پروتكل

كوكي ايمن

چكيده:

كوكي‌ها در تصديق درخواستهاي HTTP و نگه داشتن وضعيت‌هاي كلاينت‌ها براي برنامه هاي كاربردي وب وسيله اصلي هستند. بيشتر برنامه‌هاي كاربردي وب (از قبيل تجارت الكترونيك) يك پروتكل كوكي ايمن مي‌خواهند.

اين قبيل پروتكل‌ها لزوما بايد چهار خدمت زير را فراهم كنند: تعيين اعتبار، قابليت اعتماد، تماميت و ضد حمله تکرار. چندين پروتكل كوكي ايمن در مقالات قبلي پيشنهاد شده بود. هر چند که هيچ يك از آنها كاملا رضايت بخش نبود. در اين مقاله، ما يك پروتكل كوكي ايمن كه موثر، كارامد و نیز گسترش يافتن آن هم آسان است را پيشنهاد مي‌كنيم. برحسب كارايي، پروتكل ما همه چهار خدمت امنيتي بالا را فراهم مي‌كند. برحسب اثر بخشي، پروتكل ما شامل هيچ جستجوي ديتابيس يا كليد عمومي رمزنگاري نيست. برحسب توانايي گسترش يافتن، پروتكل ما مي تواند به آساني روي وب سرور موجود گسترش يابد و نياز به هيچ تغييری در ويژگيهاي كوكي اينترنتی ندارد. ما در PHP پروتكل كوكي ايمن خود را بكار برديم و نتيجه هاي علمي نشان مي‌دهد كه پروتكل ما خيلي كارآمد است.

1. مقدمه:

HTTP (پروتكل انتقال فرامتن) بطور گسترده در مدل درخواست و پاسخ بكار مي‌رود. اول، يك كلاينت يك تقاضا(هر دو آنها درخواست براي يك فايل يا احضار يك برنامه مي‌كنند) به سرور مي فرستد. دوم،‌ سرور درخواست را پردازش مي‌كند و يك پاسخ به كلاينت برمي‌گرداند.

بعد از اين، ارتباط بين كلاينت و سرور رها شده و فراموش مي شود. HTTP بدون تابعيت است چونكه سرور HTTP با هر درخواست مستقل از درخواستهاي قبلي رفتار مي‌كند. به هر حال، چندین برنامه کاربردی نوشته شده بر روی وب نیاز به استفاده از نگهداری حالت دارند.

براي مثال، بيشتر برنامه‌هاي خريد آنلاین نياز دارند كه ردپايي از خريد مشتريان خود را نگه دارند. برنامه‌هاي كاربردي وب اغلب كوكي‌ها را براي نگهداري وضعيت استفاده مي‌كنند. يك كوكي، يك كمي از اطلاعاتي كه سوابق وضعيتي يك كلاينت است. زماني كه يك سرور نياز دارد كه برخي از اطلاعات وضعيت براي يك كلاينت را به ياد بياورد، سرور يك كوكي مي سازد كه شامل اطلاعات وضعيت است و آن را به كلاينت مي فرستد.كلاينت سپس كوكي را يا روي حافظه ياروي ديسك سخت ذخيره مي‌كند. سپس مشتری کوکی را به هر زیر درخواست خود الحاق می کند. بيشتر برنامه هاي كاربردي وب(از قبيل تجارت الكترونيك) نياز به يك پروتكل كوكي ايمن دارند. يك پروتكل كوكي ايمن كه بين يك كلاينت ويك سرور راه اندازي مي‌شود، نياز دارد كه چهار خدمات زير را فراهم كند: : تعيين اعتبار، قابليت اعتماد، تماميت، ضد تکرار.

1. تعيين اعتبار:

يك پروتكل كوكي ايمن بايد اجازه دهد، سرور رسيدگي كند كه كلاينت در يك دوره زماني معين تصديق شده است.علاوه بر اين، هيچ كوكي نبايد قادر به جعل يك كوكي معتبر باشد.

در برنامه هاي كاربردي وب ايمن، يك نوع session بين يك كلاينت ويك سرور از دو مرحله تشكيل شده است.مرحله اول معروف به مرحله‌ي login است و مرحله دوم معروف به مرحله‌ي subsequent-requests است.

