Digest Authentication

Basic Authentication راحت و منعطف است اما کاملاً ناامن است. نام‌های کاربری و گذرواژه‌ها به صورت واضح ارسال می‌شوند و هیچ تلاشی برای محافظت از پیام‌ها در برابر دستکاری وجود ندارد. تنها راه برای استفاده ایمن از Basic Authentication استفاده از آن در ارتباط با SSL است.

Digest Authentication به عنوان جایگزینی سازگار و امن تر برای احراز هویت اولیه توسعه داده شد. ما این فصل را به Digest Authentication اختصاص می‌دهیم. حتی اگر Digest Authentication هنوز به طور گسترده مورد استفاده قرار نگرفته است، مفاهیم هنوز برای هر کسی که تراکنش‌های ایمن را اجرا می‌کند مهم است.

The Improvements of Digest Authentication

Digest Authentication یک پروتکل احراز هویت HTTP جایگزین است که سعی می‌کند جدی‌ترین نقص‌های Basic Authentication را برطرف کند. به طور خاص، Digest Authentication:

- هرگز رمزهای عبور مخفی را در سراسر شبکه به صورت شفاف ارسال نمی‌کند.
- از گرفتن و پخش مجدد Authentication Handshake توسط مهاجمان جلوگیری می‌کند.
- به صورت اختیاری می‌تواند از دستکاری در محتوای پیام محافظت کند.
- در برابر چندین اشکال رایج حملات دیگر نیز موارد محافظتی را اعمال می‌کند.

توجه داشته باشید که Digest Authentication ایمن‌ترین پروتکل ممکن نیست. بسیاری از نیازها برای تراکنش‌های HTTP ایمن با Digest Authentication برآورده نمی‌شوند. برای این نیازها، امنیت لایه حمل و نقل (TLS) و HTTP امن (HTTPS) پروتکل‌های مناسب‌تری هستند.

با این حال، Digest Authentication به طور قابل توجهی قوی‌تر از Basic Authentication است، که برای جایگزینی طراحی شده است. Digest Authentication نیز قوی‌تر از بسیاری از طرح‌های محبوب ارائه شده برای سایر سرویس‌های اینترنتی است، مانند CRAM-MD5، که برای استفاده با LDAP، POP و IMAP پیشنهاد شده است.

تا به امروز، Digest Authentication به طور گسترده اجرا نشده است. با این حال، به دلیل خطرات امنیتی ذاتی Basic Authentication، معماران HTTP در RFC 2617 توصیه می‌کنند که «هر سرویسی که در حال حاضر از Basic استفاده می‌کند باید به محض عملی شدن به Digest تبدیل شود.».

هنوز مشخص نیست که این استاندارد چقدر موفق بوده است.

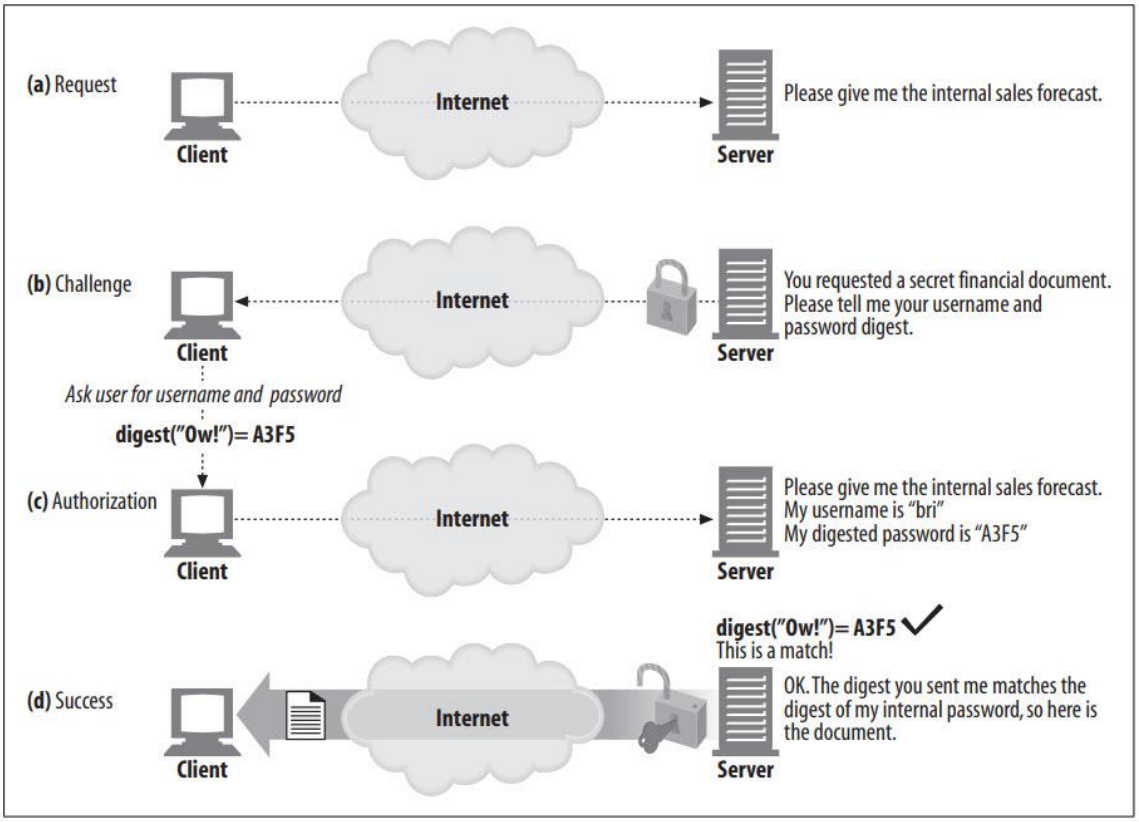
Using Digests to Keep Passwords Secret

شعار Digest Authentication "هرگز رمز عبور را در سراسر شبکه ارسال نکنید."

به جای ارسال رمز عبور، کلاینت یک "Fingerprint" یا "Digest" رمز عبور را ارسال می‌کند که درهم شکسته غیرقابل برگشت (Irreversible Scrambling) رمز عبور است. کلاینت و سرور هر دو رمز عبور مخفی را می‌دانند، بنابراین سرور می‌تواند تأیید کند که Digest تطابق درستی برای رمز عبور ارائه کرده است. با توجه به Digest، یک پسر بد راه آسانی برای یافتن رمز عبور آن ندارد، به جز اینکه همه رمزهای عبور در جهان را مرور کند و هر کدام را امتحان کند!

بیایید ببینیم این مورد چگونه کار می‌کند (این یک نسخه ساده شده است):

در شکل بخش a زیر، کلاینت یک سند محافظت شده را درخواست می‌کند.



در بخش b شکل، سرور از ارائه سند خودداری می‌کند تا زمانی که کلاینت هویت آن را با اثبات اینکه رمز عبور را می‌داند، احراز هویت کند. سرور چالشی را برای کلاینت ارسال می‌کند و نام کاربری و فرم Digest شده رمز عبور را می‌خواهد.

در بخش c شکل، کلاینت با ارسال Digest رمز عبور ثابت می‌کند که رمز عبور را می‌داند. سرور گذرواژه‌های همه کاربران را می‌داند، بنابراین می‌تواند با مقایسه Digest ارائه‌شده توسط کلاینت با Digest محاسبه‌شده داخلی سرور، تأیید کند که کاربر رمز عبور را می‌داند. طرف دیگر اگر رمز عبور را نمی‌دانست به راحتی نمی‌توانست Digest درستی را بسازد.

در بخش d شکل، سرور Digest ارائه شده توسط کلاینت را با Digest محاسبه شده داخلی سرور مقایسه می‌کند. اگر مطابقت داشته باشند، نشان می‌دهد که کلاینت رمز عبور را می‌داند (یا حدس واقعاً خوش شانسی انجام داده است!). تابع Digest را می‌توان طوری تنظیم کرد که ارقام زیادی تولید کند که حدس زدن تصادفی به طور مؤثر غیرممکن باشد. هنگامی که سرور مطابقت را تأیید می‌کند، سند به کلاینت ارائه می‌شود - همه این‌ها بدون ارسال رمز عبور از طریق شبکه.

