Praktyka

Problemy z uwierzytelnianiem HTTP

Emilio Casbas

stopień trudności

Protokół HTTP, oferuje nam mechanizm uwierzytelniania żądanie-odpowiedź, który może zostać użyty przez serwer sieciowy lub serwer proxy, do umożliwiania lub odmawiania dostępu do zasobów sieciowych.

Obecnie w sieci dokonują się miliony transakcji oraz udostępniane są dane prywatne oraz poufne. Sieć to wszystko umożliwia, ale konieczne jest także zachowania bezpieczeństwa, musimy wiedzieć kto ma dostęp do naszych wrażliwych danych i kto może realizować działania uprzywilejowane.

Musimy wiedzieć, że nieupoważnieni użytkownicy, nie mogą obejrzeć dokumentów, do których nie mają dostępu. Serwery muszą w jakiś sposób dowiedzieć się, kim jest każdy użytkownik i na tej podstawie zdecydować jaki rodzaj działań mogą podjąć.

Uwierzytelnianie to technika identyfikacji oparta na znajomości, to znaczy na czymś, co zna użytkownik, jak na przykład haśle lub numerze PIN. HTTP dostarcza naturalną funkcjonalność do uwierzytelniania HTTP. W rzeczywistości HTTP definiuje dwa oficjalne protokoły uwierzytelniania: uwierzytelnianie podstawowe oraz uwierzytelnianie digest. Tutaj skoncentruję się szczególnie na metodzie uwierzytelniania podstawowego, która jest szerzej wykorzystywana przez klientów i serwery sieciowe, a także mniej bezpieczna.

Obszary stosowania tej metody uwierzytelniania:

- Serwery sieciowe w internecie - tu znaleźlibyśmy się w najbardziej powszechnej sytuacji. Użytkownik z domu, za pośrednictwem zwykłego połączenia lub z kafejki internetowej wchodzi na serwer sieciowy na którym skonfigurowano uwierzytelnianie HTTP, aby uzyskać dostęp do pewnych jego obszarów. Przeglądając choćby kilka korporacyjnych stron internetowych, możemy zaobserwować wielką ilość stron, które wykorzystują ten rodzaj uwierzytelniania, aby przejść do zastrzeżonych części strony.

Z artykułu dowiesz się...

- Różne zakresy uwierzytelniania HTTP
- Różnice w zatwierdzaniu HTTP w różnych obszarach
- Przykłady praktyczne konwersacji HTTP
- Słabości uwierzytelniania
- Rozwiązania lub alternatywy

Powinieneś wiedzieć...

- Model OSI
- Znajomość protokołu HTTP

Krótkie wprowadzenie do uwierzytelniania

Uwierzytelnianie digest:

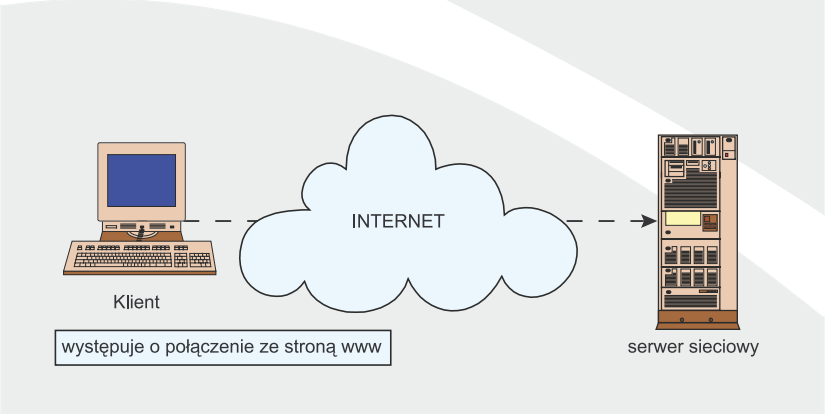
- Celem uwierzytelniania digest, jest nie wysyłanie nigdy hasła przez sieć", w tym celu wysyła na serwer "podsumowanie" lub "ślad" hasła w sposób nieodwracalny.
- Uwierzytelnianie digest zostało rozwinięte w sposób zgodny i jako bezpieczniejsza alternatywa w stosunku do uwierzytelniania podstawowego, ale nie jest to jeden z protokołów zwanych bezpiecznymi, porównywanymi do tych, które stosują mechanizmy klucza publicznego (SSL) lub mechanizmy wymiany biletów (kerberos). Uwierzytelnianie digest nie posiada silnego uwierzytelniania, ani nie oferuje ochrony poufności poza ochroną hasła, pozostała część żądania i odpowiedzi przechodzą jako zwykły tekst.