**مرحله‌ي login:**

در اين مرحله، كلاينت و سرور دو طرف هديگر را تصديق مي‌كنند.از يك طرف، كلاينت با استفاده از گواهينامه PKI

(Public Key Infrastructure) سرور ،سرور را تصديق مي‌كند بعد از آن او يك اتصالSSL

 (Secure Sockets Layer)با سرور برقرار مي‌كند.از طرف ديگر، سرور با استفاده نام و رمز كاربر كلاينت ، كلاينت را تصديق مي‌كند و يك كوكي ايمن( اين نيز به يك"نشانه تصديق" يا يك "تاييد كننده اعتبار" در مقالات پيشين معروف است) به كلاينت مي‌فرستد.

**مرحله subsequent-requests:**

در اين مرحله، كلاينت كوكي ايمن را همراه با هر درخواست به سرور مي‌فرستد. سرور رسيدگي مي‌كند كه آيا كوكي معتبر است، و اگر هست، درخواست ها را سرويس مي‌دهد.

1. قابليت اعتماد:

محتواي يك كوكي ايمن فقط براي سروری كه مي‌خواند معني مي‌دهد. دو سطح از قابليت اعتماد وجود دارد كه يك پروتكل كوكي ايمن قادر به انجام دادن است: قابليت اعتماد درسطح پايين تر و قابليت اعتماد در سطح بالاتر.

1. قابليت اعتماد در سطح پايين:

يك پروتكل كوكي ايمن با قابليت در سطح پايين از همه بجز سرور وكلاينت براي خواندن محتواي يك كوكي جلوگيري مي‌كند.براي دست يافتن به قابليت اعتماد سطح پايين، يك پروتكل كوكي ايمن معمولا روي SSL راه اندازي مي‌شود.بخاطر داشته باشيد SSL با استفاده از كليد جلسه هر پيام بين كلاينت و سرور رمز مي كند كه فقط كلاينت و سرور مي‌فهمند.در اين مقاله، ما فرض مي‌كنيم كه همه پروتكل كوكي ايمن روي SSL راه اندازي مي‌شوند.

1. قابليت اعتماد در سطح بالا:

يك پروتكل كوكي ايمن با قابليت اعتماد در سطح بالا جلوگيري مي‌كند از همه به جز سرور براي خواندن اطلاعات مهم با كوكي كه سرور نمي‌خواهد براي كلاينت آشكار شود. براي مثال، محتويات كوكي ممكن است شامل برخي از اطلاعات كلاينت از قبيل دسته بندي داخلي آنها يا خريدوفروش اعتباري،كه سرور ممكن است نخواهد كلاينت از آنها آگاه باشد.

برنامه هاي كاربردي وب مختلف ممكن است نياز به سطح هاي مختلف قابليت اعتماد داشته باشند.بنابراين، يك پروتكل كوكي امن بايد قادر به پشتيباني از قابليت در سطح پايين يا قابليت اعتماد در سطح بالا داشته باشد

1. تماميت:

يك پروتكل كوكي امن بايد اجازه بدهد به سرور تا كشف كند كه آيا يك كوكي تغيير داده شده است؟

1. ضدتكرار:

در صورتي كه يك حمله كننده تكرار كند يك كوكي دزديده شده را، يك پروتكل كوكي امنيتي بايد قادر باشد تا تشخيص بدهد كه آن كوكي نامعتبر است.

در غير اينصورت، مهاجم به عنوان مشتري اي شناخته خواهد شد كه كوكي تكرار شده براي آنجا صادر شده بود.

در طراحي يك پروتكل امنيتي كوكي، علاوه بر نيازمندي هاي بالا، ما همچنين به اعتبار براي صدور كوكي و توانايي گسترش پذيري نياز داريم. با توجه به اعتبار شركت ها، يك پروتكل امنيتي كوكي، بايد نيازمندي هاي يك سروري را فراهم كند كه جستجو در پايگاه داده براي اعتبارسنجي يك كوكي را انجام مي دهد و بايد كليد عمومي رمز گذاري را فراهم كند.