One-Way Digests

Digest «تراکم مجموعه‌ای از اطلاعات» است. Digest ها به عنوان توابع یک طرفه عمل می‌کنند و معمولاً تعداد نامحدودی از مقادیر ورودی ممکن را به محدوده محدودی از تراکم تبدیل می‌کنند. دنباله‌ای از بایت‌ها، با هر طول، به یک خلاصه ۱۲۸ بیتی.

آنچه در مورد این Digest ها مهم است این است که اگر رمز عبور مخفی را ندانید، حدس زدن Digest درست برای ارسال به سرور برای شما بسیار سخت است. و به همین ترتیب، اگر Digest را داشته باشید، تشخیص اینکه کدام یک از بی‌نهایت مقادیر ورودی آن را ایجاد کرده‌اند، بسیار سخت خواهد بود.

۱۲۸ بیت خروجی MD5 اغلب به صورت ۳۲ کاراکتر هگزادسیمال نوشته می‌شود که هر کاراکتر نشان دهنده ۴ بیت است. جدول زیر چند نمونه از Digest های MD5 را نشان می‌دهد. توجه داشته باشید که چگونه MD5 ورودی‌های دلخواه را دریافت می‌کند و یک خروجی Digest با طول ثابت ایجاد می‌کند.

Input	MD5 digest
"Hi"	C1A5298F939E87E8F962A5EDFC206918
"bri:0w!"	BEAAA0E34EBDB072F8627C038AB211F8
"3.1415926535897"	475B977E19ECEE70835BC6DF46F4F6DE
"http://www.http-guide.com/index.htm"	C617C0C7D1D05F66F595E22A4B0EAAA5
"WE hold these Truths to be self-evident, that all Men are created equal, that they are endowed by their Creator with certain unalienable Rights, that among these are Life, Liberty and the Pursuit of Happiness—That to secure these Rights, Governments are instituted among Men, deriving their just Powers from the Consent of the Governed, that whenever any Form of Government becomes destructive of these Ends, it is the Right of the People to alter or to abolish it, and to institute new Government, laying its Foundation on such Principles, and organizing its Powers in such Form, as to them shall seem most likely to effect their Safety and Happiness."	66C4EF58DA7CB956BD04233FBB64E0A4

توابع Digest گاهی اوقات Checksum های رمزنگاری، توابع هش یک طرفه یا توابع Fingerprint نامیده می‌شوند.

Using Nonces to Prevent Replays

Digest یک طرفه ما را از ارسال رمزهای عبور به صورت شفاف نجات می‌دهد. ما فقط می‌توانیم Digest ای از رمز عبور را به جای آن ارسال کنیم و مطمئن باشید که هیچ طرف مخربی نمی‌تواند به راحتی رمز عبور اصلی را از Digest رمزگشایی کند.

متأسفانه، رمزهای عبور مبهم به تنهایی ما را از خطر نجات نمی‌دهد، زیرا یک نفوذگر می‌تواند Digest را ضبط کرده و بارها و بارها آن را برای سرور پخش کند، حتی اگر رمز عبور را نداند. Digest به اندازه رمز عبور خوب است.

برای جلوگیری از چنین حملات تکراری، سرور می‌تواند توکن خاصی به نام *nonce را برای کلاینت ارسال کند که اغلب (شاید در هر میلی‌ثانیه یا برای هر احراز هویت) تغییر می‌کند. کلاینت قبل از محاسبه Digest، این توکن nonce را به رمز عبور اضافه می‌کند.

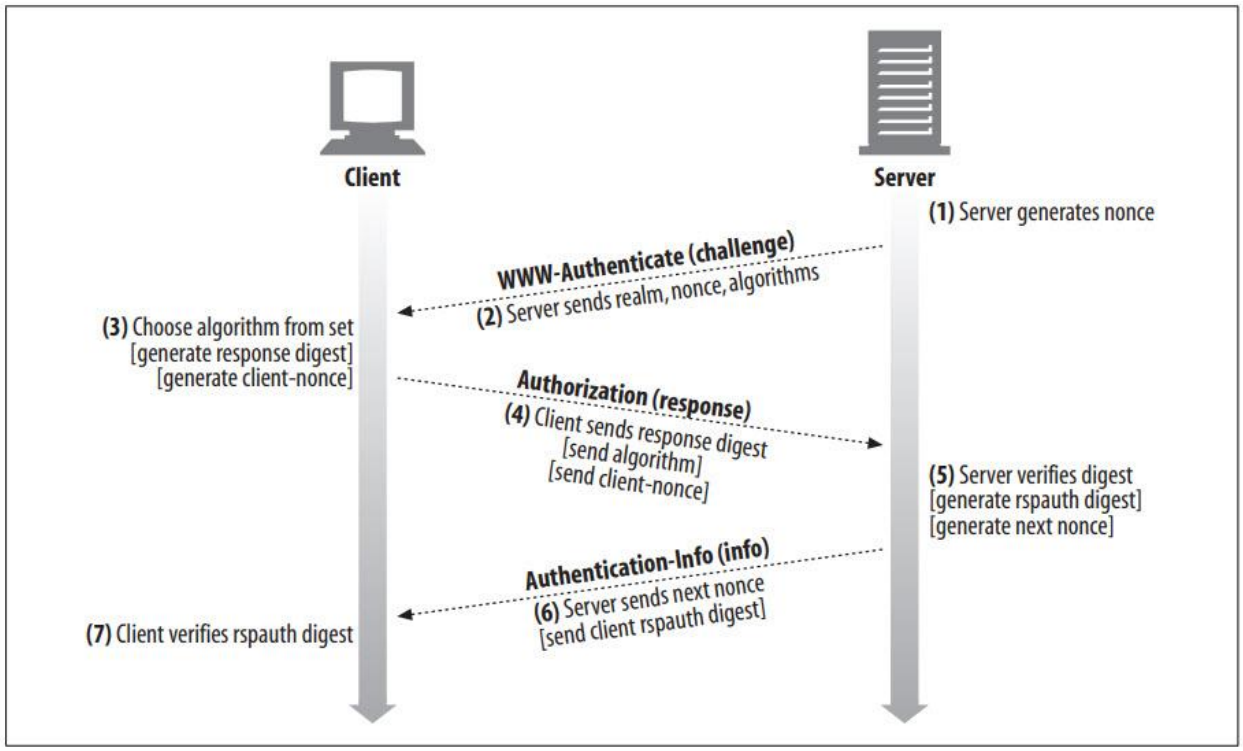
مخلوط کردن nonce با رمز عبور باعث می‌شود که هر بار که nonce تغییر می‌کند، Digest تغییر کند. این از حملات تکراری جلوگیری می‌کند، زیرا Digest رمز ثبت شده فقط برای یک مقدار nonce خاص معتبر است و بدون رمز عبور مخفی، مهاجم نمی‌تواند Digest درست را محاسبه کند.

Digest Authentication نیاز به استفاده از nonces دارد، زیرا یک ضعف در پخش بی‌اهمیت باعث می‌شود Un-nonced Digest Authentication به‌اندازه Basic Authentication ضعیف باشد. Nonces در چالش WWW-Authenticate از سروری به کلاینت دیگر منتقل می‌شود.

The Digest Authentication Handshake

پروتکل Digest Authentication یک نسخه پیشرفته از احراز هویت است که از هدرهایی مشابه آنچه در Basic Authentication استفاده می‌شود استفاده می‌کند. برخی از گزینه‌های جدید به هدرهای سنتی اضافه می‌شوند و یک هدر اختیاری جدید، Authorization-Info، اضافه می‌شود.

دست دادن سه مرحله ای ساده شده Digest Authentication در شکل زیر نشان داده شده است.