Uwierzytelnianie podstawowe:

- Uwierzytelnianie podstawowe jest jednym z częściej używanych protokołów uwierzytelniania HTTP. Wdraża je większość klientów i serwerów sieciowych. Najlepiej jest najpierw zobaczyć opis rysunkowy, żeby zorientować się, czym ono jest.
- Poniżej szczegółowo wytłumaczymy wcześniejsze kroki uwierzytelniania HTTP.
- Użytkownik ubiega się o jakiś zasób sieciowy (na przykład obrazek)
- Serwer sprawdza, że jest to zasób chroniony i wysyła klientowi żądanie o hasło z nagłówkiem HTTP "Authorization Required" i kodem 401.
- Przeglądarka użytkownika otrzymuje kod i nagłówek uwierzytelnienia i pokazuje dialog użytkownik/hasło. Kiedy użytkownik wprowadza dane, przeglądarka przeprowadza kodowanie na base64 z wprowadzonymi danymi i przesyła je na serwer w nagłówku użytkownika „Authorization".
- Serwer dekoduje nazwę użytkownika i hasło oraz sprawdza, że ma on dostęp do chronionego zasobu.

Jak można stwierdzić, uwierzytelnienie podstawowe przesyła parę użytkownik:hasło w postaci nie szyfrowanej z przeglądarki na serwer i jako takie, nie powinno być używane dla wrażliwych loginów, o ile nie pracuje się w środowisku szyfrowanym, takim jak SSL.

- Serwery sieciowe w intranecie- w tym przypadku obszar zastosowania jest mniejszy, jako że jest on ograniczony jedynie do intranetu firmowego, ale problemy związane z tym rodzajem uwierzytelniania są takie same jak omawiane wcześniej, jakikolwiek zasób dostępny wewnątrz sieci będzie tak samo narażony.
- Serwery proxy w internecie - może także zajść przypadek, gdy na przykład, żeby surfować po niektórych zasobach określonej sieci, można to robić tylko przez serwer proxy tej instytucji lub aby kontrolować jakikolwiek rodzaj dostępu. Dlatego uwierzytelnianie HTTP może także być wdrożone w proxy, aby wszystkie dane szły także przez internet.
- Serwery proxy w intranecie - bardzo popularna jest również sytuacja, gdy wewnątrz sieci korporacyjnych jedyną możliwością dostępu do internetu jest dostęp przez serwer proxy, co ma na celu całkowite kontrolowanie użytkowania internetu. Dlatego też ustawienie serwera proxy, aby żądał potwierdzania w celu kontroli dostępu użytkowników do sieci, jest całkowicie naturalne w tym przypadku. Z reguły ten rodzaj potwierdzenia będzie zintegrowany z innymi mechanizmami istniejącymi w tej sieci intranetowej. Pogarszając problem Single Sign On, którym zajmiemy się później.



Rysunek 1. *Serwery sieciowe w Internecie*

Przykład praktyczny

Od tego momentu wiemy już, czym jest uwierzytelnianie HTTP i jakie są kolejne kroki jego działania, w tej chwili przyjrzymy się tym etapom bardziej wnikliwie i w sposób praktyczny, przy użyciu naszej przeglądarki internetowej mozilla. Będzie nam do tego potrzebne rozszerzenie aby przechwycić nagłówki HTTP, które realizujemy.

Potrzebne nam rozszerzenie nazywa się *livehttpheaders* i można je ściągnąć z *http://livehttpheaders.mozdev.org/*. Instalujemy je wykonując kroki opisane na stronie, a następnie w mozilli pojawi się opcja Live HTTP Headers, widoczna na Rysunku 6.

