoo.com https://*.cedexis.com https://*.cedexis-radar.net https://*.cedexis-test.com; report-uri https://csp.flickr.com/beacon/csp?src=adsecflickr;
dropbox.com	img-src https://* data: blob: ; connect-src https://* ws://127.0.0.1:*/ws ; media-src https://* ; object-src https://cf.dropboxstatic.com/static/ https://www.dropboxstatic.com/static/ 'self' https://flash.dropboxstatic.com https://swf.dropboxstatic.com https://dbxlocal.dropboxstatic.com ; default-src 'none' ; font-src https://* data: ; frame-src https://* carousel://* dbapi-6://* dbapi-7://* dbapi-8://* itms-apps://* itms-appss://* ; style-src https://* 'unsafe-inline' 'unsafe-eval' ; script-src https://ajax.googleapis.com/ajax/libs/jquery/

	'unsafe-eval' https://www.dropbox.com/static/ https://cf.dropboxstatic.com/static/javascript/ https://www.dropboxstatic.com/static/javascript/ https://cf.dropboxstatic.com/static/api/ https://www.dropboxstatic.com/static/api/ https://www.google.com/recaptcha/api/ 'unsafe-inline' 'nonce-uKUyFkBYowwEBxoXxesw' ;
blogspot.com	script-src 'self' *.google.com *.google-analytics.com 'unsafe-inline' 'unsafe-eval' *.gstatic.com *.googlesyndication.com *.blogger.com *.googleapis.com uds.googleusercontent.com https://s.ytimg.com www-onepick-opensocial.googleusercontent.com www-bloggervideo-opensocial.googleusercontent.com www-blogger-opensocial.googleusercontent.com https://www.blogblog.com; report-uri /cspreport
buzzfeed.com	upgrade-insecure-requests
cnn.com	default-src 'self' http://*.cnn.com:* https://*.cnn.com:* *.cnn.net:* *.turner.com:* *.ugdturner.com:* *.vgtf.net:*; script-src 'unsafe-inline' 'unsafe-eval' 'self' *; style-src 'unsafe-inline' 'self' *; frame-src 'self' *; object-src 'self' *; img-src 'self' * data: blob:; media-src 'self' *; font-src 'self' *; connect-src 'self' *;
rottentomatoes.com	frame-ancestors 'self' rottentomatoes.com *.rottentomatoes.com ;

1.4. Przykładowa implementacja

Wersja demonstracyjna zrealizowana została na platformie CentOS 6.6, z wykorzystaniem serwera HTTP Nginx 1.8.1, do którego konfiguracji dodano obsługę nagłówka "Content-Security-Policy". Na demonstracyjnej stronie WWW znajdują się dwa obrazki i dwa pola tekstowe. Obrazek "LOCAL" przechowywany jest na localhost, natomiast "REMOTE" na dysku internetowym Dropbox z publicznym URL. Za wpisywanie tekstu do pola z napisem "LOCAL" jest odpowiedzialny skrypt JavaScript przechowywany na localhost, natomiast do pola "REMOTE" na dysku internetowym Dropbox z publicznym URL. W zależności od ustawień CSP, czyli określenia białej listy źródeł strony, możliwe jest określenie zaufanego pochodzenia źródeł.

2. HTTP Strict Transport Security

HSTS jest standardem, który umożliwia wymuszenie komunikacji *HTTPS* między stroną internetową a przeglądarką. Znajduje zastosowanie w przypadku, gdy strona internetowa akceptuje połączenie *HTTP*, przekierowując użytkownika na stronę, używając *HTTPS*. Stwarza to możliwość przeprowadzenia ataku *man-in-the-middle*, gdyż na początku nawiązywane jest połączenie nieszyfrowane, przez co ruch może zostać przechwycony i skierowany na złośliwe treści, zamiast na wcześniej wybrany adres, ale po *HTTPS*. Dzięki *HTTP Strict Transport Security*, zapytania *HTTP* są konwertowane na *HTTPS*, w rezultacie czego, istnieje pewność, że cały ruch korzysta z *HTTPS*, uniemożliwiając tym samym ataki typu *man-in-the-middle*.