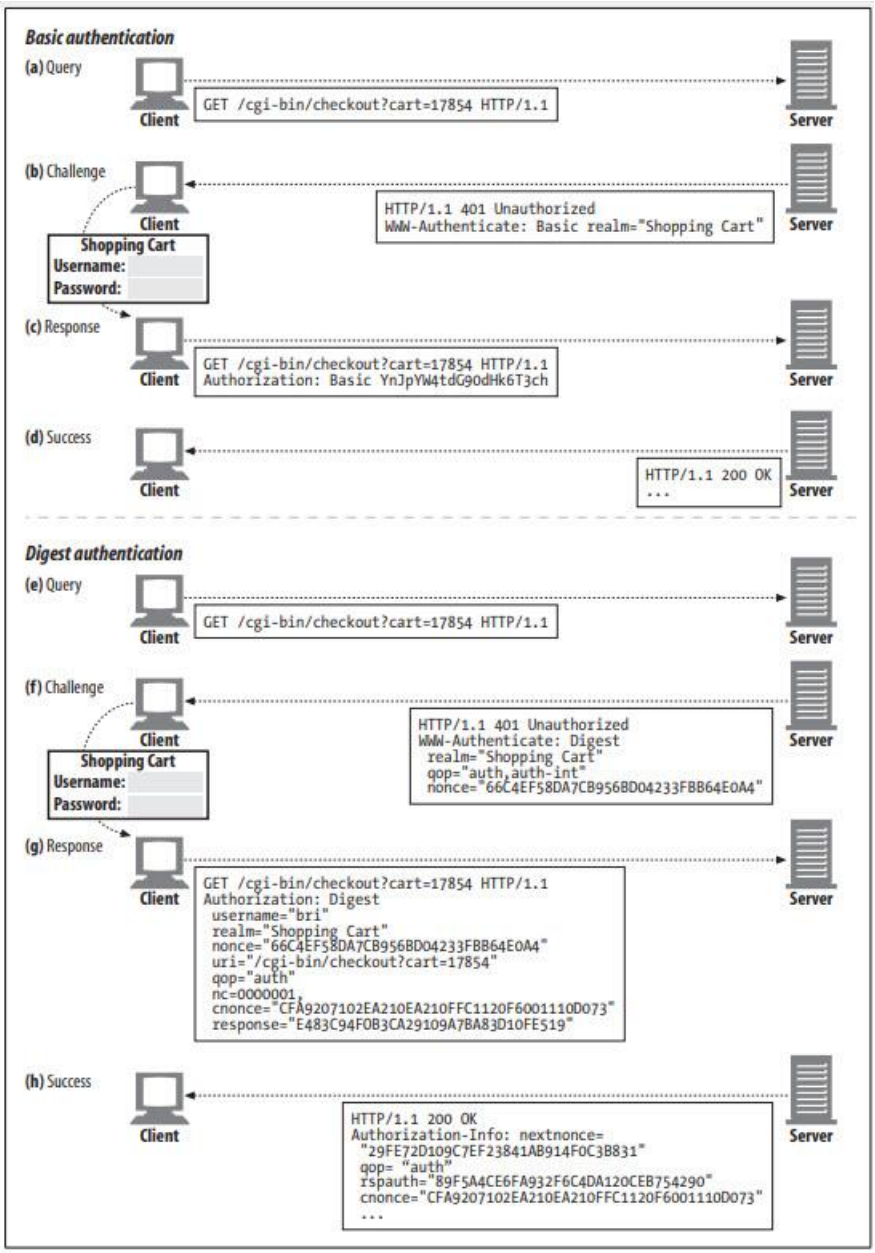
آنچه در شکل بالا اتفاق می‌افتد در اینجا آمده است:

- در مرحله ۱، سرور یک مقدار nonce را محاسبه می‌کند. در مرحله ۲، سرور nonce را در یک پیام چالشی WWW-Authenticate به همراه فهرستی از الگوریتم‌هایی که سرور پشتیبانی می‌کند، برای کلاینت ارسال می‌کند.

- در مرحله ۳، کلاینت الگوریتمی را انتخاب می‌کند و Digest رمز عبور مخفی و سایر داده‌ها را محاسبه می‌کند. در مرحله ۴، Digest را در یک پیام مجوز به سرور ارسال می‌کند. اگر کلاینت بخواهد سرور را احراز هویت کند، می‌تواند یک Client Nonce ارسال کند.
- در مرحله ۵، سرور Digest، الگوریتم انتخاب شده، و داده‌های پشتیبانی را دریافت می‌کند و همان Digest ای را که کلاینت انجام می‌دهد محاسبه می‌کند. سپس سرور Digest تولید شده محلی را با Digest ارسال شده از طریق شبکه مقایسه می‌کند و مطابقت آن‌ها را تأیید می‌کند. اگر کلاینت به طور متقارن سرور را با یک کلاینت به چالش بکشد، Digest مشتری ایجاد می‌شود. علاوه بر این، nonce بعدی را می‌توان از قبل محاسبه کرد و از قبل به کلاینت تحویل داد، بنابراین کلاینت می‌تواند دفعه بعد Digest مناسب را صادر کند.

بسیاری از این اطلاعات اختیاری هستند و دارای پیش فرض هستند. برای روشن شدن موارد، شکل ۱۳-۳ پیام‌های ارسال شده برای احراز هویت اولیه (a-d) را با یک مثال ساده از احراز هویت خلاصه (شکل e-h) مقایسه می‌کند.

اکنون بیایید کمی دقیق‌تر به عملکرد داخلی احراز هویت Digest نگاه کنیم.



Digest Calculations

قلب Digest Authentication، Digest است که یک طرفه ترکیبی از اطلاعات عمومی، اطلاعات مخفی و یک مقدار time-limited nonce می‌باشد. بیایید اکنون به نحوه محاسبه Digest ها نگاه کنیم. محاسبات Digest به طور کلی ساده است.

Digest Algorithm Input Data

Digest ها به وسیله سه جزء مورد محاسبه قرار می‌گیرند:

- یک جفت توابع متشکل از یک تابع هش یک طرفه H(d) و Digest KD(s,d)، که در آن s مخفف Secret و d مخفف داده است.
- تکه‌ای از داده‌ها حاوی اطلاعات امنیتی، از جمله رمز عبور مخفی، به نام A1
- تکه‌ای از داده‌ها حاوی ویژگی‌های غیرمخفی پیام درخواست به نام A2

دو قطعه از داده‌ها، A1 و A2، توسط H و KD پردازش می‌شوند تا یک Digest را ارائه دهند.

The Algorithms H(d) and KD(s,d)

احراز هویت Digest از انتخاب انواع الگوریتم‌های Digest پشتیبانی می‌کند. دو الگوریتم پیشنهاد شده در RFC 2617 MD5 و MD5-sess هستند (که در آن "sess" مخفف session است) و اگر الگوریتم دیگری مشخص نشده باشد، الگوریتم به طور پیش فرض روی MD5 قرار می‌گیرد.

اگر از MD5 یا MD5-sess استفاده شود، تابع H MD5 داده‌ها را محاسبه می‌کند، و تابع KD digest MD5 داده‌های مخفی و غیرمخفی متصل به کولون را محاسبه می‌کند. به عبارت دیگر:

H(<data>) = MD5(<data>)

KD(<secret>,<data>) = H(concatenate(<secret>:<data>))

The Security-Related Data (A1)

تکه‌ای از داده‌ها به نام A1 محصول اطلاعات محرمانه و حفاظتی مانند نام کاربری، رمز عبور، قلمرو حفاظتی(Protection Realm) و nonces است. A1 فقط به اطلاعات امنیتی مربوط می‌شود، نه به خود پیام اصلی. A1 به همراه H، KD و A2 برای محاسبه Digest ها استفاده می‌شود.

RFC 2617 بسته به الگوریتم انتخابی دو روش برای محاسبه A1 تعریف می‌کند:

MD5

هش‌های یک طرفه برای هر درخواست اجرا می‌شوند. A1 سه گانه نام کاربری، Realm و رمز عبور مخفی است که با کولون متصل می‌شود.

MD5-sess

تابع هش فقط یک بار در اولین دست دادن WWW-Authenticate اجرا می‌شود. هش CPU-intensive نام کاربری، Realm و رمز عبور مخفی یک بار انجام می‌شود و به مقادیر nonce و Client nonce (cnonce) فعلی اضافه می‌شود.

تعاریف A1 در جدول زیر نشان داده شده است.