نامه اي كه پايگاه داده جستجو مي كند به طور چشم گيري كندتر از سرعتي است كه يك سرور زمان مي برد تا يك كوكي را تاييد اعتبار كند. با توجه به گسترش شركت، يك پروتكل امنيتي كوكي بايد فراهم كند نيازمندي هاي يك مشتري را تا يك كليد عمومي و كليد خصوصي بگيرد، كه اين به طور پيش فرض غير عملي است. چندين پروتكل امنيتي پيشنهاد شده اند[2,5,8,10]. به هر حال، هيچ كدام از اين پروتكل ها به طور رضايت‌بخشي كامل نيستند.

كوكي پروتكل [5] دارد سه تا نقطه ضعف دارد:

1. اين نمي‌تواند يك مكانيسمي براي تامين سطح بالايي از قابليت اعتماد فراهم كند
2. در معرض حمله تكرار كوكي قرار دارد
3. اين مكانيسم براي دفاع از حملات پر حجم ناكار آمد است و مقياس پذير نيست.

سه مكانيسم تشخيص هويت در پروتكل كوكي [8] بي‌اثر هستند يا براي برنامه نویسی مشکل اند. پروتكل‌های كوكي در [2] و [10] نیز بيهوده هستند، زيرا آنها به پايگاه داده‌اي براي جستجو در تاييد اعتبار يك كوكي نياز دارند.

در اين مقاله، ما يك پروتكل امنيتي كوكي كه مفيد، موثر و راحت براي برنامه نویسی است را پيشنهاد مي دهيم. بر حسب كارايي، پروتكل امنيتي كوكي ما همه چهار سرويس امنيتي نام برده شده در بالا را فراهم مي‌كند. بر حسب درجه كارايي، پروتكل كوكي امنيتي ما شامل هيچ پايگاه داده جستجوگري يا كليد عمومي رمزنگاري نيست. بر حسب درجه گسترش پذيري، پروتكل كوكي امنيتي مي‌تواند به آساني روي وب سرور موجود گسترش پيدا كند، و اين هیچ وابستگي به تغيير روي كوكي های اينترنتي مخصوص ندارد [7].

ادامه اين مقاله حاصل زير است:

در قسمت 2، ما پروتكل امنيتي كوكي مان را به تفضيل نمايش مي دهيم.

در قسمت 3، ما پياده سازي پروتكل امنيتي كوكي و بازدهي آن را بحث مي كنيم.

در قسمت 4، ما پروتكل هاي كوكي موجود را بررسي و آزمايش مي كنيم.

ما آخرين توضيحات را در قسمت 5 مي دهيم.

1. پروتكل امنيتي كوكي

جديدترين پروتكل امنيتي كوكي ارائه شده توسط Fu et al. طبق [5]. در اين قسمت، ما اول اين پروتكل را بررسي مي‌كنيم، كه ما آن را به عنوان پروتكل كوكي Fu’s ارجاع داديم.

ما نشان مي دهيم كه اين پروتكل سه مساله مهم دارد، و ما يك راه حل به هر يك از آنها مي دهيم. نهايتا، ما پروتكل امنيتي كوكي خودمان را ارايه مي‌دهيم. علايم استفاده شده در اين قسمت در جدول زير ليست شده اند.



چکیده پیام استفاده شده در تشخيص هويت استفاده شده در اين مقاله فرض مي شود قابل تایید و مستحکم هستند: مي گيرد با استفاده از پيغام m و كليد k، HMAC(m,k) محاسبه مي‌شود. به هر حال، HMAC(m,k) را بدست می آوریم که محاسبه پيغام m و كليد k امکان ناپذیر است. مثالهايي برای پیغام های چکیده شده تشخیص هویت از قبیل MAC-MD5 و HMAC-SHA1 هستند [1, 4, 6, 9].

كليد سرور (i.e., sk) يك كليد امنيتي است كه فقط سرور آن را مي‌داند.

* 1. پروتكل امنيتي Fu’s

پروتكل كوكي نشان داده شده در شكل 1



در اين پروتكل كوكي، يك كوكي امنيتي بوسيله يك سرور برای یک مشتری صادر مي شود که شامل چهار مورد زیر در مقادیر فیلدهای کوکی HTTP است.