Włączenie tej funkcjonalności zamyka się jedynie w zdefiniowaniu nagłówka HTTPS o nazwie *Strict-Transport-Security*, lecz należy pamiętać, że nagłówek ten ustawiany jest tylko dla komunikacji *HTTPS*, przez co pojawia się problem, iż tak czy inaczej, pierwsze połączenie przeglądarki ze stroną internetową będzie nawiązane po *HTTP*. Rozwiązaniem tego problemu jest jeden z atrybutów nagłówka, którym jest *preload*, dzięki któremu możliwe jest ominięcie pierwszego połączenia po *HTTP* i zamienienie go na *HTTPS*, zanim strona zostanie wyświetlona pierwszy raz. Oprócz tego, istnieją jeszcze dwa atrybuty nagłówka *Strict-Transport-Security*: *max-age*, czyli jak długo przeglądarka ma pamiętać o wymogu i *includeSubdomains*, czyli zastosowanie wymogu również dla domen niższego poziomu. Poniżej przedstawiono przykładowe wystąpienie nagłówka:

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

Standard ten jest obsługiwany przez przeglądarki: *Firefox* od wersji 4, *Chrome* od wersji 4, *Opera* od wersji 12, *Safari* od wersji 7, *Spartan* i *Internet Explorer* w wersji 11.

3. HTTP Public Key Pinning

Zapewnienie autentyczności komunikacji w HTTPS realizowane jest dzięki standardowi X.509, który definiuje infrastrukturę klucza publicznego, umożliwiając tym samym potwierdzenie, czy host, z którym klient próbuje się połączyć, jest prawdziwy. Potwierdzenie autentyczności hosta odbywa się przy użyciu szeregu certyfikatów:

- głównego (*root certificate*), który należy do głównego urzędu certyfikującego (*root CA*) i jest samopodpisany (*self-signed*)
- pośredniego (*intermediate certificate*), dzięki któremu główne urzędy certyfikujące mogą uprawniać inne podmioty do działania w ich imieniu
- domeny (*domain certificate*).

Sprawdzenie poprawności łańcucha certyfikatów odbywa się poprzez walidację podpisów cyfrowych każdego z nich. Jeśli walidacja się powiedzie, przeglądarka bądź system operacyjny uznają, że połączenie jest zaufane. Niestety, główne oraz pośrednie centra certyfikacji nie mają wyszczególnionych konkretnych domen, do których mogą wystawiać certyfikaty. Oznacza to, że każdy z tych podmiotów może wystawić zaufany certyfikat dla dowolnej domeny. Głównym urzędom certyfikacji trudno utrzymać kontrolę nad wszystkimi z nich w wyniku czego dochodzi do nadużyć. W celu rozwiązania problemu, w przeglądarkach Chrome oraz Firefox zaczęto stosować przypinanie certyfikatów (*certificate pinning*). Mechanizm ten pozwala określić, które urzędy certyfikacji są uprawnione do wystawienia certyfikatu dla danej domeny. Na przykład Google może zdefiniować, że wszystkie certyfikaty dla ich domen muszą zostać podpisane przez urząd certyfikacji Google Internet Authority G2. Gdy ten warunek nie zostanie spełniony, przeglądarka wyświetli ostrzeżenie.

Mechanizm przypinania certyfikatów w przeglądarce Chrome jest stosowany już od 2011 roku. Dotychczas jednak polegał on na tym, że w przeglądarce zdefiniowano listę konkretnych domen oraz przypiętych certyfikatów. W kwietniu 2015 r. została opublikowana finalna wersja RFC 7469, w którym zdefiniowano nagłówek odpowiedzi Public-Key-Pins, dzięki któremu właściciel dowolnej witryny internetowej może skorzystać z przypinania certyfikatów.

Struktura nagłówka i opis pól:

Public-Key-Pins:
pin-sha256="public-key-sha256-base64"; max-age=expire-time; includeSubdomains; report-uri="report-uri"

pin-sha256 – pole wymagane, odcisk palca jednego z certyfikatów z łańcucha, wartość hasha musi być enkodowana w base64;
max-age – pole wymagane, czas ważności danego przypięcia w sekundach;
includeSubDomains – pole opcjonalne, dzięki użyciu tej dyrektywy przypinanie działa także dla wszystkich poddomen;

report-uri – pole opcjonalne, pozwala zdefiniować adres, pod który przeglądarka zgłosi naruszenie certyfikatu;

4. HTTP Header X-Frame-Options

Nagłówek odpowiedzi HTTP o nazwie X-Frame-Options służy do wskazania przeglądarce czy powinna być w stanie wyświetlać zawartość strony w ramce (elementy <frame> i <iframe>). Witryny mogą korzystać z tego mechanizmu do ochrony przed atakami typu clickjacking, zyskując pewność, że ich zawartość nie będzie umieszczona wewnątrz innych witryn.

Nagłówek X-Frame-Options może przyjmować 3 możliwe wartości:

- DENY - strona nie może być wyświetlana w ramce bez względu na adres witryny, która próbuje tego dokonać.
- SAMEORIGIN - strona może być wyświetlana w ramce pod warunkiem, że pochodzi z tego samego miejsca co strona zawierająca ramkę.
- ALLOW-FROM *uri* - strona może być wyświetlana w ramce wyłącznie wtedy, gdy wczytująca ją witryna załadowana została z podanego miejsca.