Algorithm	A1
MD5	A1 = <user>:<realm>:<password>
MD5-sess	A1 = MD5(<user>:<realm>:<password>):<nonce>:<cnonce>

The Message-Related Data (A2)

تکه‌ای از داده‌ها به نام A2 اطلاعات مربوط به خود پیام را نشان می‌دهد، مانند URL، متد درخواست و بدنه موجودیت پیام. A2 برای کمک به محافظت در برابر دستکاری روش، منبع یا پیام استفاده می‌شود. A2 به همراه H، KD و A1 برای محاسبه Digest ها استفاده می‌شود.

RFC 2617 بسته به quality of protection یا qop انتخاب شده، دو طرح را برای A2 تعریف می‌کند:

- طرح اول فقط شامل روش درخواست HTTP و URL است. این بخش زمانی استفاده می‌شود که qop="auth" باشد که حالت پیش فرض است.
- طرح دوم بدنه موجودیت پیام را اضافه می‌کند تا درجه‌ای از بررسی یکپارچگی پیام را فراهم کند. زمانی که qop="auth-int" باشد استفاده می‌شود.

تعاریف A2 در جدول زیر نشان داده شده است.

qop	A2
undefined	<request-method>:<uri-directive-value>
auth	<request-method>:<uri-directive-value>
auth-int	<request-method>:<uri-directive-value>:H(<request-entity-body>)

request-method روش درخواست HTTP است. uri-directive-value URI درخواستی از خط درخواست است. این ممکن است «*»، «URL مطلق» یا «abs_path» باشد، اما باید با URI درخواست موافق باشد. به ویژه، اگر URI درخواست یک URL مطلق باشد، باید یک URL مطلق ارائه شود.

Overall Digest Algorithm

RFC 2617 با توجه به H، KD، A1 و A2، دو روش را برای محاسبه Digest ها تعریف می‌کند:

- راه اول برای سازگاری با مشخصات قدیمی RFC 2069 در نظر گرفته شده است که در صورت عدم وجود گزینه qop استفاده می‌شود. Digest را با استفاده از هش اطلاعات مخفی و داده‌های پیام nonced محاسبه می‌کند.

- راه دوم رویکرد مدرن و ترجیحی است - شامل پشتیبانی از شمارش nonce و احراز هویت متقارن است. این رویکرد زمانی استفاده می‌شود که "qop "auth یا "auth-int" باشد. داده‌های nonce count، qop و cnonce را به Digest اضافه می‌کند.

تعاریف تابع Digest حاصل در جدول زیر نشان داده شده است. به Digest های حاصل از H، KD، A1 و A2 توجه کنید.

qop	Digest algorithm	Notes
undefined	KD(H(A1), <nonce>:H(A2))	Deprecated
auth or auth-int	KD(H(A1), <nonce>:<nc>:<cnonce>:<qop>:H(A2))	Preferred

گم شدن در تمام لایه‌های کپسوله‌سازی مشتق کمی آسان است. این یکی از دلایلی است که برخی از خوانندگان با RFC 2617 مشکل دارند. جدول زیر برای آسان‌تر کردن آن، تعاریف H و KD را گسترش می‌دهد و Digest هایی را بر حسب A1 و A2 به جا می‌گذارد.

qop	Algorithm	Unfolded algorithm
undefined	<undefined> MD5 MD5-sess	MD5(MD5(A1):<nonce>:MD5(A2))

qop	Algorithm	Unfolded algorithm
auth	<undefined> MD5 MD5-sess	MD5(MD5(A1):<nonce>:<nc>:<cnonce>:<qop>:MD5(A2))
auth-int	<undefined> MD5 MD5-sess	MD5(MD5(A1):<nonce>:<nc>:<cnonce>:<qop>:MD5(A2))

Digest Authentication Session

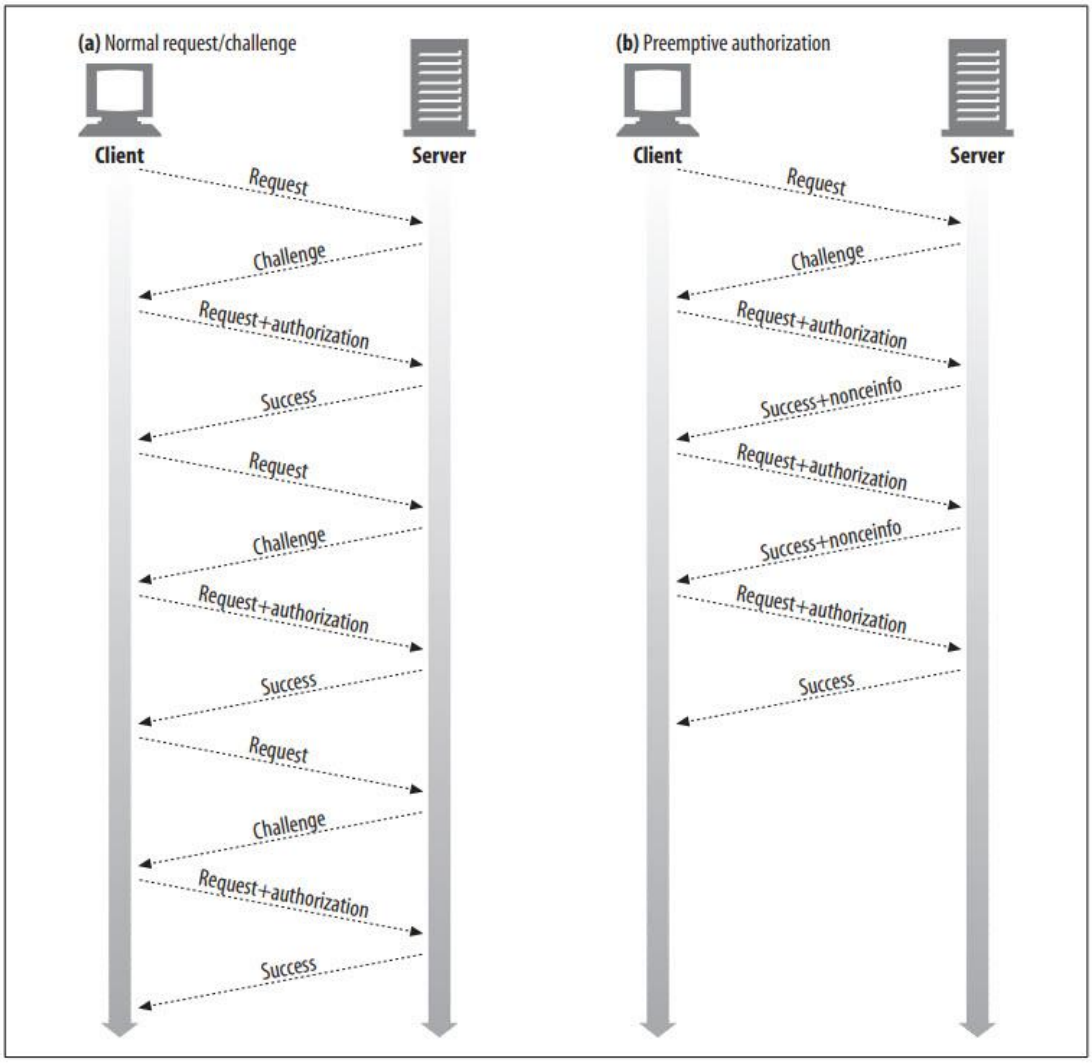
پاسخ کلاینت به چالش WWW-Authenticate برای یک فضای حفاظتی، جلسه احراز هویت را با آن فضای حفاظتی شروع می‌کند (Realm ترکیب شده با ریشه متعارف سروری که به آن دسترسی پیدا می‌شود، «فضای حفاظتی» را تعریف می‌کند).

جلسه احراز هویت تا زمانی که کلاینت چالش WWW-Authenticate دیگری را از هر سروری در فضای حفاظتی دریافت کند، ادامه دارد. یک کلاینت باید نام کاربری، رمز عبور، nonce، تعداد nonce

و مقادیر opaque مرتبط با جلسه احراز هویت را به خاطر بسپارد تا از آن برای ساخت هدر Authorization در درخواست‌های آینده در فضای حفاظتی استفاده کند.