1. نام کاربری: مقدار این فیلد منحصر به فرد است برای تشخیص هویت افراد.
2. تاریخ انقضا: مقدار این فیلد نشان می دهد که کوکی چه موقع منقضی می‌شود و سرور نباید آن را بپذیرد.
3. اطلاعات: مقدار این فیلد می‌تواند هر چیزی که سرور می‌خواهد به یاد بیارد باشد. حالت اطلاعات ارتباط بین مشتری و سرور معمولا در این فیلد ذخیره می‌شود.
4. HMAC: مقدار این فیلد چکیده درهم ریخته شده سه مقدار بالا است.

پروتکل Fu's سه عیب امنیتی اصلی دارد. اول، یک مکانیسم برای تامین محرمانگی سطح بالا ندارد. دوم، این در مقابل حمله تکرار کوکی آسیب پذیر است. سوم، این مکانیسم برای دفاع کردن در مقابل حملات ظرفیت بالا بی‌‌اثر است و مقیاس پذیر نیست. سپس ما تک تک این سه نقطه ضعف را توضیح می‌دهیم و یک راه حل امنیتی برای هر کدام از آنها ارائه می دهیم.

* 1. مساله یک: اعتبار کوکی

ما بحث کردیم که بعضی از صفحات کاربردی وب به سطح بالایی از اعتبار برای کوکی های خود نیاز دارند. برای فراهم کردن سطح بالای اعتبار، یک پروتکل کوکی امن باید اطلاعات فیلدهای کوکی را رمزنگاری کند.

حالا سوال این است: سرور چه کلیدی را باید برای رمزنگاری استفاده کند؟

پروتکل Fu's به این سوال یک پاسخ مناسب در اختیار قرار نمی‌دهد.

فقط و فقط یک کلید مبهم در پروتکل Fu's وجود دارد که کلید سرور نامیده می شود.

یک راه حل درست استفاده از این کلید سرور، رمزنگاری هرکدام از اطلاعات فیلدهای کوکی هست.

به هرحال، این راه حل ایمن نیست.

یک حمله کننده می تواند به عنوان یک مشتری قانونی ثبت نام کند و سپس تعداد زیادی از کوکی‌ها را که توسط سرور فرستاده می‌شوند را گرد آوری می‌کند.

اگر فیلدهای اطلاعاتی همه این کوکی‌ها بوسیله همان کلید سرور رمزنگاری شوند، شاید حمله کننده بتواند بوسیله آنالیز نوشته رمز شده کلید رمز را پیدا کند. اگر چه آنالیز رمز مشکل سخت است، اما از روی احتیاط باید از چنین امکانی جلوگیری کنیم.

پروتکل کوکی پیشنهاد شده در [10] کلیدهای رمزنگاری مخصوص کوکی را طبق زیر مدیریت می‌کند. پایگاه داده یک سرور نام کاربری و کلید رمزنگاری شده هر یک از کاربران را نگهداری می‌کند. وقتی یک سرور یک کوکی برای یک مشتری ایجاد می کند، سرور یک کلید تصادفی جدید برای رمزنگاری کوکی تولید می کند و کلیدهای قدیمی مرتبط با مشتری را در این پایگاه داده با این کلید جدید جایگزین می‌کند. وقتی یک سرور یک کوکی رمزنگاری شده از یک مشتری را دریافت می‌کند، سرور از نام کاربری مشتری استفاده می کند تا در پایگاه داده برای کلید متناظر با آن نام را جستجو کند. این راه حل 2 ضرر عمده دارد: اول، این به خاطر سربار جستجو در پایگاه داده به طور زیادی کم بازده است. دوم، وقتی کلید قدیمی مشتری پاک بشود، همه کوکی‌هایی که بوسیله کلید قدیمی رمزنگاری شده‌اند بی‌ارزش می‌شوند. این خیلی می‌تواند مخرب باشد. شاید یک مشتری همان کوکی‌های رمز شده را به بیش از یک درخواست به یک سرور الحاق کند و بعد از دریافت اولین درخواست، سرور یک کوکی جدید و کلید رمزنگاری جدید برای مشتری ایجاد می‌کند، که حاصل همه درخواست‌های دیگر با همان کوکی بوسیله سرور غیرمجاز می‌شود.

