جواب سوال ۱

پاسخ بخش آ:

آ. مراحل تبادل کلید بین آزاده و بهرام در پروتکل X3DH :

در پروتکل DH۳X که بخشی از پروتکل Signal است، تبادل کلید به صورت زیر انجام می‌شود:

کلیدهای بلندمدت:

- **آزاده (A)** : دارای یک جفت کلید بلندمدت (IKA و SKA) است که کلید عمومی IKA و کلید خصوصی SKA هستند.
- **بهرام (B)** : نیز دارای جفت کلید بلندمدت (IKB و SKB) می‌باشد.

کلیدهای پیش‌نویس (Prekeys) :

- **بهرام (B)** : علاوه بر کلیدهای بلندمدت، یک کلید Signed PreKey (SPKB) دارد که با کلید بلندمدت خود امضا شده است. همچنین مجموعه‌ای از کلیدهای یک‌بار مصرف (OPKB) را در سرور منتشر می‌کند.

فرآیند تبادل کلید:

آزاده (A) :

الف) ابتدا اطلاعات کلیدهای عمومی بهرام (IKB ، SPKB و OPKB) را از سرور دریافت می‌کند.

ب) یک کلید موقت (EKA) تولید می‌کند.

ج) محاسبات دیفی‌ـ هلمن را با استفاده از کلیدهای دریافتی انجام می‌دهد:

$$DH۱ = DH(IKA, SPKB)$$

$$DH۲ = DH(EKA, IKB)$$

$$DH۳ = DH(EKA, SPKB)$$

$$DH۴ = DH(EKA, OPKB)$$

د) از یک تابع مشتق کلید (KDF) برای ترکیب نتایج دیفی‌ـ هلمن و تولید کلید مشترک (SK) استفاده می‌کند:

$$SK = KDF(DH۱||DH۲||DH۳||DH۴)$$

ه) پیام اولیه شامل IKA ، EKA و شناسه‌های OPKB مورد استفاده را به بهرام ارسال می‌کند.

بهرام (B) :

الف) پیام اولیه از آزاده را دریافت می‌کند.

ب) با استفاده از اطلاعات دریافتی و کلیدهای خود، همان محاسبات دیفی‌ـ‌هلمن را انجام داده و کلید مشترک SK را محاسبه می‌کند.

نتیجه:

هر دو طرف، آزاده و بهرام، به یک کلید مشترک SK دست می‌یابند که برای رمزنگاری ارتباطات بعدی استفاده می‌شود.

پاسخ بخش ب:

ب. اجزای قابل تعویض و اجزای ثابت در پروتکل X3DH :

در پروتکل X3DH ، برخی اجزا به طور منظم تعویض می‌شوند تا امنیت سیستم افزایش یابد، در حالی که برخی اجزا باید ثابت باقی بمانند تا هویت کاربران حفظ شود.

اجزای قابل تعویض:

- **کلیدهای PreKey یک‌بار مصرف (OPKB) :** این کلیدها تنها برای یک نشست خاص استفاده می‌شوند و پس از استفاده حذف می‌گردند. تعویض مکرر OPKB ها از بازپخش و استفاده مجدد توسط مهاجم جلوگیری می‌کند.
- **کلید PreKey Signed (SPKB) :** این کلید به طور دوره‌ای (مثلاً هر چند هفته یک بار) تعویض می‌شود تا از خطر فاش شدن یا سوء استفاده از کلیدهای قدیمی جلوگیری شود و امنیت ارتباطات آینده تضمین شود.

اجزای ثابت:

- **کلیدهای هویت بلندمدت (IKA و IKB) :** این کلیدها باید ثابت باقی بمانند تا کاربران بتوانند هویت یکدیگر را به طور معتبر اثبات کنند. تغییر این کلیدها می‌تواند فرآیند احراز هویت را مختل کند و اعتماد کاربران را تحت تاثیر قرار دهد.

چرا برخی اجزا باید ثابت باشند؟

- **حفظ هویت کاربران:** کلیدهای هویت بلندمدت برای تأیید هویت کاربران ضروری هستند. اگر این کلیدها تغییر کنند، دیگر امکان تأیید هویت کاربر به درستی وجود نخواهد داشت.
- **یکپارچگی امنیت:** تعویض دوره‌ای OPKB و SPKB باعث می‌شود که حتی اگر یک کلید به خطر بیفتد، تنها ارتباطات آینده تحت تاثیر قرار گیرند و ارتباطات گذشته همچنان امن باقی بمانند.

پاسخ بخش ج:

ج. اجزایی که اصالت دوجانبه را تضمین می‌کنند و راهکارهای مقابله با حمله شخص میانی:

تضمین اصالت دوجانبه:

- **کلیدهای هویت بلندمدت (IKA و IKB) :** این کلیدها برای امضای دیجیتال پیام‌ها استفاده می‌شوند. هر کاربر با استفاده از کلید خصوصی خود پیام‌ها را امضا کرده و طرف مقابل با استفاده از کلید عمومی طرف دیگر صحت امضا را بررسی می‌کند.
- **کلید PreKey Signed (SPKB) :** این کلید با کلید هویت بلندمدت امضا شده است، که اطمینان می‌دهد SPKB از طرف صاحب اصلی آن صادر شده و قابل اعتماد است.

راهکارهای مقابله با حمله شخص میانی (Man-in-the-Middle) :

- **بررسی دقیق امضاهای دیجیتال:** کاربران باید تمامی امضاهای دیجیتال را به دقت بررسی کنند تا از صحت و اعتبار آن‌ها اطمینان حاصل کنند.
- **استفاده از پروتکل‌های امن‌تر مانند TLS :** افزودن لایه‌های امنیتی اضافی می‌تواند از نفوذ مهاجم در مسیر ارتباط جلوگیری کند.
- **احراز هویت چندعاملی:** استفاده از روش‌های احراز هویت اضافی مانند رمزهای یک‌بار مصرف یا گواهی‌های دیجیتال می‌تواند امنیت ارتباط را افزایش دهد و از حملات شخص میانی جلوگیری کند.

اگر سرور معتمد نباشد:

- **پروتکل‌های احراز هویت قوی:** کاربران می‌توانند از پروتکل‌هایی که نیاز به تأییدیه‌های اضافی دارند استفاده کنند تا از صحت هویت طرف مقابل اطمینان حاصل کنند.
- **تبادل کلید به صورت مستقیم:** در صورت امکان، تبادل کلید به صورت مستقیم بین کاربران بدون استفاده از سرور می‌تواند از حملات شخص میانی جلوگیری کند.

پاسخ بخش د:

د. اجزایی که ویژگی‌های Forward Secrecy و Backward Secrecy را تضمین می‌کنند و روش‌های انجام آن‌ها:

Forward Secrecy (محرمانگی پیشرو):

- **کلیدهای یک‌بار مصرف (OPKB) :** استفاده از OPKB ها که تنها برای یک نشست خاص استفاده می‌شوند، تضمین می‌کند که حتی اگر یک کلید جلسه فعلی فاش شود، ارتباطات قبلی همچنان امن باقی می‌مانند.
- **کلید موقت (EKA) :** تولید کلید موقت برای هر نشست جدید باعث می‌شود که هر نشست دارای کلید جدید و مستقل باشد، که از دسترسی به ارتباطات گذشته جلوگیری می‌کند.