وقتی nonce منقضی می‌شود، سرور می‌تواند اطلاعات هدر مجوز قدیمی را بپذیرد، حتی اگر مقدار nonce گنجانده شده تازه نباشد. از طرف دیگر، سرور ممکن است یک پاسخ ۴۰۱ را با یک مقدار nonce جدید برگرداند و باعث شود کلاینت درخواست را دوباره امتحان کند. با مشخص کردن "stale=true" با این پاسخ، سرور به مشتری می‌گوید که بدون درخواست نام کاربری و رمز عبور جدید، با nonce جدید دوباره امتحان کند.

Preemptive Authorization



در احراز هویت عادی، هر درخواست قبل از تکمیل تراکنش نیاز به یک چرخه request/challenge نیاز دارد. این در بخش a شکل بالا نشان داده شده است.

این چرخه request/challenge را می‌توان حذف کرد اگر کلاینت از قبل بداند nonce بعدی چیست، بنابراین می‌تواند هدر Authorization صحیح را قبل از درخواست سرور ایجاد کند. اگر کلاینت بتواند هدر Authorization را قبل از درخواست محاسبه کند، کلاینت می‌تواند به طور پیشگیرانه هدر Authorization را به سرور صادر کند، بدون اینکه ابتدا request/challenge را پشت سر بگذارد. تاثیر عملکرد در بخش b شکل بالا نشان داده شده است.

مجوز پیشگیرانه برای Basic Authentication بی‌اهمیت (و رایج) است. مرورگرها معمولاً پایگاه داده‌های client-side ای از نام‌های کاربری و رمز عبور را نگهداری می‌کنند. هنگامی که کاربر به یک سایت احراز هویت می‌کند، مرورگر معمولاً هدر Authorization صحیح را برای درخواست‌های بعدی به آن URL ارسال می‌کند (به فصل ۱۲ مراجعه کنید).

مجوز پیشگیرانه برای Digest Authentication کمی پیچیده‌تر است، زیرا فناوری nonce در نظر گرفته شده برای خنثی کردن Replay Attack است. از آنجایی که سرور، nonces دلخواه را تولید می‌کند، همیشه راهی برای کلاینت وجود ندارد که تعیین کند تا زمانی که یک چالش را دریافت کند، چه هدر Authorization را ارسال کند.

Digest Authentication چند ابزار برای مجوز پیشگیرانه ارائه می‌دهد در حالی که بسیاری از ویژگی‌های ایمنی را حفظ می‌کند. در اینجا سه راه بالقوه وجود دارد که کلاینت می‌تواند بدون انتظار برای چالش جدید WWW-Authenticate، Nonce صحیح را به دست آورد:

- سرور، nonce بعدی را در هدر موفقیت Authentication-Info از قبل ارسال می‌کند.
- سرور اجازه می‌دهد تا از همان nonce برای یک پنجره زمانی کوچک استفاده مجدد شود.
- هم کلاینت و هم سرور از یک الگوریتم nonce-generation همگام و قابل پیش بینی استفاده می‌کنند.

Next nonce pregeneration

مقدار nonce بعدی را می‌توان از قبل توسط هدر موفقیت AuthenticationInfo در اختیار کلاینت قرار داد. این هدر همراه با پاسخ OK 200 از یک احراز هویت موفق قبلی ارسال می‌شود.

```
Authentication-Info: nextnonce="<nonce-value>"
```

با توجه به nonce بعدی، کلاینت می‌تواند به صورت پیشگیرانه یک هدر Authorization صادر کند.

در حالی که این مجوز پیشگیرانه از چرخه request/challenge جلوگیری می‌کند (سرعت تراکنش را افزایش می‌دهد)، همچنین به طور موثر توانایی Pipeline کردن چندین درخواست به یک سرور را باطل می‌کند، زیرا مقدار nonce بعدی باید قبل از صدور درخواست بعدی دریافت شود. از آنجا که انتظار می‌رود Pipeline یک فناوری اساسی برای جلوگیری از تأخیر باشد، جریمه عملکرد ممکن است زیاد باشد.

Limited nonce reuse

به جای ایجاد پیش‌تولید دنباله‌ای از nonces، رویکرد دیگر اجازه استفاده مجدد محدود از nonces است. به عنوان مثال، یک سرور ممکن است اجازه دهد یک nonce ۵ بار یا برای ۱۰ ثانیه استفاده مجدد شود.

در این حالت، کلاینت می‌تواند آزادانه درخواست‌هایی را با هدر Authorization صادر کند و می‌تواند آن‌ها را Pipeline کند، زیرا nonce از قبل شناخته شده است. زمانی که nonce در نهایت منقضی شود، انتظار می‌رود سرور یک چالش ۴۰۱ غیر مجاز را با مجموعه دستورات WWW-Authenticate: stale=true برای کلاینت ارسال کند:

```
WWW-Authenticate: Digest
 realm="<realm-value>"
 nonce="<nonce-value>"
 stale=true
```

استفاده مجدد از nonces امنیت را کاهش می‌دهد، زیرا موفقیت مهاجم را در Replay Attack آسان‌تر می‌کند. از آنجایی که طول عمر استفاده مجدد nonce قابل کنترل است، از عدم استفاده مجدد تا استفاده مجدد بالقوه طولانی، می‌توان بین پنجره‌های آسیب پذیری و عملکرد، معاوضه‌هایی ایجاد کرد.

علاوه بر این، از ویژگی‌های دیگری نیز می‌توان برای دشوارتر کردن Replay Attack استفاده کرد، از جمله افزایش شمارنده و آزمایش آدرس IP. با این حال، در حالی که حملات را ناخوشایندتر می‌کنند، این تکنیک‌ها آسیب‌پذیری را از بین نمی‌برند.

Synchronized nonce generation

امکان استفاده از الگوریتم‌های nonce-generation همگام‌سازی شده با زمان وجود دارد، که در آن کلاینت و سرور می‌توانند دنباله‌ای از nonces یکسان را بر اساس یک کلید مخفی مشترک ایجاد کنند، که شخص ثالث به راحتی نمی‌تواند آن‌ها را پیش‌بینی کند (مانند کارت‌های شناسایی امن).

این الگوریتم‌ها فراتر از محدوده مشخصات Digest Authentication هستند.

Nonce Selection

محتویات nonce مبهم (opaque) و وابسته به اجرا هستند. با این حال، کیفیت عملکرد، امنیت و راحتی به انتخاب هوشمندانه بستگی دارد.

RFC 2617 این فرمول فرضی nonce را پیشنهاد می‌کند:

BASE64(time-stamp H(time-stamp ":" ETag ":" private-key))

در جایی که time-stamp یک زمان تولید شده توسط سرور یا مقدار غیر تکراری دیگر است، ETag مقدار هدر HTTP ETag مرتبط با موجودیت درخواستی است و کلید خصوصی داده‌هایی است که فقط برای سرور شناخته شده است.

با وجود nonce این فرم، سرور پس از دریافت هدر احراز هویت کلاینت، بخش هش را مجدداً محاسبه می‌کند و در صورتی که با nonce از آن هدر مطابقت نداشته باشد یا مقدار time-stamp به اندازه کافی جدید نباشد، درخواست را رد می‌کند. به این ترتیب سرور می‌تواند مدت اعتبار nonce را محدود کند.

گنجاندن ETag از درخواست مجدد برای نسخه به روز شده منبع جلوگیری می‌کند. (توجه داشته باشید که قرار دادن آدرس IP کلاینت در nonce به نظر می‌رسد به سرور این امکان را می‌دهد که استفاده مجدد از nonce را به همان کلاینتی که در ابتدا آن را دریافت کرده است محدود کند. با این حال، این کار مزرعه‌های پروکسی (Proxy Farms) را که در آن درخواست‌های یک تک کاربر اغلب از طریق پروکسی‌های مختلف استفاده می‌کند. همچنین جعل آدرس IP چندان سخت نیست.)

یک پیاده سازی ممکن است برای محافظت در برابر Replay Attack، یک nonce یا Digest استفاده شده قبلی را نپذیرد. یا، یک پیاده‌سازی ممکن است استفاده از nonces یا Digest های یک‌باره را برای درخواست‌های POST یا PUT و time-stamp برای درخواست‌های GET انتخاب کند.