پروتکل کوکی پیشنهاد شده در [8] کلیدهای رمزنگاری مخصوص کوکی را به شیوه‌های مختلف مدیریت می‌کند: وقتی یک سرور یک کوکی را در یک سطح بالای اعتبار ایجاد می‌کند، سرور یک کلید تصادفی تولید می‌کند و کلید را با کلید عمومی همان سرور رمزنگاری می کند، و کلید رمز شده را در کوکی خودش به جای یک پایگاه داده در سمت سرور ذخیره می کند. تایید هر کوکی شامل کلید عمومی رمزنگاری باعث می شود پروتکل کوکی پیچیده و غیر موثر بشود.

راه حل ما برای این مساله ساده و موثر است. ما پیشنهاد می کنیم از HMAC(نام کاربری|تاریخ انقضا) به عنوان کلید رمزنگاری استفاده شود. این راه حل سه خاصیت خوب زیر را دارد: اول، کلید رمزنگاری منحصر به فرد برای هر کوکی متفاوت وجود دارد، به دلیل نام کاربری و تاریخ انقضای متفاوتی که در کوکی قرار می‌گیرد. دوم، کلید رمزنگاری مستحکم است، زیرا کلید سرور به طور امن نگهداری می‌شود. سوم، کلید رمزنگاری هر کوکی به هیچ ذخیره سازی سمت سرور یا در داخل یک کوکی نیاز ندارد، ترجیحا این بوسیله یک سرور پویا محاسبه می شود.

* 1. مساله دو: حملات تکرار

پروتکل کوکی Fu's آسیب پذیر در مقابل حملات تکرار است، که حمله کننده‌ها می‌توانند در دو قدم زیر شروع به کار کنند.

قدم اول دزدیدن یک کوکی است که یک سرور برای یک مشتری دیگر صادر می‌کند. شاید یک حمله کننده چندین راه برای دزدین یک کوکی از اشخاص دیگر داشته باشد. برای مثال، اگر یک مشتری کوکی را در هارد دیسک ذخیره کند، حمله کننده می‌تواند آن را سرقت بوسیله استفاده از تروجان ها، کرمها یا ویروسها کند.

یک حمله کننده شاید سرقت کند یک کوکی را بوسیله فعال کردن یک حمله Denning-Sacco [8].

در چنین حمله‌ای، یک حمله کننده اول تعداد زیادی از پیغام‌هایی را که بوسیله همان کلید جلسه SSL رمزنگاری شده‌اند گرد آوری می کند، و سپس کلید جلسه مورد استفاده را به شیوه‌های مختلفی بدست می‌آورد. در گام دوم یک حمله تکرار، حمله کننده یک ارتباط SSL با سرور برقرار می کند و یک کوکی دزدیده شده را که هنوز منقضی نشده است تکرار می کند. بنابراین، سرور به طور نادرست حمله کننده را به عنوان یک مشتری شناسایی میکند، و به حمله کننده اجازه می دهد تا به حساب کاربری مشتری دسترسی داشته باشد.

برای شمارش حملات تکرار، ما پیشنهاد می کنیم تا کلید جلسه SSL را به چکیده پیغام تشخیص هویت اضافه کنید تا از HMAC(نام کاربری| تاریخ انقضا| اطلاعات| کلید جلسه) به عنوان چکیده پیغام تشخیص هویت هر کوکی استفاده کنیم. بنابراین یک کوکی برای خودش کلید جلسه مخصوص دارد. حتی اگر یک حمله کننده هم یک کوکی را بدزدد، او نمی تواند آن را به طور موفق تکرار کند، زیرا کلید جلسه فقط مشتری قانونی را که سرور کوکی او را ایجاد کرده، معرفی می‌کند.