Backward Secrecy (محرمانگی پسرو):

- **تولید کلیدهای جدید برای هر نشست:** با تولید کلیدهای موقت جدید برای هر جلسه و حذف کلیدهای قدیمی پس از استفاده، ارتباطات آینده نیز از دسترسی مهاجمان محافظت می‌شود.
- **حذف کلیدهای قدیمی پس از استفاده:** با حذف کلیدهای یک‌بار مصرف پس از استفاده، مهاجم نمی‌تواند از کلیدهای قدیمی برای رمزگشایی ارتباطات جدید استفاده کند.

چگونگی تضمین این ویژگی‌ها:

- **ترکیب کلیدهای موقت و یک‌بار مصرف:** پروتکل X3DH همراه با پروتکل Double Ratchet با استفاده از ترکیب کلیدهای موقت و یک‌بار مصرف، اطمینان می‌دهد که هر نشست جدید دارای کلید جدید و مستقل است.
- **حذف کلیدها پس از استفاده:** با حذف کلیدهای یک‌بار مصرف و کلیدهای موقت پس از استفاده، پروتکل از ایجاد نقاط ضعف برای مهاجمان جلوگیری می‌کند و امنیت ارتباطات را تضمین می‌نماید.

پاسخ بخش ه:

ه. مدیریت پیام‌های از دست رفته در پروتکل X3DH و Double Ratchet :

در شرایطی که برخی پیام‌ها به مقصد نرسند، پروتکل‌های X3DH و Double Ratchet از روش‌های مختلفی برای مدیریت این وضعیت استفاده می‌کنند تا امنیت و تداوم ارتباطات حفظ شود:

استفاده از کلیدهای یک‌بار مصرف جدید (OPKB) :

در صورت از دست رفتن پیام‌ها، کاربران می‌توانند از OPKB های جدید برای ادامه ارتباطات استفاده کنند. این امر تضمین می‌کند که پیام‌های بعدی امن باقی می‌مانند و پیام‌های از دست رفته تاثیری بر امنیت ارتباط ندارند.

پروتکل Double Ratchet :

این پروتکل با استفاده از فرآیندی دو مرحله‌ای برای تبادل کلید، اطمینان می‌دهد که حتی اگر برخی پیام‌ها از دست بروند، پیام‌های بعدی بدون مشکل ارسال و دریافت می‌شوند. همچنین، مکانیزم‌های همگام‌سازی و حفاظت در برابر ارسال مجدد پیام‌ها (Replay Protection) به حفظ امنیت ارتباطات کمک می‌کنند.

پیام‌های ناهمگام (Out-of-Order Messages):

سیستم قادر است پیام‌های ناهمگام را شناسایی کرده و آن‌ها را به ترتیب صحیح مرتب‌سازی کند تا ارتباط به درستی ادامه یابد.

بازسازی ارتباط:

در صورت از دست رفتن پیام‌های کلیدی یا پیام‌های اصلی، کاربران می‌توانند مجدداً کلیدهای جدید تولید کرده و ارتباط را بازسازی کنند تا از ادامه امنیت ارتباط اطمینان حاصل شود.

نتیجه‌گیری:

پروتکل‌های X3DH و Double Ratchet با استفاده از کلیدهای یک‌بار مصرف و مکانیزم‌های همگام‌سازی، توانایی مدیریت پیام‌های از دست رفته را دارند و از تداوم امنیت و محرمانگی ارتباطات اطمینان می‌دهند.

پاسخ بخش و:

و. اهمیت OPKB و نقش SPKB در امنیت پروتکل:

اهمیت OPKB (One-Time PreKey) :

- **افزایش امنیت:** OPKBها تنها برای یک نشست خاص استفاده می‌شوند و پس از استفاده حذف می‌گردند، که این امر از استفاده مجدد و بازپخش آن‌ها توسط مهاجم جلوگیری می‌کند.
- **مقاومت در برابر حملات :Replay** با استفاده از OPKBهای یک‌بار مصرف، ارسال مجدد پیام‌ها یا استفاده از کلیدهای قدیمی برای نفوذ به ارتباطات جدید غیرممکن می‌شود.

نقش SPKB (Signed PreKey) در امنیت پروتکل:

- **احراز هویت:** SPKBها با استفاده از کلید هویت بلندمدت امضا شده‌اند، که این امر اطمینان می‌دهد SPKBها از طرف صاحب اصلی خود صادر شده و قابل اعتماد هستند.
- **پشتیبانی از Forward Secrecy:** SPKBها به طور دوره‌ای تعویض می‌شوند و همراه با OPKBها، ویژگی‌های Forward Secrecy را تضمین می‌کنند، زیرا کلیدهای جدید امنیت ارتباطات جدید را حفظ می‌کنند حتی اگر کلیدهای قبلی فاش شوند.

اهمیت تأمین OPKB و نقش SPKB در امنیت مکالمه:

- **تأمین OPKB :** تامین کافی از OPKBها اطمینان می‌دهد که همیشه یک کلید یک‌بار مصرف برای برقراری ارتباط جدید در دسترس باشد. کمبود OPKBها ممکن است منجر به استفاده مجدد از OPKBهای قبلی شود که امنیت ارتباطات را کاهش می‌دهد.
- **نقش SPKB :** SPKBها به عنوان کلیدهای امضا شده توسط کلید هویت بلندمدت، نقش مهمی در احراز هویت و تضمین اصالت ارتباطات دارند. این کلیدها مانع از ایجاد SPKBهای جعلی توسط مهاجم می‌شوند و امنیت ارتباطات را حفظ می‌کنند.

تأثیر کمبود OPKB و SPKB بر امنیت مکالمه:

- **کمبود OPKB :** ممکن است کاربران مجبور به استفاده مجدد از OPKBهای قبلی شوند که این امر می‌تواند امنیت ارتباطات را تهدید کند، زیرا کلیدهای تکراری ممکن است توسط مهاجم شناسایی و مورد سوء استفاده قرار گیرند.
- **کمبود SPKB :** نبود SPKBهای جدید می‌تواند احراز هویت کاربران را مختل کند و مهاجم ممکن است بتواند حملات جعل هویت را انجام دهد. همچنین، نبود SPKBهای جدید ویژگی‌های Forward Secrecy را تحت تاثیر قرار می‌دهد و امنیت ارتباطات جدید را کاهش می‌دهد.

پاسخ بخش ز:

ز. نیاز آزاده به انتظار آنلاین شدن بهرام برای تبادل کلید در صورت آفلاین بودن بهرام:

خیر، آزاده نیازی ندارد که منتظر آنلاین شدن بهرام بماند تا تبادل کلید انجام شود. دلیل این امر استفاده از کلیدهای پیش‌تعیین شده و ذخیره شده توسط بهرام در سرور است.