Symmetric Authentication

RFC 2617، Digest Authentication را گسترش می‌دهد تا به کلاینت امکان تأیید اعتبار سرور را بدهد. این کار را با ارائه یک مقدار nonce کلاینت انجام می‌دهد، که سرور یک Digest پاسخ صحیح را بر اساس دانش صحیح اطلاعات مخفی مشترک ایجاد می‌کند. سپس سرور این Digest را در هدر Authorization-Info به کلاینت برمی‌گرداند.

این احراز هویت متقارن استاندارد RFC 2617 است. برای سازگاری با استاندارد قدیمی‌تر RFC 2069 اختیاری است، اما از آنجایی که پیشرفت‌های امنیتی مهمی را ارائه می‌کند، به همه کلاینت‌ها و سرورهای مدرن قویاً توصیه می‌شود که همه ویژگی‌های RFC 2617 را پیاده‌سازی کنند. به طور خاص، احراز هویت متقارن باید هر زمان که یک دستورالعمل qop وجود دارد انجام شود و لازم است زمانی که دستورالعمل qop وجود ندارد، انجام نشود.

Digest پاسخ مانند Digest درخواست محاسبه می‌شود، با این تفاوت که اطلاعات بدنه پیام (A2) متفاوت است، زیرا هیچ متدی در پاسخ وجود ندارد و داده‌های موجودیت پیام متفاوت است. روش‌های محاسبه A2 برای Digest درخواست و پاسخ در جداول زیر مقایسه شده است.

qop	A2
undefined	<request-method>:<uri-directive-value>
auth	<request-method>:<uri-directive-value>
auth-int	<request-method>:<uri-directive-value>:H(<request-entity-body>)

qop	A2
undefined	:<uri-directive-value>
auth	:<uri-directive-value>
auth-int	:<uri-directive-value>:H(<response-entity-body>)

مقدار cnonce و مقدار nc باید برای درخواست کلاینت باشد که این پیام به آن پاسخ می‌دهد. اگر qop="auth" یا qop="auth-int" مشخص شده باشد، دستورات پاسخ auth، cnonce و nonce count باید وجود داشته باشند.

Quality of Protection Enhancements

فیلد qop ممکن است در هر سه هدر Digest وجود داشته باشد: WWW-Authenticate، Authorization و Authentication-Info.

فیلد qop به کلاینت‌ها و سرورها اجازه می‌دهد تا برای انواع و کیفیت‌های مختلف حفاظت، مذاکره کنند. برای مثال، برخی از تراکنش‌ها ممکن است بخواهند یکپارچگی متن پیام را بررسی کنند، حتی اگر این امر انتقال را به میزان قابل توجهی کُند کند.

سرور ابتدا لیستی از گزینه‌های qop جدا شده با کاما را در هدر WWW-Authenticate صادر می‌کند. سپس کلاینت یکی از گزینه‌هایی را که پشتیبانی می‌کند و نیازهایش را برآورده می‌کند انتخاب می‌کند و آن را در قسمت Authorization qop خود به سرور ارسال می‌کند.

استفاده از qop اختیاری است، اما فقط برای سازگاری با مشخصات قدیمی RFC 2069. گزینه qop باید توسط همه پیاده سازی‌های Digest مدرن پشتیبانی شود.

RFC 2617 دو کیفیت اولیه ارزش حفاظتی را تعریف می‌کند: "auth" که نشان دهنده احراز هویت است و "aut-int" که نشان دهنده احراز هویت با محافظت از یکپارچگی پیام است.

Message Integrity Protection

اگر حفاظت یکپارچگی اعمال شود (qop="auth-int")، H (بدنه موجودیت) هش بدنه موجودیت است، نه بدنه پیام. قبل از اعمال رمزگذاری انتقال توسط فرستنده و پس از حذف آن توسط گیرنده محاسبه می‌شود. توجه داشته باشید که این شامل مرزهای چند قسمتی و هدرهای تعبیه شده در هر قسمت از هر نوع محتوای چند بخشی است.

Digest Authentication Headers

هر دو پروتکل احراز هویت Basic و Digest شامل یک چالش مجوز است که توسط هدر WWW-Authenticate و یک پاسخ مجوز که توسط هدر مجوز انجام می‌شود. احراز هویت Digest یک هدر اختیاری Authorization-Info را اضافه می‌کند که پس از احراز هویت موفقیت‌آمیز ارسال می‌شود تا یک دست دادن سه مرحله‌ای تکمیل شود و در امتداد nonce بعدی استفاده شود. هدرهای احراز هویت Basic و Digest در جدول زیر نشان داده شده است.

هدرهای احراز هویت Digest کمی پیچیده‌تر هستند.

Phase	Basic	Digest
Challenge	`WWW-Authenticate: Basic realm="<realm-value>"`	`WWW-Authenticate: Digest realm="<realm-value>" nonce="<nonce-value>" [domain="<list-of-URIs>"] [opaque="<opaque-token-value>"] [stale=<true-or-false>] [algorithm=<digest-algorithm>] [qop="<list-of-qop-values>"] [<extension-directive>]`
Response	`Authorization: Basic <base64(user:pass)>`	`Authorization: Digest username="<username>" realm="<realm-value>" nonce="<nonce-value>" uri=<request-uri> response="<32-hex-digit-digest>" [algorithm=<digest-algorithm>] [opaque="<opaque-token-value>"] [cnonce="<nonce-value>"] [qop=<qop-value>] [nc=<8-hex-digit-nonce-count>] [<extension-directive>]`
Info	n/a	`Authentication-Info: nextnonce="<nonce-value>" [qop="<list-of-qop-values>"] [rspauth="<hex-digest>"] [cnonce="<nonce-value>"] [nc=<8-hex-digit-nonce-count>]`

Practical Considerations

چندین چیز وجود دارد که باید هنگام کار با احراز هویت Digest در نظر گرفته شود. این بخش برخی از این مسائل را مورد بحث قرار می‌دهد.

Multiple Challenges

یک سرور می‌تواند چندین چالش برای یک منبع ایجاد کند. به عنوان مثال، اگر سروری توانایی‌های یک کلاینت را نداند، ممکن است چالش‌های اساسی هم در احراز هویت Basic و هم Digest را فراهم کند. هنگام مواجهه با چالش‌های متعدد، کلاینت باید با قوی‌ترین مکانیزم احراز هویت که پشتیبانی می‌کند، پاسخ دهد.

اگر مقدار فیلد هدر WWW-Authenticate یا ProxyAuthenticate حاوی بیش از یک چالش باشد یا اگر بیش از یک فیلد هدر WWW-Authenticate ارائه شده باشد، باید در تجزیه و تحلیل مقدار فیلد WWW-Authenticate یا ProxyAuthenticate دقت ویژه‌ای داشته باشند، زیرا

چالش ممکن است خود حاوی لیستی از احراز هویت جدا شده با کاما باشد. توجه داشته باشید که بسیاری از مرورگرها فقط احراز هویت Basic را می‌شناسند و نیاز دارند که اولین مکانیزم احراز هویت ارائه شده باشد.

هنگام ارائه طیفی از گزینه‌های احراز هویت، نگرانی‌های امنیتی آشکاری مانند «ضعیف‌ترین لینک» وجود دارد. سرورها باید احراز هویت Basic را تنها در صورتی شامل شوند که حداقل قابل قبول باشد و مدیران باید به کاربران در مورد خطرات اشتراک گذاری رمز عبور یکسان در سیستم‌ها در هنگام استفاده از سطوح مختلف امنیتی هشدار دهند.

Error Handling

در احراز هویت Digest، اگر یک دستورالعمل یا مقدار آن نامناسب باشد، یا اگر دستورالعمل مورد نیاز وجود نداشته باشد، پاسخ مناسب 400 Bad Request است.

اگر Digest درخواست مطابقت نداشته باشد، یک login failure باید ثبت شود. failure های مکرر از یک کلاینت ممکن است نشان دهنده تلاش مهاجم برای حدس زدن رمزهای عبور باشد.