* 1. مساله سه: حملات حجیم

با استفاده از پروتکل کوکی Fu's، یک سرور از همان کلید سرور برای محاسبات HMAC هر کوکی استفاده می‌کند. این باعث می‌شود تا در مقابل حملات حجیم به صورت بالقوه آسیب پذیر بشود. برای شروع کردن به حملات حجیم، یک حمله کننده اول تعداد زیادی از HMAC های پیغام های متفاوتی را جمع آوری می کند که هر HMAC به وسیله همان کلید محاسبه ‌شده است. سپس حمله کننده کلید را به وسیله متدهای مختلف کشف می‌کند. از بین الگوریتم‌ها تاکنون HMAC-MD5 و SHA1 گمان نمی‌رود که برای حملات حجیم زیان آور باشند [6].

راه حل پیشنهاد شده در [5] برای دفاع در برابر حملات حجیم این است که در هر دوره کلیدش تغییر می‌کند. به هرحال، این راه حل هست بیهوده است و مقیاس پذیر نیز نیست. اولا، هر بار که کلید سرور تغییر می کند، سرور نیاز به استفاده از هر دو کلید اصلی و جدید برای بازبینی کوکی ها در یک دوره مقطعی دارد. اجازه بدهید t را به عنوان زمانی که کلید سرور از k به k' تغییر می کند مشخص کنیم، و ∆ نیز بزرگترین تاریخ انقضایی را که سرور برای یک کوکی قبل از زمان t صادر کرده است مشخص می‌کند. بنابراین، در زمان val [t; t + ∆]، سرور دریافت کند کوکی‌های قانونی را که با استفاده از هردو کلیدهای قدیمی (k) و کلید جدید (k) ایجاد شده‌اند. بنابراین، اگر کوکی منقضی نشده باشد و در تایید اعتبار بوسیله کلید جدید شکست بخورد، سرور از کلید قدیمی برای تایید اعتبار کوکی استفاده می کند. به طور واضح، تایید اعتبار دوباره کوکی نیاز به دو کلید متفاوت دارد که این بی‌فایده است. دوما، برای اطمینان از اینکه کلید جدید به یک کلید تصادفی که تاکنون از آن استفاده نشده باشد، برنامه سرور به طور دستی به روزآوری و دوباره چک کردن کلید در طول زمان تغییر کلید می‌پردازد.

راه حل ما برای این مساله موثر و مقیاس پذیر است. ما پیشنهاد می‌دهیم تا استفاده کنید از کلید رمزگذاری، با نام HMAC(user name|expiration time)، به عنوان کلید در محاسبه HMAC هر کوکی. این راه حل سه خصوصیت خوب زیر را دارا است: اول، همچنانکه در قسمت 2.2 بحث شد، این کلید به دلیل نام کاربری و تاریخ انقضای متفاوت برای هر کوکی جدید منحصر به فرد است. دوم، این کلید به دلیل کلید سرور مستحکم است. سوم، به دلیل اینکه این کلید همان رمزنگاری شده قبلی است، محاسبه HMAC(user name|expiration time, sk) فقط نیاز دارد تا یکبار انجام بشود.

* 1. پروتکل کوکی ایمن ما

پروتکل کوکی امن ما در شکل شماره دو نشان داده شده است. (data)k نشانگر اطلاعات رمزنگاری شده با کلید K و sk نشانگر کلید سرور است.

توجه کنید که همه چهار فیلد نام کاربری و تاریخ انقضا و (data)k و

 HMAC(user name|expiration time|data|session key, k) مقادیری هستند در فیلدهای مشخص کوکی [7]. دو فیلد نام کاربری و تاریخ انقضا به فرم text ساده هستند، زیرا سرور نیاز دارد تا از آنها در محاسبه HMAC(user name|expiration time, sk) استفاده کند. توجه کنید که فیلد

HMAC(user name|expiration time|data|session key, k) به وسیله پروتکل ما برای تعیین کردن تشخیص هویت، جامعیت و ضد حملات تکرار صادر می شود.

پروتکل کوکی ما می تواند شکل بدهد به فراهم کردن سطح بالا یا پایینی از اعتبار. چیزی که در شکل 2 نمایش داده شد، تامین می کند سطح بالایی از اعتبار را. اگر سطح پایینی از اعتبار کوکی مطلوب باشد، یکی از فیلدها را می توان به صورت ساده و رمز نشده رها کرد و ... جای گزین کنید (data)k را با text ساده.