توضیح:

- **کلیدهای PreKey از قبل آپلود شده:** بهرام قبلاً مجموعه‌ای از کلیدهای یک‌بار مصرف (OPKB) و کلیدهای Signed PreKey (SPKB) را به سرور Signal آپلود کرده است.
- **دریافت کلیدها توسط آزاده:** هنگامی که آزاده قصد ارسال پیام به بهرام را دارد، می‌تواند این کلیدهای یک‌بار مصرف را از سرور دریافت کند بدون نیاز به آنلاین بودن بهرام.
- **ارسال پیام اولیه:** آزاده با استفاده از این کلیدهای دریافت شده، پیام اولیه خود را به بهرام ارسال می‌کند. وقتی بهرام آنلاین می‌شود، پیام اولیه را دریافت و پردازش می‌کند و ارتباط رمزگذاری شده را برقرار می‌کند.

مزایا:

- **ایجاد ارتباط حتی در شرایط آفلاین:** این روش امکان ارسال پیام‌ها به کاربران آفلاین را فراهم می‌کند بدون اینکه ارسال‌کننده نیاز به انتظار داشته باشد.
- **افزایش انعطاف‌پذیری و کارایی:** کاربران می‌توانند در هر زمانی با ارسال پیام‌های امن ارتباط برقرار کنند بدون نیاز به هماهنگی زمان‌های آنلاین.

پاسخ بخش ح:

ح. انکارپذیری در پروتکل سیگنال و تأثیر آن بر اصالت پیام‌ها:

تعریف انکارپذیری:

انکارپذیری به معنای این است که فرستنده پیام نمی‌تواند سپس ادعای نکند که آن پیام را ارسال کرده است. در پروتکل سیگنال، این ویژگی به گونه‌ای پیاده‌سازی شده که فرستنده نمی‌تواند بعداً از ارسال پیام انکار کند.

تضاد انکارپذیری با اطمینان از اصالت پیام‌ها:

- **اصالت پیام‌ها:** اصالت پیام‌ها تضمین می‌کند که پیام از طرف فرستنده واقعی ارسال شده است.
- **انکارپذیری:** در حالی که اصالت پیام‌ها را تضمین می‌کند، انکارپذیری باعث می‌شود که فرستنده نمی‌تواند پس از ارسال پیام آن را انکار کند.

جزئیات بیشتر:

انکارپذیری توسط هر دو طرف: در پروتکل سیگنال، هر دو طرف مکالمه می‌توانند پیام‌ها را انکار کنند زیرا پیام‌ها با استفاده از کلیدهای موقت امضا می‌شوند که تنها طرف مقابل قادر به تأیید آن‌هاست. این امر باعث می‌شود که هیچ‌کدام از طرفین نتوانند بعداً ادعا کنند که پیام خاصی را ارسال نکرده‌اند.

ساخت مکالمات جعلی توسط اشخاص ثالث: فرد ثالثی مانند سودابه می‌تواند بدون داشتن هیچ پیام واقعی بین آزاده و بهرام، مجموعه‌ای از پیام‌های جعلی را ایجاد کند که به نظر می‌رسد از طرف یکی از کاربران ارسال شده‌اند. از آنجا که پیام‌های جعلی نیز با استفاده از کلیدهای یک‌بار مصرف و موقت ایجاد می‌شوند، کاربران نمی‌توانند به سادگی تشخیص دهند که این پیام‌ها واقعی هستند یا جعلی.

کاربردهای انکارپذیری در دنیای واقعی:

- **حفظ حریم خصوصی:** انکارپذیری به کاربران اجازه می‌دهد که پس از ارسال پیام‌ها، دیگر نتوانند آن‌ها را به‌طور قانونی برای اثبات ارسالشان مورد استفاده قرار دهند، که این امر حریم خصوصی را تقویت می‌کند.
- **پیشگیری از استفاده نادرست:** در مواردی که افراد ممکن است تحت فشار قانونی قرار گیرند تا پیام‌های خاصی را ارسال کنند، انکارپذیری به آن‌ها امکان می‌دهد که از الزام به ارسال پیام‌های ناخواسته جلوگیری کنند.
- **تسهیل ارتباطات آزاد و امن:** کاربران می‌توانند با اطمینان بیشتر و بدون نگرانی از ثبت دائمی پیام‌ها، ارتباطات خود را برقرار کنند.

نتیجه‌گیری:

اگرچه انکارپذیری ممکن است در برخی موارد تضاد با تأیید اصالت پیام‌ها داشته باشد، اما در پروتکل سیگنال این دو ویژگی به گونه‌ای پیاده‌سازی شده‌اند که همزمان حفظ حریم خصوصی کاربران و امنیت ارتباطات امکان‌پذیر باشد. انکارپذیری در دنیای واقعی به کاربران کمک می‌کند تا ارتباطات خود را بدون ترس از سوء استفاده‌های قانونی یا ثبت دائمی پیام‌ها حفظ کنند.

در نتیجه پروتکل X3DH و Double Ratchet در پروتکل سیگنال با استفاده از ترکیب کلیدهای بلندمدت، موقت و یک‌بار مصرف، امنیت ارتباطات را از طریق ویژگی‌هایی مانند Forward Secrecy و Backward Secrecy تضمین می‌کنند. همچنین، با مدیریت دقیق تبادل کلید و جلوگیری از حملات شخص میانی، اصالت و امنیت ارتباطات حفظ می‌شود. اهمیت OPKB و SPKB در افزایش امنیت و جلوگیری از حملات Replay و جعل هویت نقش کلیدی دارد. علاوه بر این، ویژگی انکارپذیری در پروتکل سیگنال با حفظ حریم خصوصی و جلوگیری از سوء استفاده‌های قانونی، به کاربران امکان می‌دهد تا ارتباطات خود را به صورت امن و مطمئن برقرار کنند.

جواب سوال ۲

بخش اول

وقتی یک پیام m ابتدا توسط الگوریتم فشرده‌سازی $\mathrm{C(m)}$ فشرده می‌شود و سپس با استفاده از رمزنگاری بلوکی با اندازه بلوک ۸ بیت پد شده و رمزگذاری می‌گردد، طول نهایی پیام رمزگذاری شده مستقیماً به طول پیام فشرده شده $\mathrm{C(m)}$ بستگی دارد. مهاجم که تنها تعداد بلوک‌های ارسال شده را مشاهده می‌کند، می‌تواند اطلاعاتی درباره طول $\mathrm{C(m)}$ به دست آورد. از آنجایی که طول $\mathrm{C(m)}$ نشان‌دهنده تعداد و طول رشته‌های ۱ متوالی در پیام اصلی m است، مهاجم می‌تواند استنتاج‌هایی درباره الگوی تکراری در m انجام دهد.