Użycie DENY oznacza zatem zablokowanie możliwości ładowania strony w ramce umieszczonej w innych witrynach, ale również w ramce, która znajduje się w tej samej witrynie, natomiast wartość SAMEORIGIN pozwoli na ładowanie strony w ramce, dopóki witryna z ramką jest tą samą, która serwuje zawartość strony włączanej w ramce.

Gdy dokonywana jest próba wczytania zawartości do ramki, lecz pojawia się odmowa wykonania działania spowodowana ustawieniem wartości nagłówka X-Frame-Options, obecna wersja przeglądarki Firefox wyświetla w ramce zawartość pochodzącą z lokalizacji specjalnej *about:blank*. W przyszłych wydaniach planuje się dodanie opcji wyświetlania odpowiedniego komunikatu w miejsce pustej strony.

Obsługa DENY i SAMEORIGIN jest zaimplementowana na wszystkich najpopularniejszych przeglądarkach (Chrome, Firefox, IE, Opera, Safari), natomiast obsługa ALLOW-FROM jest obecna jedynie w przeglądarce IE i Firefox (z błędami).

Ograniczenia:

- konieczność definiowania nagłówka na każdej stronie osobno;
- problemy ze stronami wielodomenowymi, brak możliwości dostarczenia listy z akceptowanymi domenami;
- bardzo słabe wsparcie przegądarek dla ALLOW-FROM

- brak wsparcia dla stosowania wielu opcji jednocześnie
- zagnieżdżone ramki nie współpracują z SAMEORIGIN i ALLOW-FROM

5. Filtr XSS

Ataki XSS są aktualnie jednym z powszechniejszych zagrożeń w sieci. Ich celem jest wykorzystanie luk czy słabych punktów stron internetowych, które odwiedzają użytkownicy. Działanie tych ataków opiera się najczęściej na "wstrzykiwaniu" złośliwych treści w oryginalną stronę internetową, przez co są one trudne do wykrycia. Dla przykładu, mogą one posłużyć do zbierania kolejno naciskanych przez użytkownika klawiszy (keylogger), przez co wykradzione mogą zostać jego dane. Filtr XSS jest w stanie wykryć tego typu ataki i je zablokować.

W przeglądarkach Internet Explorer od wersji 8 oraz Google Chrome zaimplementowano filtr mający na celu ochronę przed atakami XSS typu “reflected”. Filtr w IE 8 porównuje podane parametry żądania, jakie przeglądarka wysyła do serwera, do zbioru wyrażeń regularnych w celu poszukiwania prób ataku XSS. Poprzez dopasowywanie początku każdego tagu w skrypcie w parametrach żądania, filtr XSS wyłącza wszystkie skrypty “inline” na całej stronie. Możliwe jest także wyłączeni skryptów zewnętrznych.

6. X-Content-Type-Options: nosniff

Wprowadzono nowy nagłówek X-Content-Type-Options z jedną zdefiniowaną wartością “nosniff”. Rozwiązanie to chronić ma przeglądarki Internet Explorer i Google Chrome przed atakami typu MIME-sniffing. Ponadto zostało zaaplikowane w przeglądarce Chrome również podczas pobierania nowych rozszerzeń.

7. Literatura

1. https://www.w3.org/TR/2012/CR-CSP-20121115/
2. http://www.w3.org/TR/CSP2/
3. http://caniuse.com/#search=csp
4. http://www.html5rocks.com/en/tutorials/security/content-security-policy/
5. https://mikewest.org/2013/09/frontend-security-frontendconf-2013
6. https://www.quantcast.com/top-sites/PL
7. https://www.quantcast.com/top-sites/GB
8. https://www.quantcast.com/top-sites/US
9. http://lollyrock.com/articles/content-security-policy/
10. http://windows.microsoft.com/en-us/internet-explorer/products/ie-9/features/cross-site-scripting-filter
11. https://developer.mozilla.org/en-US/docs/Web/Security/HTTP_strict_transport_security
12. https://en.wikipedia.org/wiki/HTTP_Strict_Transport_Security
13. http://sekurak.pl/mechanizm-http-public-key-pinning/
14. https://kryptosfera.pl/post/http-public-key-pinning/
15. https://www.owasp.org/index.php/Clickjacking_Defense_Cheat_Sheet
16. https://developer.mozilla.org/pl/docs/HTTP/X-Frame-Options
17. https://blogs.msdn.microsoft.com/ie/2008/09/02/ie8-security-part-vi-beta-2-update/