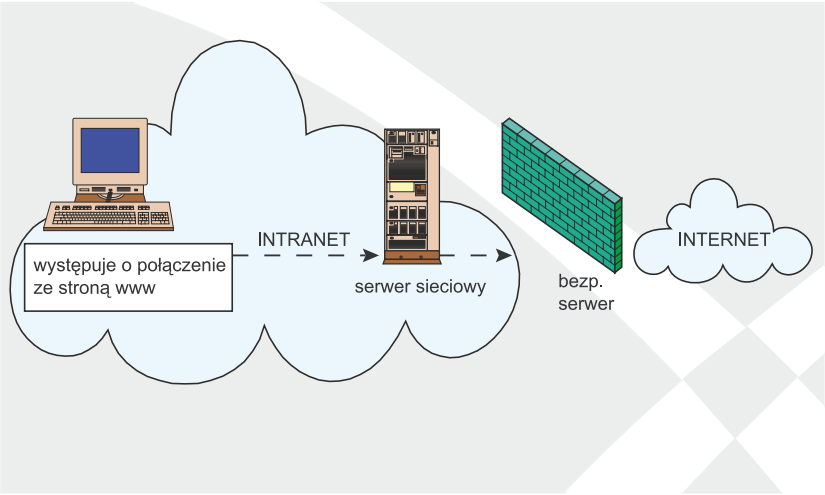
Klikamy na tę nową opcję a wtedy pojawi się następująca ramka widoczna na Rysunku 2.

Od tego momentu będziemy otrzymywać wszystkie informacje o nagłówkach HTTP ze wszystkich stron po których surfujemy, w ten sposób zdołaliśmy przechwycić nagłówki, które wyjaśnione są poniżej.

Nagłówki i wyjaśnienie

Klient wysyła standardowe żądanie HTTP prosząc o dany zasób.

```
GET /doc/ecasbas/ HTTP/1.1\rn
Host: www.prueba.es\rn
User-Agent: Mozilla/5.0
 (Windows; U;
```



Rysunek 2. *Serwery sieciowe w Intranecie*

```
 Windows NT 5.1; en-US; rv:1.7.12)
Accept: text/xml,text/html;q=0.9,
 text/plain;q=0.8,
 image/png,*/*;q=0.1\rn
Accept-Language: en-us,en;
 q=0.7,es;q=0.3\rn
Accept-Encoding: gzip,deflate\rn
Accept-Charset: ISO-8859-1,
 utf-8;q=0.7,*;q=0.7\rn
Keep-Alive: 300\rn
Connection: keep-alive\rn
```

Serwer czyta swoje archiwum ustawień i określa, że żądany zasób jest chroniony hasłem. Serwer może udostępnić go tylko znanym użytkownikom.Wtedy serwer odpowiada klientowi żądaniem autoryzacji oznaczając to kodem HTTP 401.

```
HTTP/1.0 401 Unauthorized\rn
Date:
 Mon, 16 Jan 2006 11:17:51 GMT\rn
Server: Apache/2.0.55 (Unix)
 mod_ssl/2.0.55 OpenSSL/0.9.7g
 PHP/5.1.1\rn
WWW-Authenticate:
 Basic realm="ByPassword"\rn
Accept-Ranges: bytes\rn
Content-Length: 3174\rn
Content-Type: text/html\rn
X-Cache:
 MISS from www.prueba.es\rn
Connection: keep-alive\rn
```

Przeglądarka klienta interpretuje ten kod HTTP 401 jako żądanie do uwierzytelnienia, i wtedy przeglądarka pokazuje prompt użytkownik:password pokazując nazwę hosta i realm. Ilustruje to rysunek 3.