با یک مثال درباره‌ی این مورد بیشتر توضیح می‌دهم. **مثال:**

پیام $m_۱$: ۱۱۱۱۱۱۱۱۱۱۰۱۱۱۱۱۰۱۰۱۱۱

$C(m_۱) =$ ۱۰۱۰۰۱۰۱۰۰۰۱۰۰۱۱

طول $C(m_۱) =$ ۱۶ بیت $\rightarrow$ ۲ بلوک ۸ بیتی

پیام $m_۲$: ۱۱۱۱۱۰۱۱۱۱۱۰۱۱۱۱۱۰۱۱۱۱

$C(m_۲)$ ممکن است طول بیشتری داشته باشد (فرضاً ۲۴ بیت) $\rightarrow$ ۳ بلوک ۸ بیتی

در این حالت، اگر مهاجم مشاهده کند که دو بلوک ارسال شده‌اند، احتمال می‌دهد پیام $m_۱$ با الگوی تکراری مناسب فشرده شده است. اگر سه بلوک ارسال شود، ممکن است پیام $m_۲$ با الگوی کمتر تکراری فشرده شده باشد.

مهاجم می‌تواند از تفاوت‌های فشرده‌سازی بین دو پیام با طول اصلی برابر استفاده کند تا تشخیص دهد کدام پیام رمزگذاری شده است. اگر دو پیام $m_۱$ و $m_۲$ دارای طول اصلی یکسان باشند اما الگوهای متفاوتی از رشته‌های ۱ متوالی داشته باشند، فشرده‌سازی آن‌ها نیز متفاوت خواهد بود. پیام با الگوی فشرده‌تر (بیشتر رشته‌های تکراری و طولانی‌تر ۱ها) منجر به $C(m)$ کوتاه‌تر و در نتیجه تعداد بلوک‌های کمتر خواهد شد.

با یک مثال درباره‌ی این مورد بیشتر توضیح می‌دهم. **مثال:**

پیام $m_۱$: ۱۱۱۱۱۱۱۱۱۱۰۱۱۱۱۱۰۱۰۱۱۱

$C(m_۱) =$ ۱۰۱۰۰۱۰۱۰۰۰۱۰۰۱۱

تعداد بلوک‌های رمزگذاری شده: ۲ بلوک

پیام $m_۲$: ۱۱۱۱۱۰۱۱۱۱۱۰۱۱۱۱۱۰۱۱۱۱

$C(m_۲) =$ ۱۰۱۰۱۰۱۰۰۱۰۱۰۱۰۱

تعداد بلوک‌های رمزگذاری شده: ۳ بلوک

در این حالت، اگر مهاجم مشاهده کند که تعداد بلوک‌های رمزگذاری شده ۲ بلوک است، نتیجه می‌گیرد که پیام $m_۱$ رمزگذاری شده است. اگر ۳ بلوک مشاهده کند، نتیجه می‌گیرد که پیام $m_۲$ رمزگذاری شده است.

این روش به مهاجم اجازه می‌دهد تا با تحلیل طول پیام‌های رمزگذاری شده و مقایسه آن‌ها با الگوهای مختلف فشرده‌سازی، اطلاعاتی درباره محتوای پیام اصلی بدست آورد. این نوع حمله نشان‌دهنده آسیب‌پذیری‌های احتمالی در ترکیب فشرده‌سازی و رمزنگاری است که پس از حملات CRIME در پروتکل‌های جدید مانند TLS 1.3 حذف شده‌اند تا از چنین تهدیداتی جلوگیری شود.

بخش دوم

در این بخش، مهاجم توانایی افزودن بیت‌های دلخواه به ابتدای پیام‌های اصلی را دارد و با انجام این کار، می‌خواهد تعداد ۱های رشته اول پیام m را کشف کند. برای این منظور، مهاجم از ویژگی‌های الگوریتم فشرده‌سازی بی‌اتلاف $\mathrm{C(m)}$ استفاده می‌کند که تعداد ۱ های هر رشته را به صورت یک مقدار ۴ بیتی نمایش می‌دهد و محدودیت‌هایی در تعداد ۱ های متوالی دارد.

روش حمله

افزودن بیت‌های کنترل‌شده به ابتدای پیام

مهاجم می‌تواند به تعداد دلخواه بیت‌های ۱ یا ۰ را به ابتدای پیام m اضافه کند. هدف این است که با تنظیم تعداد بیت‌های ۱ اضافه شده، تغییراتی در فشرده‌سازی رخ دهد که به کمک آن بتوان تعداد ۱های رشته اول m را تعیین کرد.

تعیین نقطه تغییر در فشرده‌سازی

از آنجا که الگوریتم فشرده‌سازی $\mathrm{C(m)}$ فقط تا پانزده ۱ متوالی را به درستی فشرده می‌کند و اگر تعداد ۱ ها بیشتر از ۱۵ شود، فقط پانزده ۱ اول را فشرده می‌کند و بقیه را نادیده می‌گیرد، مهاجم می‌تواند با افزایش تدریجی تعداد ۱ های اضافه شده، نقطه‌ای را پیدا کند که در آن تعداد ۱ های متوالی به ۱۵ برسد.

تحلیل تعداد بلوک‌های رمزگذاری شده

پس از افزودن بیت‌های کنترل‌شده و فشرده‌سازی پیام، پیام فشرده شده به بلوک‌های ۸ بیتی تقسیم می‌شود و رمزگذاری می‌گردد. تعداد بلوک‌های رمزگذاری شده به طور مستقیم به طول پیام فشرده شده بستگی دارد. مهاجم با مشاهده تعداد بلوک‌ها می‌تواند تشخیص دهد که آیا تعداد ۱ های متوالی افزوده شده به m به ۱۵ رسیده یا نه.

گام‌های دقیق حمله

ابتدا یک سری فرضیات اولیه داریم.

- فرض کنید پیام اصلی m با k عدد ۱ در ابتدای آن شروع می‌شود.
- مهاجم قصد دارد تعداد ۱های k را تعیین کند.

افزودن بیت‌های ۱ کنترل‌شده مهاجم به ترتیب از $t = ۰$ تا $t = ۱۵$ بیت ۱ به ابتدای پیام m اضافه می‌کند، یعنی پیام‌های زیر را ارسال می‌کند:

- $t = ۰ : m$
- $t = ۱ : ۱m$
- $t = ۲ : ۱۱m$
- ...
- $t = ۱۵ : ۱۱۱۱۱۱۱۱۱۱۱۱۱۱۱m$

مشاهده تعداد بلوک‌های رمزگذاری شده برای هر مقدار t ، مهاجم تعداد بلوک‌های رمزگذاری شده را مشاهده می‌کند.