پردازش برای تایید اعتبار کوکی ایجاد شده بوسیله پروتکل کوکی ما در شکل 3 نمایش داده شده است. توجه کنید که اگر FALSE برگردانده شود، کوکی نامعتبر فرض می شود و مشتری باید دوباره وارد بشود با استفاده از نام کاربری و کلمه عبورش.



1. اجرا کردن و بازدهی

در این قسمت ما بحث می کنیم درباره پیاده سازی و کارایی ناشی از پروتکل کوکی ایمن خودمان. برای اندازه گیری کارایی پروتکل کوکی ایمن خودمان در مقایسه با پروتکل کوکی Fu's، ما برنامه نویسی کردیم چهار ابزار زیر را در PHP:

1. پروتکل کوکی Fu's با درجه اعتبار پایین
2. پروتکل کوکی ایمن خودمان با درجه امنیت پایین
3. پروتکل کوکی Fu's با درجه اعتبار بالا
4. پروتکل کوکی ایمن خودمان با درجه امنیت بالا

چون پروتکل کوکی Fu's فراهم نمی کند قابلیت اعتماد بالایی را برای ما، برای ارزیابی کارایی آن، ما فرض می کنیم که اطلاعات فیلدهای کوکی آن با کلید سرور رمزنگاری شده اند. علاوه بر فراهم کردن یک خط پایه ای برای مقایسه، ما نیز اجرا می کنیم یک پروتکل کوکی نا امن را کوکی ها HMAC ندارد یا اینکه رمزنگاری نمی شوند و ...هر کوکی فقط شامل سه فیلد به صورت ساده است: نام کاربری، تاریخ انقضا، و دیتا.

اینجا ما بحث می کنیم بعضی از جزئیات اجرای پروتکل های کوکی های بالا را. ما استفاده می کنیم از HMAC-SHA1 به عنوان تابع HMAC. ما استفاده می کنیم از الگوریتم Rijndael-256 از طریق ECB به عنوان مدل برای الگوریتم رمزنگاری. کلید سرور شامل 160 بیت است. خروجی HMAC شامل 320 بیت است.

محیط تست ما شامل یک وب سرور سطح منوسط است که استفاده می کند از یک پردازنده 2.4 GH Celeron و از رم 512 MB و سیستم عامل Windows 2003 Standard Edition و IIS 6.0 و PHP 4.3.10 و MySQL 3.23 و یک مشتری که استفاده می کند از از یک پنتیوم 4 با یک پردازنده 2.8 GH و 512 MB رم و از سیستم Red Hat 3.0. . سرور و کلاینت به هم متصل هستند با یک ارتباط ختصاصی گیگابیت.

ما اجرا کردیم پنج پروتکل کوکی را روی ارتباط SSL بین مشتری و سرور. برای هر پروتکل، مشتری ایجاد می کند 10،000 درخواست پی در پی به سرور، به طوری که کوکی درخواست ها هستند یک کوکی حمله کننده. ما اندازه گیری در طرف مشتری، متوسط زمانی را از وقتی که درخواست فرستاده می شود تا وقتی که جواب درخواست برسد. در اجراهای ما، سرور ایجاد می کند یک کوکی جدید بعد از دریافت یک کوکی معتبر. به عبارت دیگر، رکورد پایان به پایانی که ما اندازه گیری کردیم هست شامل:

1. زمانی که انتقال پیدا می کند یک درخواست با یک کوکی از مشتری به یک سرور.
2. زمانی که سرور تایید اعتبار می کند کوکی درخواستی را.
3. زمانی که سرور ایجاد می کند یک کوکی جدید و
4. زمانی که کوکی جدید پاسخ انتقال می یابد از سرور به طرف مشتری.

وری هم رفته اندازه کوکی هست شامل HTTP header که به طور متوسط 1KB است.نتایج آزمایش در شکل 4 نشان داده شده است.



از اطلاعات موجود در شکل 4 متوجه می شویم که بازدهی پروتکل کوکی ایمن ما هست خیلی نزدیک به طرح پروتکل کوکی Fu's، در حالی که پروتکل کوکی ما امنیت خیلی بهتری را ایجاد می کند.

1. کارهای وابسته

در این قسمت، ما امتحان می کنیم پروتکل های کوکی امنیتی قبل را و مقایسه می کنیم آنها را با پروتکل کوکی ایمن خودمان.