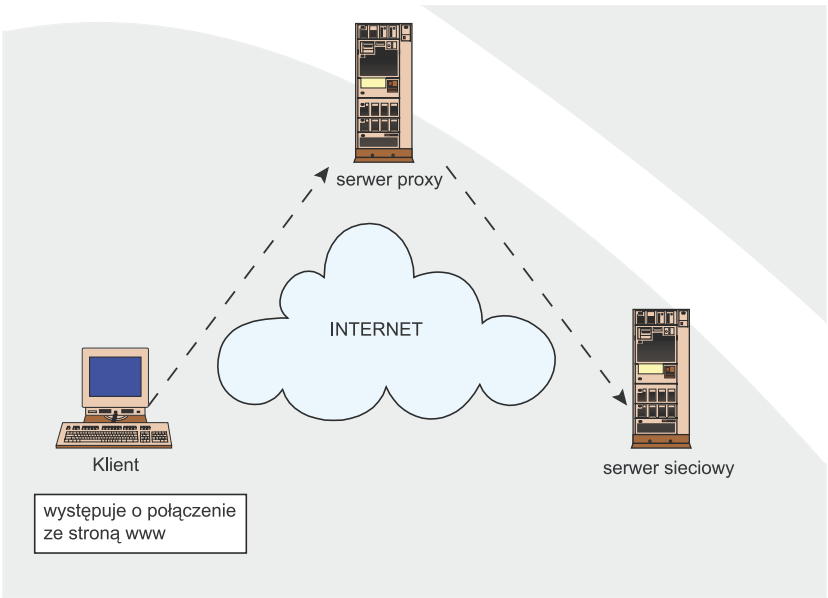
Klient odeśle żądanie z wprowadzonymi danymi o użytkowniku/haśle

```
GET /doc/ecasbas/ HTTP/1.1\rn
Host: www.unav.es\rn
User-Agent: Mozilla/5.0
 (Windows; U; Windows NT 5.1;
 en-US; rv:1.7.12)
Accept: text/tml+xml,text/html,
 image/jpeg,image/gif;
 q=0.2,*/*;q=0.1\rn
Accept-Language:
 en-us,en;q=0.7,es;q=0.3\rn
Accept-Encoding: gzip,deflate\rn
Accept-Charset:
 ISO-8859-1,utf-8;q=0.7,*;q=0.7\rn
Keep-Alive: 300\rn
Connection: keep-alive\rn
Authorization:
 Basic ZWNhc2JhczpwcnVlYmE=\rn
```

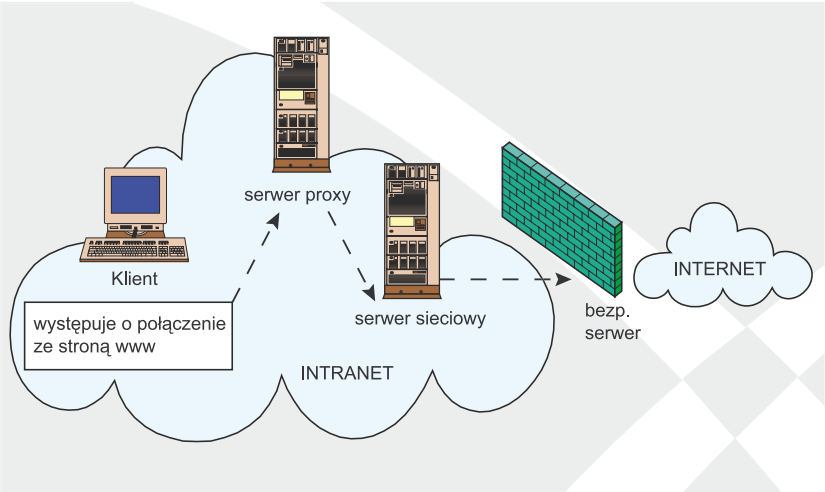
Następnie serwer porównuje informację od klienta ze swoją listą użytkowników/haseł. Jeżeli autoryzacja szwankuje, serwer znowu wyśle nagłówek wymaganego uwierzytelnienia HTTP 401. Jeżeli wprowadzone dane są prawidłowe, serwer pokarze żądany zasób. Po tym serwer przechodzi do żądanego zasobu.

```
HTTP/1.0 200 OK\rn
Date: Mon, 16 Jan 2006 11:17:58 GMT\rn
Server: Apache/2.0.55 (Unix)
 mod_ssl/2.0.55
 OpenSSL/0.9.7g PHP/5.1.1\rn
Last-Modified:
 Fri, 13 Jan 2006 10:31:02 GMT\rn
ETag: "125b019-5-f636a580"\rn
Accept-Ranges: bytes\rn
Content-Length: 5\rn
Content-Type: text/html\rn
X-Cache:
 MISS from www.prueba.es\rn
Connection: keep-alive\rn
```