تجزیه و تحلیل تغییرات

- اگر $t + k \leq$ ۱۵ باشد: تعداد ۱های متوالی در پیام فشرده شده برابر با $t + k$ است و فشرده‌سازی به طور بهینه انجام می‌شود.
- اگر $t + k >$ ۱۵ باشد: فشرده‌سازی فقط پانزده ۱ اول را در نظر می‌گیرد و بقیه نادیده می‌گیرد، که منجر به افزایش طول پیام فشرده شده و در نتیجه افزایش تعداد بلوک‌های رمزگذاری شده می‌شود.

تعیین تعداد ۱ های اولیه (k)

مهاجم به دنبال کمترین مقداری از t است که در آن $t + k >$ ۱۵ می‌شود. این نقطه تغییر نشان‌دهنده آن است که $k = ۱۶ - t$. بنابراین، با یافتن مقداری از t که در آن تعداد بلوک‌های رمزگذاری شده افزایش می‌یابد، مهاجم می‌تواند مقدار k را محاسبه کند.

حالا برای مثال بدین شکل داریم:

فرض کنید پیام اصلی m با $k =$ ۱۰ عدد ۱ شروع می‌شود:

افزودن $t =$ ۰ بیت ۱

- پیام تغییر یافته: m (۱۰ عدد ۱ متوالی)
- تعداد ۱های فشرده شده: ۱۰
- تعداد بلوک‌های رمزگذاری شده: به عنوان مثال، ۲ بلوک

افزودن $t =$ ۵ بیت ۱

- پیام تغییر یافته: m۱۱۱۱۱ (۵ + ۱۰ = ۱۵ عدد ۱ متوالی)
- تعداد ۱های فشرده شده: ۱۵
- تعداد بلوک‌های رمزگذاری شده: همچنان ۲ بلوک

افزودن $t =$ ۶ بیت ۱

- پیام تغییر یافته: m۱۱۱۱۱۱ (۶ + ۱۰ = ۱۶ عدد ۱ متوالی)
- تعداد ۱های فشرده شده: فقط ۱۵ عدد ۱ اول
- تعداد بلوک‌های رمزگذاری شده: افزایش به ۳ بلوک

با مشاهده افزایش تعداد بلوک‌ها زمانی که $t =$ ۶ است، مهاجم می‌داند که $t+k =$ ۱۶ از این رو $k =$ ۱۶$-$۶ $=$ ۱۰

نتیجه‌ی بخش دوم

با استفاده از افزودن بیت‌های کنترل‌شده به ابتدای پیام و تحلیل تغییرات در تعداد بلوک‌های رمزگذاری شده، مهاجم قادر است تعداد ۱ های متوالی در ابتدای پیام اصلی m را به طور دقیق تعیین کند. این روش نشان‌دهنده آسیب‌پذیری ترکیب فشرده‌سازی و رمزگذاری در پروتکل‌های امنیتی مشابه TLS 1.2 است که منجر به حملاتی مانند CRIME شده است. به همین دلیل، در نسخه‌های جدیدتر مانند TLS 1.3 ، قابلیت فشرده‌سازی حذف شده تا از چنین حملاتی جلوگیری شود.

بخش سوم

حمله CRIME (Compression Ratio Info-leak Made Easy) یک حمله جانبی (side-channel attack) علیه پروتکل‌های امنیتی مانند TLS است که از ویژگی فشرده‌سازی داده‌ها سوء استفاده می‌کند. در این حمله، مهاجم با استفاده از قابلیت فشرده‌سازی و مشاهده تغییرات در طول داده‌های رمزگذاری‌شده، می‌تواند اطلاعات حساس مانند کوکی‌های کاربر را به‌دست آورد.

شرایط لازم برای موفقیت حمله CRIME

فعال بودن فشرده‌سازی در ارتباط TLS

پروتکل TLS باید قابلیت فشرده‌سازی داده‌ها را فعال کرده باشد. در نسخه‌های قدیمی‌تر مانند TLS 1.2 این ویژگی وجود دارد، در حالی که در نسخه‌های جدیدتر مانند TLS 1.3 به منظور جلوگیری از حملات مشابه CRIME، فشرده‌سازی حذف شده است.

توانایی مهاجم در تزریق داده به جریان اطلاعات

مهاجم باید بتواند داده‌هایی را به جریان اطلاعاتی که بین کاربر و سرور رد و بدل می‌شود، اضافه کند. این معمولاً از طریق ضعف‌های موجود در مرورگرها یا کاربردهای مبتنی بر وب انجام می‌گیرد.

قابلیت مشاهده طول داده‌های رمزگذاری‌شده

مهاجم باید بتواند طول داده‌های رمزگذاری‌شده را قبل و بعد از فشرده‌سازی مشاهده کند. این اطلاعات به او کمک می‌کند تا تغییرات در فشرده‌سازی را تحلیل کند.

نحوه انجام حمله CRIME

تزریق داده‌های کنترل‌شده

مهاجم داده‌های خاصی را به درخواست‌های HTTP ارسال می‌کند که در کنار کوکی‌های کاربر قرار می‌گیرند. هدف این است که داده‌های تزریق‌شده با بخش‌های مخفی (مانند کوکی‌ها) تکراری شوند.

تحلیل تغییرات در طول فشرده‌سازی

زمانی که داده‌های تزریق‌شده با کوکی‌ها تکراری باشند، الگوریتم فشرده‌سازی می‌تواند این تکرارها را به‌طور موثرتری فشرده کند، که منجر به کاهش طول داده‌های رمزگذاری‌شده می‌شود. مهاجم با مقایسه طول‌های مختلف داده‌های رمزگذاری‌شده برای حدس‌های مختلف، می‌تواند تشخیص دهد که کدام حدس‌ها صحیح هستند.

بازسازی کوکی‌ها به صورت مرحله‌ای

مهاجم به صورت تدریجی و با حدس‌های ترتیبی، بخش‌های مختلف کوکی را بازسازی می‌کند. هر بار که یک حدس صحیح باشد، کاهش قابل‌توجهی در طول داده‌های فشرده‌شده مشاهده می‌شود که به مهاجم نشان می‌دهد که آن بخش از کوکی صحیح حدس زده شده است.

حالا در یک مثال این مورد رو بررسی می‌کنم.

فرض کنید کوکی کاربر شامل رشته‌ای مخفی مانند session id=ABC123 است:

تزریق داده‌های کنترل‌شده توسط مهاجم

مهاجم داده‌هایی مانند A، AB، ABC و ... را به ابتدای درخواست‌های HTTP اضافه می‌کند.

تحلیل تغییرات طول داده‌های رمزگذاری‌شده

هر بار که یک بخش از حدس‌های مهاجم با کوکی واقعی مطابقت داشته باشد (مثلاً A با A یا AB با AB)، فشرده‌سازی بهینه‌تر انجام شده و طول داده‌های رمزگذاری‌شده کاهش می‌یابد.

بازسازی کامل کوکی

مهاجم با تحلیل این تغییرات طول، به تدریج می‌تواند کل کوکی را بازسازی کند.