در مورد [5] بحث شد در قسمت دو، پروتکل کوکی Fu's دارد سه مشکل اصلی: این ندارد یک مکانیسم برای فراهم آوردن یک سطح بالایی از اعتبار، این هست آسیب پذیر در مقابل حملات تکرار و این مکانیسم برای دفاع در مقابل حجیم غیرموثر هست و مقیاس پذیر نیست.

در [8]، Park و Sandhu پیشنهاد دادند یک پروتکل کوکی که استفاده می کند از یک مجموعه ای از عناصر داخلی کوکی از قبیل نام کوکی و زمان عمر کوکی و یک کلمه عبور کوکی و یک مهر کوکی و ... . همینطور بحث شد در قسمت 2.2، در مورد مکانیسمی که برای تامین اعتبار کوکی دراین پروتکل هست بیهوده. بعد، ما بررسی کردیم سه مکانیسم پیشنهاد شده برای تشخیص هویت را در [8]: تشخیص هویت برپایه آدرس، کلمه عبور و امضای دیجیتالی. در تشخیص هویت بر پایه آدرس، هرکوکی دارد یک IP کوکی که شامل IP مشتری می شود. یک سرور تشخیص هویت، کوکی دریافت شده را بررسی می کند که آیا آدرس IP این کوکی، برابر با IP کوکی است یا نه. این مکانیسم تشخیص هویت دارد سه مساله اصلی. اول، این هست آسیب پذیر در مقابل IPهای قلابی. دوم، یک IP مشتری شاید به طور دینامیک تغییر کند. سوم، یک مشتری شاید استفاده کند از NAT(انتقال دهنده آدرس شبکه) یا یک سرور نماینده و بنابراین شاید سهیم بشود همان آدرس IP خودش را با مشتری های دیگر در همان دامین. در استفاده از تشخیص هویت برپایه کلمه عبور، هر مجموعه کوکی یک کلمه عبور کوکی دارد که شامل پیغام نماینده ای از کلمه عبور مشتری است. یک سرور تشخیص هویت دریافت می کند یک مجموعه کوکی را و سپس تایید اعتبار می کند آن را که آیا ارزش کلمه عبور کوکی هست درست یا نه، که نیاز دارد به جستجو در پایگاه داده. استفاده از تشخیص هویت بر پایه امضای دیجیتال، هر زمان یک مشتری بخواهد تا ایجاد کند یک HTTP برای سرور، مشتری اول باید تولید کند یک امضای کوکی که شامل مهرزمانی و امضای مشتری برای مهرزمانی است. دوم اینکه، مشتری می فرستد درخواست HTTP را با مجموعه کوکی صادر شده بوسیله سرور برای مشتری و امضای کوکی ایجاد شده بوسیله کوکی. سرور تشخیص هویت می کند یک کوکی رسیده را بوسیله تایید اعتبار کوکی. این مکانیسم تایید اعتبار هست مشکل، زیرا این تصور می کند که هر کوکی دارد یک کلید عمومی و یک کلید خصوصی. علاوه بر آن، این مکانیسم تشخیص هویت هست گران قیمت، زیرا به هردوی پایگاه داده(برای کلید عمومی مشتری) و کلید عمومی برای رمزنگاری.

در [10] Xu et al. ارائه کرد یک پروتکل کوکی که هست استفاده می کند از یک سرور برای ذخیره اطلاعات کارت اعتباری هر مشتری. همانطوری که بحث شد در قسمت 2.2، این پروتکل نمی تواند کاملا به طور صحیح تایید کند چند درخواست همزمان را با همان کوکی. به علاوه، این پروتکل هست غیر مفید به دلیل ترافیک پایگاه داده به خاطر جستجو و به روزآوری برای تایید هر کوکی.

در [2]، Blundo et al. مطرح کرد یک تشخیص هویت وب را که استفاده می کند از کوکی های رمزنگاری شده. یکی از جنبه های این پروتکل کوکی این هست که نیاز دارد به یک سرور تا انجام بدهد جستجو را برای تایید اعتبار کوکی رسیده.

1. نتیجه گیری