We wcześniejszych polach widzieliśmy pola specjalne, które zostały dodane do różnych nagłówków HTTP. W etapie 3, kiedy serwer wysyła odpowiedź z nagłówkiem 401, zawarte



Rysunek 3. *Serwery proxy w Internecie*



Rysunek 4. *Serwery proxy w Intranecie*

jest specjalne pole.

```
WWW-Authenticate:
 Basic realm="ByPassword"\rn
```

Wartość “Basic” pokazuje, że prosimy browser o użycie uwierzytelnienia podstawowego. Informacja łańcucha “realm” jest łańcuchem arbitralnym wysłanym, aby pokazać użytkownikowi powiadomienie o rodzaju uwierzytelnienia. Rysunek 3 pokazuje, jak okno dialogowe mozilli prosi o uwierzytelnienie pokazując realm i hosta.

Użytkownik wypełnia formularz i wysyła go. Przeglądarka automatycznie odsyła prośbę. Jak pokazano wcześniej niektóre pola dodane zostały do standardowego żądania HTTP.

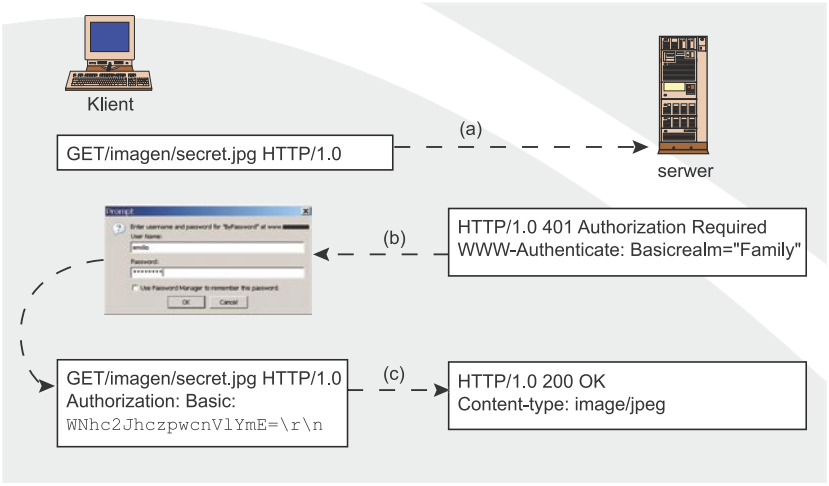
```
Authorization:
 Basic ZWNhc2JhczpwcnVlYmE=\rn
```

Tu właśnie przeglądarka internetowa wysyła informację o aktualnej autoryzacji na serwer. Widać, że pole “Authorization” złożone jest z dwóch wartości. Słowo “Basic” pokazuje, jak login zostaje wysłany w zgodzie z metodą uwierzytelnienia podstawowego. Blok danych, który po nim następuje to aktualny login wysłany przez przeglądarkę. Nasze dane dotyczące loginu nie pojawiają się bezpośrednio, ale nie jest to szyfrowanie, tylko kodowane w base 64. Podsumowując, kodowanie w base64 przedstawia arbitralne sekwencje oktetów w sposób niekoniecznie czytelny dla człowieka. Algorytmy kodowania i dekodowania są proste, ale zakodowane dane z reguły są o 33% większe, niż dane nie kodowane. Aby uzyskać więcej informacji o takim kodowaniu warto skorzystać z ramki *W sieci*.