حمله CRIME نشان‌دهنده آسیب‌پذیری‌های ترکیب فشرده‌سازی و رمزنگاری در پروتکل‌های امنیتی است. با حذف قابلیت فشرده‌سازی در نسخه‌های جدیدت

جواب سوال ۳

یکی از ویژگی‌های برجستهٔ nftables در مقایسه با iptables ، اتمیک بودن تغییرات در قوانین دیواره آتش است. منظور از اتمیک بودن این است که اعمال تغییرات در قوانین، به صورت یکپارچه و به طور کامل انجام می‌شود. به عبارت دیگر، در صورتی که عملیات تغییرات به هر دلیلی متوقف شود یا با شکست مواجه گردد، سیستم به حالت قبلی بازمی‌گردد و هیچ تغییری اعمال نمی‌شود.

این ویژگی از منظر امنیتی و پایداری شبکه بسیار اهمیت دارد، زیرا از ایجاد وضعیت‌های ناخواسته و آسیب‌پذیری در زمان به‌روزرسانی قوانین جلوگیری می‌کند.

در iptables ، قوانین به صورت جداگانه و به ترتیب اعمال می‌شوند. اگر در فرآیند اعمال یک سری از قوانین، خطا یا وقفه‌ای ایجاد شود، ممکن است برخی از قوانین اعمال شده و برخی دیگر اعمال نشده باشند. این وضعیت باعث ایجاد ناهماهنگی و آسیب‌پذیری در سیستم می‌شود.

اما در nftables ، تمامی تغییرات پیشنهادی ابتدا به صورت یک مجموعه در حافظه آماده می‌شوند و سپس در یک مرحله به طور کامل و یکجا بر روی سیستم اعمال می‌شوند. اگر به هر دلیلی این فرآیند با خطا مواجه شود، هیچ تغییری در قوانین فعلی دیواره آتش رخ نمی‌دهد و سیستم به حالت پایدار قبلی باقی می‌ماند.

حالا در این بخش این کامند را ابتدا ران می‌کنیم:

```
nc -l -k 127.0.0.1 8001
```

سرور آماده‌ی دریافت می‌شود.

```
*** System restart required ***
Last login: Fri Jan  3 01:06:22 2025 from 2.190.176.191
root@s697485:~# nc -l -k 127.0.0.1 8001
```

شکل ۱ : سرور آماده‌ی دریافت

حالا در ادامه فایل rules.nft را ایجاد می‌کنیم و قوانین مربوطه را در آن می‌نویسیم.

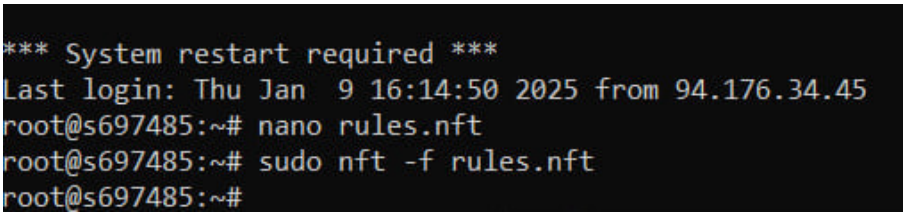
```
 GNU nano 7.2                                    rules.nf
#!/usr/sbin/nft -f

table ip nat {

    chain prerouting {
        type nat hook prerouting priority -100; policy accept;
        ip daddr 127.0.0.1 tcp dport 8000 dnat to 127.0.0.1:8001
    }

    chain output {
        type nat hook output priority -100; policy accept;
        ip daddr 127.0.0.1 tcp dport 8000 dnat to 127.0.0.1:8001
    }
}
```

شکل ۲ : فایل rules.nft

این کانفیگ شامل دو زنجیره در جدول NAT است که برای دستکاری آدرس‌های IP استفاده می‌شود. این تغییرات شامل دو زنجیره prerouting و output است که هر دو برای اعمال تغییرات مشابهی بر روی پکت‌های IP هستند:

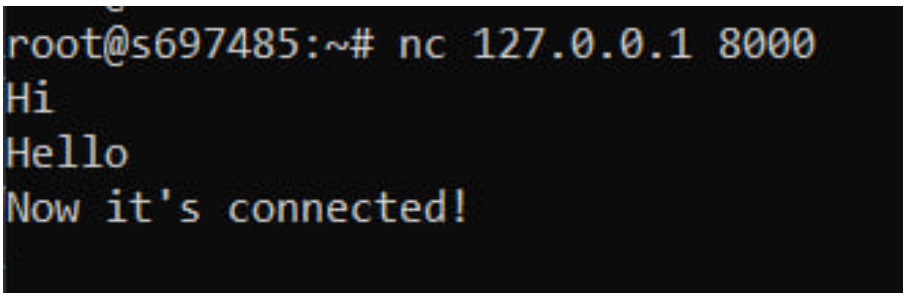
- **زنجیره prerouting**: این زنجیره قبل از روتینگ پکت‌ها اجرا می‌شود. به این ترتیب، تغییراتی که بر روی پکت‌ها اعمال می‌شود قبل از تعیین مسیر نهایی آن‌ها صورت می‌گیرد.
  - type nat hook prerouting priority -100; policy accept;:
    این خط مشخص می‌کند که این زنجیره از نوع NAT بوده و در نقطه اتصال prerouting با اولویت 100- قرار دارد. سیاست پیش‌فرض برای پکت‌هایی که مطابقت نمی‌یابند accept است، به این معنی که پکت‌ها اجازه عبور دارند.
  - ip daddr 127.0.0.1 tcp dport 8000 dnat to 127.0.0.1:8001:
    این قانون می‌گوید پکت‌هایی که به آدرس IP 127.0.0.1 و پورت TCP 8000 می‌آیند، به 127.0.0.1 در پورت 8001 منتقل شوند (DNAT) .
- **زنجیره output**: این زنجیره پس از تصمیم‌گیری برای روتینگ و قبل از خروج پکت‌ها از ماشین اجرا می‌شود.
  - type nat hook output priority -100; policy accept;:
    مشابه با زنجیره prerouting، این خط مشخص می‌کند که زنجیره در نقطه اتصال output و با اولویت 100- فعال است.
  - ip daddr 127.0.0.1 tcp dport 8000 dnat to 127.0.0.1:8001:
    همان تغییر آدرس و پورت که در prerouting اعمال شده است، در اینجا نیز تکرار می‌شود.

حالا این فایل را به عنوان ورودی به دستور nft می‌دهیم:

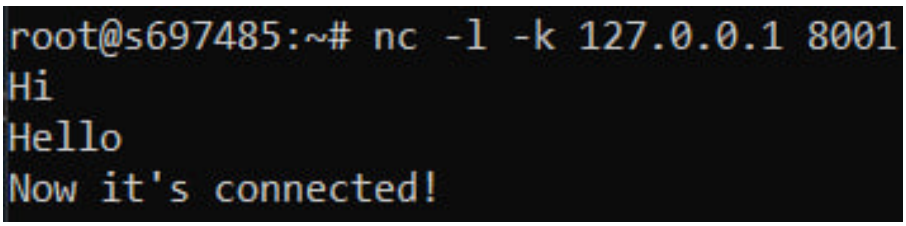


شکل ۳: ورودی دادن فایل

حالا ارتباط برقرار است و پیام‌های مدنظرمان را وارد می‌کنیم.