سرور احراز هویت باید اطمینان حاصل کند که منبع تعیین شده توسط دستورالعمل "uri" با منبع مشخص شده در خط درخواست یکسان است. اگر آن‌ها متفاوت هستند، سرور باید یک خطای 400 Bad Request را برگرداند. (از آنجایی که این ممکن است نشانه‌ای از یک حمله باشد، طراحان سرور ممکن است بخواهند ثبت چنین خطاهایی را در نظر بگیرند.) کپی کردن اطلاعات از URL درخواست در این فیلد با این احتمال که یک پروکسی میانی ممکن است خط درخواست کلاینت را تغییر دهد، سروکار دارد. این درخواست تغییر یافته منجر به خلاصه ای مشابه آنچه توسط کلاینت محاسبه می‌شود، نمی‌شود.

Protection Spaces

مقدار Realm، در ترکیب با URL ریشه متعارف سرور مورد دسترسی، فضای حفاظتی را مشخص می‌کند.

Realm ها به منابع محافظت شده روی یک سرور اجازه می‌دهند که به مجموعه‌ای از فضاهای حفاظتی تقسیم شوند که هر کدام دارای طرح احراز هویت و/یا پایگاه داده مجوز خود هستند. مقدار Realm رشته‌ای است که معمولاً توسط سرور مبدا اختصاص داده می‌شود، که ممکن است معنای خاصی برای طرح احراز هویت داشته باشد. توجه داشته باشید که ممکن است چالش‌های متعددی با طرح مجوز یکسان اما حوزه‌های مختلف وجود داشته باشد.

فضای حفاظتی، دامنه‌ای را تعیین می‌کند که Credentialها می‌توانند به طور خودکار روی آن اعمال شوند. اگر درخواست قبلی مجوز داده شده باشد، می‌توان از همان Credentialها برای تمام

درخواست‌های دیگر در آن فضای حفاظتی برای مدت زمان تعیین‌شده توسط طرح احراز هویت، پارامترها و/یا اولویت کاربر استفاده مجدد کرد. مگر اینکه توسط طرح احراز هویت به گونه دیگری تعریف شده باشد، یک فضای حفاظتی واحد نمی‌تواند خارج از محدوده سرور خود گسترش یابد.

محاسبه خاص فضای حفاظتی به مکانیسم احراز هویت بستگی دارد:

در احراز هویت Basic، کلاینت‌ها فرض می‌کنند که تمام مسیرها در URI درخواست یا پایین‌تر از آن، در همان فضای حفاظتی چالش فعلی قرار دارند. یک کلاینت می‌تواند به طور پیشگیرانه برای منابع در این فضا مجوز دهد بدون اینکه منتظر چالش دیگری از طرف سرور باشد.

در احراز هویت Digest، فیلد دامنه WWW-Authenticate: دقیقاً فضای حفاظتی را تعریف می‌کند. فیلد دامنه یک لیست نقل‌قول و جدا شده از URIها است. همه URIهای موجود در لیست دامنه و همه URIهای منطقی زیر این پیشوندها، در فضای حفاظتی یکسانی قرار دارند. اگر فیلد دامنه گم یا خالی باشد، تمام URI ها در سرور Challenging Server در فضای حفاظتی قرار دارند.

Rewriting URIs

پراکسی‌ها ممکن است URI ها را به گونه‌ای بازنویسی کنند که Syntax مربوط به URI را تغییر دهد اما منبع واقعی توصیف شده را تغییر ندهد. مثلا:

- نام هاست ممکن است عادی شده یا با آدرس های IP جایگزین شوند.
- کاراکترهای تعبیه شده (Embedded) ممکن است با فرم های Escape "%" جایگزین شوند.
- ویژگی‌های اضافی از نوعی که بر منبع Fetch شده از سرور مبدا خاص تأثیر نمی‌گذارد ممکن است به URI اضافه یا درج شود.

از آنجا که URI ها را می‌توان توسط پراکسی‌ها تغییر داد و از آنجایی که sanity احراز هویت Digest یکپارچگی مقدار URI را بررسی می‌کند، در صورت ایجاد هر یک از این تغییرات، احراز هویت Digest خراب می‌شود.

Caches

هنگامی که یک Shared Cache درخواستی حاوی یک هدر Authorization و پاسخی از ارسال آن درخواست دریافت می‌کند، نباید آن پاسخ را به عنوان پاسخ به درخواست دیگری برگرداند، مگر اینکه یکی از دو دستورالعمل Cache-Control در پاسخ وجود داشته باشد:

- اگر پاسخ اولیه شامل دستورالعمل must-revalidate Cache-Control باشد، Cache ممکن است از موجودیت آن پاسخ در پاسخ به درخواست بعدی استفاده کند. با این حال، ابتدا باید با استفاده از هدرهای درخواست مربوط به درخواست جدید، آن را با سرور مبدا تأیید مجدد کند تا سرور مبدا بتواند درخواست جدید را تأیید کند.
- اگر پاسخ اصلی شامل دستور العمل Public Cache-Control باشد، موجودیت پاسخ ممکن است در پاسخ به هر درخواست بعدی بازگردانده شود.

Security Considerations

RFC 2617 در خلاصه کردن برخی از خطرات امنیتی ذاتی در طرح های احراز هویت HTTP کار قابل تحسینی انجام می‌دهد. این بخش برخی از این خطرات را شرح می‌دهد.

Header Tampering

برای ارائه یک سیستم بدون خطا در برابر دستکاری هدر، به رمزگذاری سرتاسر یا امضای دیجیتال هدرها نیاز دارید (ترجیحاً ترکیبی از هر دو!)

احراز هویت Digest بر ارائه یک طرح احراز هویت ضد دستکاری متمرکز است، اما لزوماً این حفاظت را به داده‌ها گسترش نمی‌دهد. تنها هدرهایی که دارای سطحی از حفاظت هستند WWW-Authenticate و Authorization هستند.

Replay Attacks

Replay Attack، در شرایط فعلی، زمانی است که شخصی از مجموعه‌ای از Credentialهای احراز هویت پنهان شده از یک تراکنش معین برای تراکنش دیگر استفاده می‌کند. در حالی که این مشکل در درخواست‌های GET وجود دارد، بسیار مهم است که یک روش بی‌خطر برای جلوگیری از Replay Attack برای درخواست‌های POST و PUT در دسترس باشد. توانایی Replay موفقیت آمیز Credentialهای استفاده شده قبلی در حین انتقال داده‌های فرم می‌تواند باعث ایجاد کابوس‌های امنیتی شود.

بنابراین، برای اینکه سرور Credentialهای " Replay شده" را بپذیرد، مقادیر nonce باید تکرار شوند. یکی از راه‌های کاهش این مشکل این است که سرور یک nonce تولید کند که حاوی Digest ای از آدرس IP کلاینت، یک مهر زمانی، ETag منبع و یک کلید سرور خصوصی (همانطور که قبلاً توصیه شد). در چنین سناریویی، ترکیب یک آدرس IP و یک مقدار کوتاه مدت ممکن است مانع بزرگی برای مهاجم باشد.

با این حال، این راه حل یک اشکال اساسی دارد. همانطور که قبلاً بحث کردیم، مشکل، استفاده از آدرس IP کلاینت در ایجاد یک انتقال بدون وقفه از طریق Proxy Farms، که در آن درخواست‌های یک کاربر ممکن است از طریق پراکسی‌های مختلف انجام شود، می‌باشد. همچنین جعل IP چندان سخت نیست.

یکی از راه‌های جلوگیری از Replay Attack، استفاده از یک مقدار nonce منحصر به فرد برای هر تراکنش است. در این پیاده سازی، برای هر تراکنش، سرور یک nonce منحصربفرد به همراه مقدار timeout صادر می‌کند. مقدار nonce صادر شده فقط برای تراکنش داده شده معتبر است و فقط برای مدت زمان مقدار وقفه. این حسابداری ممکن است بار روی سرورها را افزایش دهد. با این حال، افزایش باید اندک باشد.

Multiple Authentication Mechanisms

هنگامی که یک سرور از طرح‌های احراز هویت چندگانه (مانند Basic و Digest) پشتیبانی می‌کند، معمولاً انتخاب را در هدرهای WWW-Authenticate ارائه می‌کند. از آنجایی که کلاینت ملزم به انتخاب قوی‌ترین مکانیسم احراز هویت نیست، قدرت احراز هویت حاصله به اندازه ضعیف‌ترین طرح‌های احراز هویت است.