Mając wcześniejszy kod oraz login w zwykłym tekście, można w trywialny sposób odkodować zapis do poniższego formatu użytkownik:hasło.

```
ZWNhc2JhczpwcnVlYmE=
 --> base64Decode() --> "ecasbas:próba"
```

Wdrożenie uwierzytelniania “digest” to dokładnie taki sam proces, jak przy wcześniejszym uwierzytelnianiu podstawowym, jedyna różnica to liczba argumentów dostarczonych przez przeglądarkę oraz format zwróconego loginu.



Rysunek 5. *Uwierzytelnianie podstawowe*

Listing 1. *Kod perl do dekodowania łańcucha w base64 jak wcześniej*

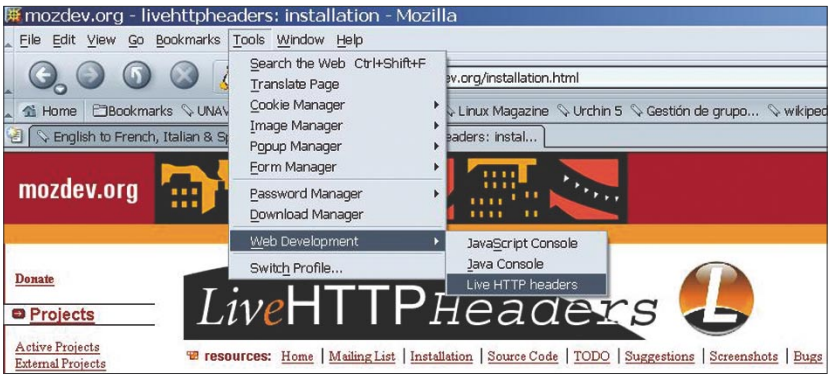
```
#!/usr/bin/perl
use MIME::Base64;
while (<>) {
  print MIME::Base64::deco-
de_base64($_);
}
```

Obydwa rodzaje uwierzytelnień “digest” i “basic” używane są przez klientów i serwery sieciowe, jednak nie należy ich używać jako stopnia ochrony informacji wrażliwych lub bezpiecznego dostępu. Bardzo powszechne jest stosowanie tej samej nazwy użytkownika i hasła do różnych serwisów, w tym przypadku należy pamiętać o tym, żeby zasoby, które chcemy chronić w ten sposób, nie były bardzo delikatne i żeby uwierzytelnienia nie funkcjonowały w innym użyciu jak np.: w poczcie lub dostępie do informacji prywatnych.

Uwierzytelnianie proxy

Wcześniejsze sekwencje odnoszą się do klienta proszącego serwer sieciowy o dostęp do chronionego zasobu. Ale można by to zastosować w przypadku, kiedy proxy wymaga potwierdzenia, aby uzyskać dostęp do zasobu. Przyjrzymy się również tej sytuacji oraz temu, jakie kody pokazują się, kiedy w grę wchodzi proxy.

Mając skonfigurowany w naszej



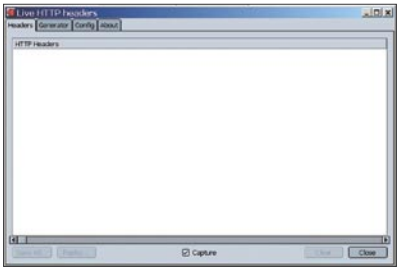
Rysunek 6. *Opcja Live HTTP Headers*

przeglądarce serwer proxy, składamy prośbę, aby móc surfować.

```
GET
 http://www.google.com/ HTTP/1.1\rn
Host: www.google.com\rn
User-Agent: Mozilla/5.0 (Windows; U;
 Windows NT 5.1; en-US; rv:1.7.12)
Accept: application/x-shockwave-flash,
text/xml,application/xml,*/*;q=0.1\rn
Accept-Language:
 en-us,en;q=0.7,es;q=0.3\rn
Accept-Encoding: gzip,deflate\rn
Accept-Charset:
 ISO-8859-1,utf-8;q=0.7,*;q=0.7\rn
Keep-Alive: 300\rn
Proxy-Connection: keep-alive\r\n
```

Proxy odpowiada nam wskazując, że musimy to potwierdzić, aby móc przeglądać internet.

```
HTTP/1.0 407
 Proxy Authentication Required\rn
Server: squid/2.5.STABLE12\rn
Mime-Version: 1.0\rn
Date: Mon, 16 Jan 2006 13:01:19 GMT\rn
Content-Type: text/html\rn
Content-Length: 3283\rn
Expires:
 Mon, 16 Jan 2006 13:01:19 GMT\rn
X-Squid-Error:
 ERR_CACHE_ACCESS_DENIED 0\rn
Proxy-Authenticate: Basic realm=
""Proxy Authentication (user/passwd)""\
                              rn
X-Cache: MISS from proxy.es\rn
Proxy-Connection: keep-alive\rn
```



Rysunek 7. *Ramka Live HTTP Headers*

Wtedy nasza przeglądarka interpretuje to jako żądanie/odpowiedź dla uwierzytelnienia podstawowego i pokazuje nam login, aby wprowadzić wymagane dane. Ilustruje to Rysunek 9.

Niektóre przeglądarki nie interpretują prawidłowo “realm” dlatego też w niektórych rzeczywiście będzie widać we wcześniej pokazanej ramce wiadomość “Proxy Authentication (user/passwd)”, ten akurat przypadek jest inny, ale będzie nam służył jako przykład.

Wprowadzamy nazwę użytkownika i hasło, a nasz klient wysyła następujące dane zwrotne do proxy.

```
GET
http://www.google.com/ HTTP/1.1\rn
Host: www.google.com\rn
User-Agent: Mozilla/5.0
 (Windows;
 U; Windows NT 5.1;
 en-US; rv:1.7.12)
Accept:
 application/x-shockwave-flash,
 text/xml,
 image/gif;q=0.2,*/*;q=0.1\rn
Accept-Language:
 en-us,en;q=0.7,es;q=0.3\rn
Accept-Encoding:
 gzip,deflate\rn
Accept-Charset:
 ISO-8859-1,utf-8;q=0.7,*;q=0.7\rn
Keep-Alive: 300\rn
Proxy-Connection:
 keep-alive\rn
Proxy-Authorization:
 Basic
 ZWNhc2Jhc0B1bmF2Lm
 VzOnBydWViYTAx\r\n
```

Serwer proxy wewnętrznie sprawdza, że rzeczywiście nazwa użytkownika i hasło są ważne i przyznaje nam dostęp do zasobu.

```
HTTP/1.0 200 OK\rn
Cache-Control: private\rn
Content-Type: text/html\rn
Content-Encoding: gzip\rn
Server: GWS/2.1\rn
Content-Length: 1408\rn
Date: Mon, 16 Jan 2006 13:05:40 GMT\rn
X-Cache: MISS from filter\rn
Proxy-Connection: keep-alive\r\n
```

Zamiast odpowiadać kodem 401 HTTP, w przypadku serwera proxy, pokazuje się kod 407 (wymagane uwierzytelnienie przez proxy) a do-

Rysunek 8. *Ramka Live HTTP Headers*

Podsumowanie Tabela *1. Uwierzytelnianie serwera sieciowego i uwierzytelnianie proxy.*

Serwer sieciowy	Serwer Proxy.
kod stanu bez autoryzacji: 401	kod stanu bez autoryzacji: 407
WWW-authenticate	Proxy-Authenticate
Authorization	Proxy-authorization
Authentication-Info	Proxy-Authentication-Info.

dawany nagłówek w przypadku serwera sieciowego WWW-authenticate, teraz dla proxy brzmi Proxy-Authenticate. A cały proces jest identyczny jak w przypadku dostępu do ograniczonego zasobu sieciowego, z wyłączeniem tych minimalnych różnic.

Ogólne spojrzenie na uwierzytelnianie HTTP

Schemat uwierzytelniania podstawowego nie jest bezpieczną metodą uwierzytelniania użytkowników. Nie ochrania w żaden sposób tożsamości użytkownika, która przekazywana jest poprzez sieć jako zwykły tekst. HTTP z drugiej strony nie unika użycia dodatkowych schematów uwierzytelnienia i mechanizmów szyfrowania, zastosowanych do zwiększenia lub polepszenia bezpieczeństwa uwierzytelnienia podstawowego.