راه‌های واضح برای جلوگیری از این مشکل این است که کلاینت‌ها همیشه قوی‌ترین طرح احراز هویت موجود را انتخاب کنند. اگر این عملی نباشد (چون اکثر ما از کلاینت‌های تجاری موجود استفاده می‌کنیم)، تنها گزینه دیگر استفاده از سرور پروکسی برای حفظ قوی‌ترین طرح احراز هویت است. با این حال، چنین رویکردی تنها در حوزه‌ای امکان‌پذیر است که در آن همه کلاینت‌ها می‌توانند از طرح احراز هویت انتخاب شده پشتیبانی کنند - به عنوان مثال، یک شبکه شرکتی.

Dictionary Attacks

حملات دیکشنری حملات معمولی با حدس زدن رمز عبور هستند. یک کاربر مخرب می‌تواند یک تراکنش را استراق سمع کند و از یک برنامه حدس زدن رمز عبور استاندارد در برابر جفت‌های nonce/response استفاده کند. اگر کاربران از رمزهای عبور نسبتاً ساده استفاده می‌کنند و سرورها از nonces ساده استفاده می‌کنند، یافتن یک مطابقت کاملاً ممکن است. اگر Aging Policy رمز عبور وجود نداشته باشد، با توجه به زمان کافی و هزینه یکباره شکستن رمزهای عبور، جمع آوری رمزهای عبور کافی برای آسیب واقعی آسان است.

واقعاً هیچ راه خوبی برای حل این مشکل وجود ندارد، به جز استفاده از رمزهای عبور نسبتاً پیچیده که به سختی شکسته می‌شوند و یک Aging Policy خوب رمز عبور.

Hostile Proxies and Man-in-the-Middle Attacks

امروزه بسیاری از ترافیک اینترنت از طریق یک پروکسی در یک نقطه یا نقطه دیگر انجام می‌شود. با ظهور تکنیک‌های تغییر مسیر و رهگیری پراکسی‌ها، کاربر ممکن است حتی متوجه نشود که درخواست او از طریق یک پروکسی انجام می‌شود. اگر یکی از آن پراکسی‌ها متخاصم یا در معرض خطر باشد، می‌تواند کلاینت را در برابر حمله Man-in-the-Middle آسیب‌پذیر کند.

چنین حمله‌ای می‌تواند به شکل استراق سمع یا تغییر طرح‌های احراز هویت موجود با حذف همه گزینه‌های ارائه شده و جایگزینی آن‌ها با ضعیف‌ترین طرح احراز هویت (مانند احراز هویت Basic) باشد.

یکی از راه‌های به خطر انداختن یک پروکسی قابل اعتماد، استفاده از رابط‌های افزونه (Extension Interfaces) آن است. پروکسی‌ها گاهی اوقات رابط‌های برنامه نویسی پیچیده‌ای را ارائه می‌دهند و با چنین پراکسی‌هایی ممکن است امکان نوشتن یک افزونه (به عنوان مثال، افزونه) برای رهگیری و تغییر ترافیک وجود داشته باشد.

هیچ راه خوبی برای رفع این مشکل وجود ندارد. راه‌حل‌های ممکن شامل ارائه سرنخ‌های بصری در مورد قدرت احراز هویت، پیکربندی کلاینت‌ها برای استفاده از قوی‌ترین احراز هویت ممکن، و غیره است، اما حتی زمانی که از قوی‌ترین طرح احراز هویت ممکن استفاده می‌شود، کلاینت‌ها همچنان در برابر استراق سمع آسیب‌پذیر هستند. تنها راه مطمئن برای محافظت در برابر این حملات استفاده از SSL است.

Chosen Plaintext Attacks

کلاینت‌هایی که از احراز هویت Digest استفاده می‌کنند از nonce ارائه شده توسط سرور برای تولید پاسخ استفاده می‌کنند. با این حال، اگر یک پروکسی آسیب‌دیده یا مخرب در وسط وجود داشته باشد که ترافیک را رهگیری می‌کند (یا یک سرور مبدا مخرب)، می‌تواند به راحتی یک nonce را برای محاسبه پاسخ توسط کلاینت ارائه کند. استفاده از کلید شناخته شده برای محاسبه پاسخ ممکن است تحلیل رمزی پاسخ را آسان‌تر کند. این حمله متن ساده انتخاب شده (Chosen Plaintext Attacks) نامیده می‌شود. چند نوع از حملات متن ساده انتخاب شده وجود دارد:

Precomputed dictionary attacks

این ترکیبی از حمله دیکشنری و حمله متن ساده انتخاب شده است. ابتدا، سرور حمله کننده مجموعه‌ای از پاسخ‌ها را با استفاده از تغییرات غیرمعمول از پیش تعیین شده و رمز عبور رایج ایجاد می‌کند و یک دیکشنری ایجاد می‌کند. هنگامی که یک دیکشنری قابل توجه در دسترس است، سرور/پراکسی مهاجم می‌تواند ممنوعیت ترافیک را کامل کند و شروع به ارسال nonces از پیش تعیین شده برای کلاینت‌ها کند.

هنگامی که از یک کلاینت پاسخ دریافت می‌کند، مهاجم دیکشنری تولید شده را برای موارد مشابه جستجو می‌کند. اگر مطابقت وجود داشته باشد، مهاجم رمز عبور آن کاربر خاص را دارد.

Batched brute-force attacks

تفاوت در حمله گروهی brute-force در محاسبه رمز عبور است. به جای تلاش برای مطابقت با یک Digest از پیش محاسبه شده، مجموعه‌ای از ماشین‌ها روی شمارش همه رمزهای عبور ممکن برای یک فضای معین کار می‌کنند. همانطور که ماشین‌ها سریعتر می‌شوند، حمله brute-force بیشتر و بیشتر قابل اجرا می‌شود.

به طور کلی، تهدید ناشی از این حملات به راحتی قابل مقابله است. یکی از راه‌های جلوگیری از آن‌ها، پیکربندی کلاینت‌ها برای استفاده از دستورالعمل cnonce اختیاری است، به طوری که پاسخ به صلاحدید کلاینت تولید می‌شود، نه از nonce ارائه‌شده توسط سرور (که می‌تواند توسط مهاجم به خطر بیفتد). این، همراه با سیاست‌های اعمال رمزهای عبور نسبتاً قوی و یک مکانیسم خوب aging رمز عبور، می‌تواند خطر حملات متن ساده انتخابی را به طور کامل کاهش دهد.

Storing Passwords

مکانیزم احراز هویت Digest، پاسخ کاربر را با آنچه در داخل سرور ذخیره می‌شود مقایسه می‌کند - معمولاً نام‌های کاربری و تاپل‌های H(A1)، که در آن H(A1) از Digest نام کاربری، Realm و رمز عبور مشتق می‌شود.

بر خلاف فایل رمز عبور سنتی در جعبه یونیکس، اگر فایل رمز عبور احراز هویت Digest به خطر بیفتد، تمام اسناد موجود در Realm `فوراً در دسترس مهاجم هستند. نیازی به مرحله رمزگشایی نیست.

برخی از راه‌های کاهش این مشکل عبارتند از:

- از فایل گذرواژه طوری محافظت کنید که گویی حاوی رمزهای عبور متنی واضح است.
- اطمینان حاصل کنید که نام Realm در بین همه Realm ها منحصر به فرد است، به طوری که اگر یک فایل رمز عبور به خطر بیفتد، آسیب به یک Realm خاص محلی شود. یک نام Realm کاملاً واجد شرایط با میزبان و دامنه باید این نیاز را برآورده کند.

در حالی که احراز هویت Digest راه حلی بسیار قوی‌تر و ایمن‌تر از احراز هویت Basic ارائه می‌دهد، اما هنوز هیچ گونه حفاظتی برای امنیت محتوا ارائه نمی‌دهد - یک تراکنش واقعا امن فقط از طریق SSL امکان پذیر است که در فصل بعدی توضیح می‌دهیم.

For More Information

http://www.ietf.org/rfc/rfc2617.txt