Mimo słabości tego rodzaju uwierzytelnienia, jak mogliśmy zobaczyć wcześniej, metody tej używa się w różnych środowiskach, gdzie największym zagrożeniem jest istnienie Single Sign On, to znaczy twoje uwierzytelnienia służą ci do potwierdzenia dostępu do jakiegokolwiek zasobu w sieci. W ten sposób jest w zasadzie wszystko jedno, czy użyte zostaną mechanizmy bezpiecznego dostępu przy użyciu SSL, bezpieczne połączenia szyfrowane w sieci (IPSEC), programy z bezpiecznym logowaniem, itp, jeżeli tylko jeden z zasobów używa tej formy uwierzytelniania, mielibyśmy natychmiastowy dostęp do wszystkich dostępnych serwisów.

Idealnym polem do wdrażania i używania tego rodzaju uwierzytelniania byłby na przykład dostęp do statystyk użycia danego serwera lub dostęp do jakiegokolwiek zasobu, jeżeli sądzimy, że bezprawny dostęp do niego, nie niesie za sobą potencjalnego zagrożenia. W połączeniu z tym, uwierzytelnienia dostępu muszą zostać dostarczone przez admina strony lub przez program generujący. Nigdy nie może wybierać ich użytkownik, jako że znaleźlibyśmy się znowu z tym samym, wcześniejszym problemem. Ludzie nie mają zwyczaju używać różnych uwierzytelnień do różnych serwisów, ale wykorzystują je wielokrotnie.

Zakończenie

Uwierzytelnienie podstawowe HTTP jest proste i wygodne, ale nie jest to metoda bezpieczna. Należy jej używać w przypadkach, gdy dostęp do informacji ma mieć charakter prywatny, a nie absolutnie konieczny i tam, gdzie jego użycie nie może narazić bezpieczeństwa innych systemów.

Ludzie dążą do tego, żeby wykorzystywać tę samą nazwę użytkownika i hasło do różnych celów, przez co pomimo tego, że używane będą w środowiskach zaufanych i w przypadku dostępu do informacji nie wrażliwej, zawsze istniałoby ryzyko, że te same uwierzytelnienia dałyby nam dostęp do ważniejszych serwisów jak poczta elektroniczna, prywatne dokumenty czy bazy danych.

Mając sniffera sieciowego i kilka skryptów odpowiednich do interpretowania przechwyconego ruchu, w przeciągu kilku minut możliwe jest otrzymanie setek par “użytkownik/hasło” metodą opisaną powyżej. Przy uwierzytelnianiu HTTP, hasła podróżują po sieci jawnie, a w trakcie połączenia nagłówki z hasłami przeciętnie podróżują więcej niż raz. W trakcie trwania połączenia, wysyłane są ponownie w przypadku każdej kolejnej przeprowadzonej transakcji. Jest to związane z właściwością protokołu HTTP, który nie zachowuje stanu i konieczne jest przypomnienie danych, które dostarcza się, w trakcie każdego połączenia, które realizujowane jest z serwerem sieciowym lub proxy. Aby ulepszyć ten sposób uwierzytelniania lub zastąpić go innymi, bardziej bezpiecznymi metodami należałoby:

- połączyć go z SSL, aby wzmocnić bezpieczeństwo kodując wszystkie dane transmisji
- zastąpić uwierzytelnieniem digest
- użyć Kerberos
- Usunąć je z miejsc, gdzie nie jest potrzebne ●

Rysunek 9. *Ramka Live HTTP Headers*

W Sieci

- *http://livehttpheaders.mozdev.org/* – Plugin dla mozilli.
- *ftp://ftp.isi.edu/in-notes/rfc2617.txt* – Uwierzytelnianie HTTP: uwierzytelnianie dostępu Basic i digest
- *http://www.faqs.org/rfcs/rfc2045.html* – Kodowanie transferu zawartości w Base64
- *http://httpd.apache.org/docs/1.3/howto/auth.html* – Konfiguracja apache, aby żądał uwierzytelniania.
- *http://rfc.sunsite.dk/rfc/rfc2617.html* – RFC 2617