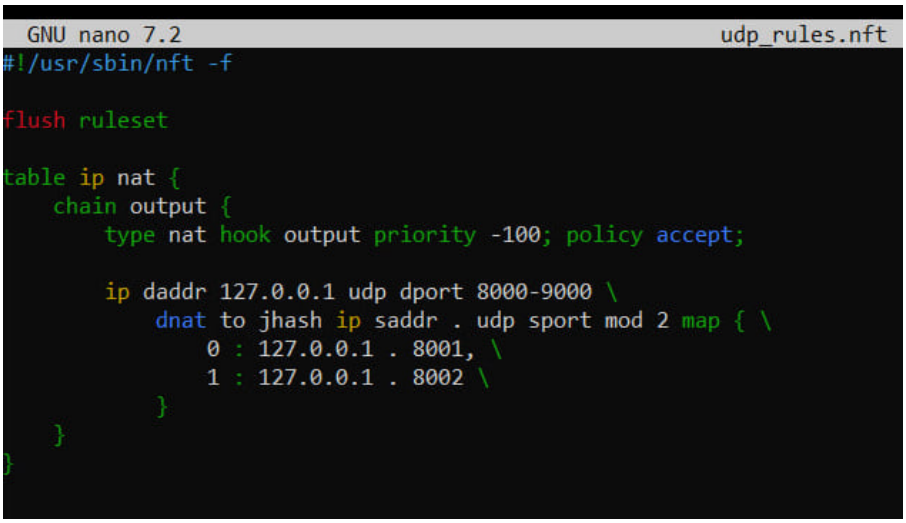
شکل ۴: تصویر سمت کارخواه



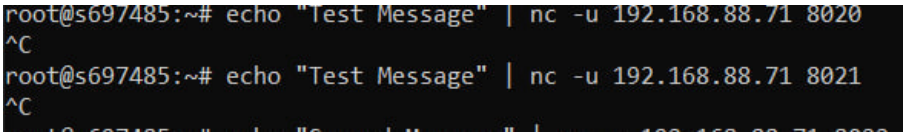
شکل ۵: تصویر سمت کارساز

حالا در ادامه قوانین را برای حالت UDP می‌نویسیم.

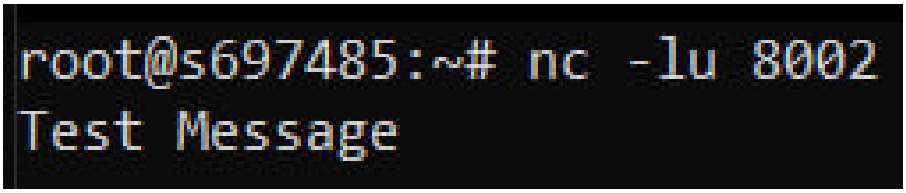
برای این قسمت باید برنامه netcat را به صورت همزمان بر روی درگاه‌های UDP ۸۰۰۱ و ۸۰۰۲ اجرا کنیم و دیواره آتش را طوری پیکربندی کنیم که بسته‌های UDP با مقصد درگاه‌های ۸۰۰۰ تا ۹۰۰۰ به طور یکنواخت به یکی از این دو درگاه ارسال شوند.



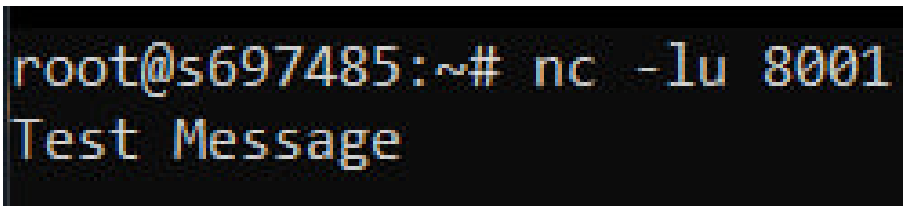
شکل ۶: قوانین UDP



شکل ۷: پیام‌های ارسالی



شکل ۸: پیام‌های دریافتی در یک پورت



شکل ۹: پیام‌های دریافتی در یک پورت دیگر

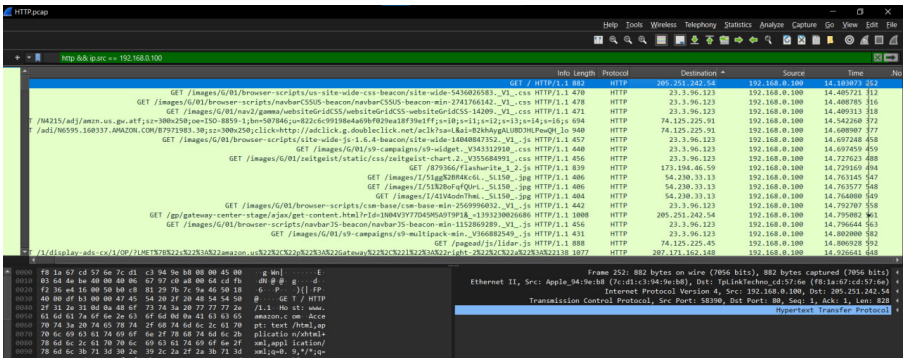
جواب سوال ۴

ترافیک HTTP

برای این بخش، در ابتدا این فیلتر را در وایرشارک اعمال می‌کنیم:

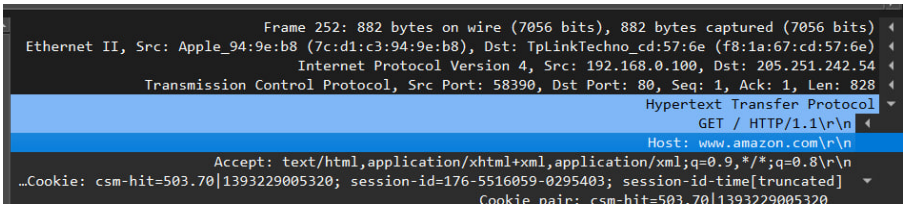
http && ip.src == 192.168.0.100

خروجی وایرشارک بدین شکل می‌شود:

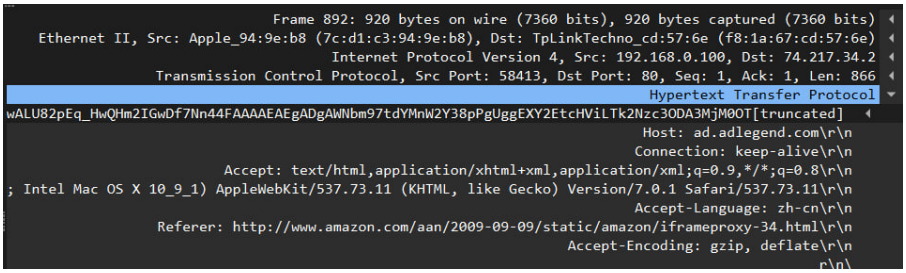


شکل ۱۰: خروجی Wireshark

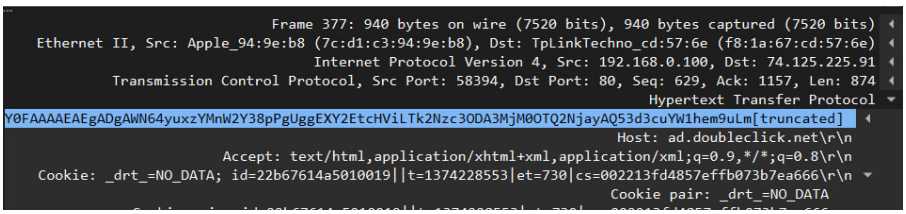
حالا ۳ تا از دامنه‌های بدست آمده این موارد هستند:



شکل ۱۱: دامنه: www.amazon.com ، آیپی: 205.251.242.54

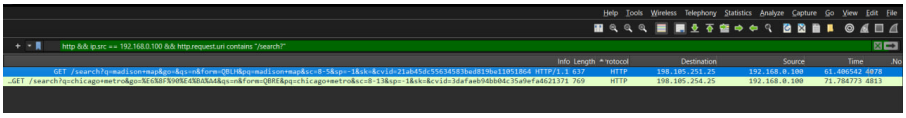


شکل ۱۲: دامنه: ad.adlegend.com ، آیپی: 74.217.34.2

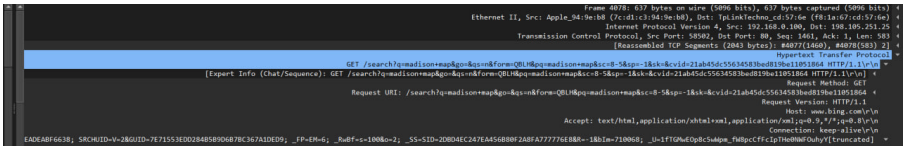


شکل ۱۳: دامنه: ad.doubleclick.net ، آیپی: 74.125.225.91

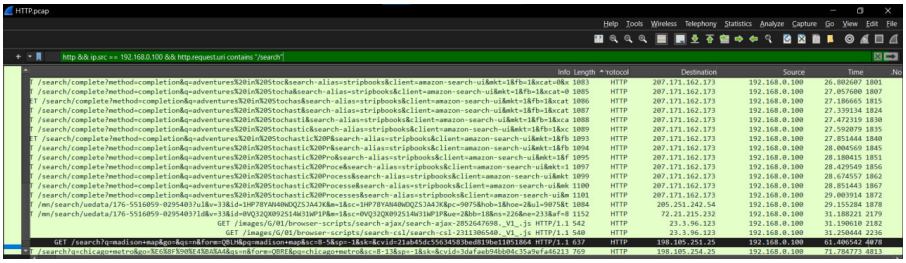
حالا در این بخش از ۲ کوئری استفاده می‌کنیم.



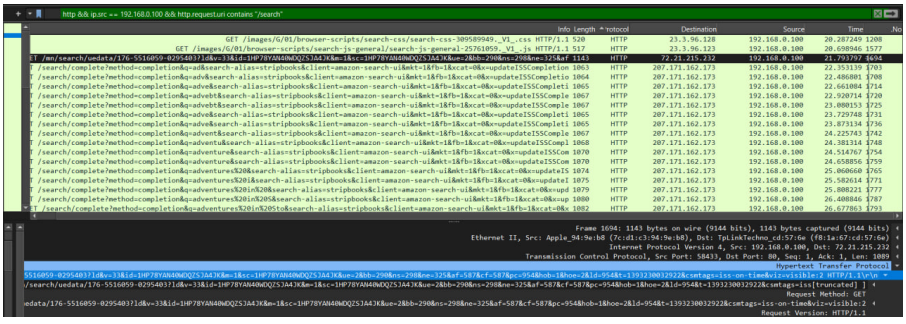
شکل ۱۴: خروجی کوئری http && ip.src == 192.168.0.100 && http.request.uri contains "/search?"



شکل ۱۵: اطلاعات کوئری



شکل ۱۶: خروجی کوئری http && ip.src == 192.168.0.100 && http.request.uri contains "/search"



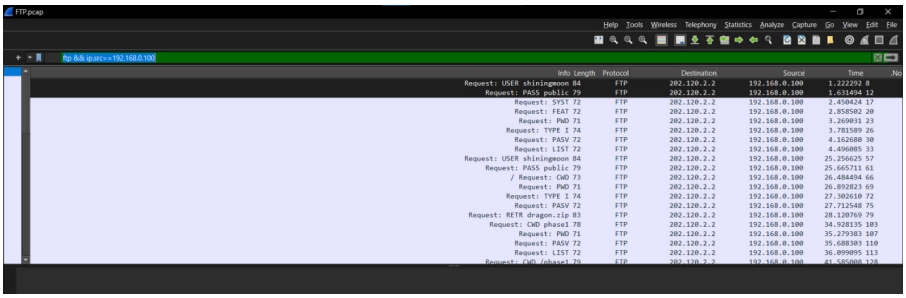
شکل ۱۷: اطلاعات کوئری

ترافیک FTP

برای این بخش، در ابتدا این فیلتر را در وایرشارک اعمال می‌کنیم:

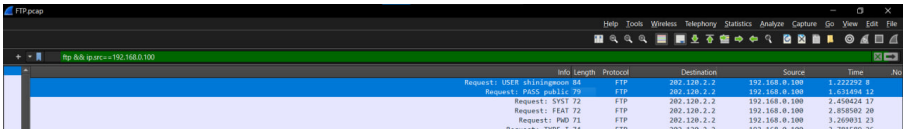
ftp && ip.src==192.168.0.100

خروجی وایرشارک بدین شکل می‌شود:



شکل ۱۸: خروجی Wireshark

پس اطلاعات مدنظر ما بدین شکل است:



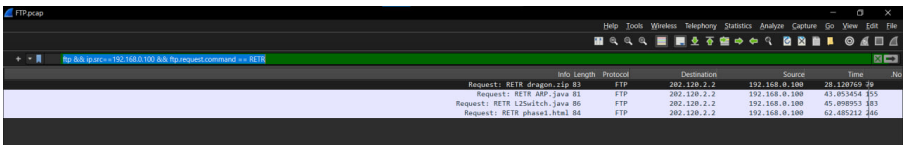
شکل ۱۹: نام کاربری shiningmoon و پسورد public

می‌دانیم که در پروتکل FTP برای دانلود یک فایل از سرور به کلاینت از کامند RETR استفاده می‌شود.

پس برای این بخش، این فیلتر را در وایرشارک اعمال می‌کنیم:

ftp && ip.src==192.168.0.100 && ftp.request.command == RETR

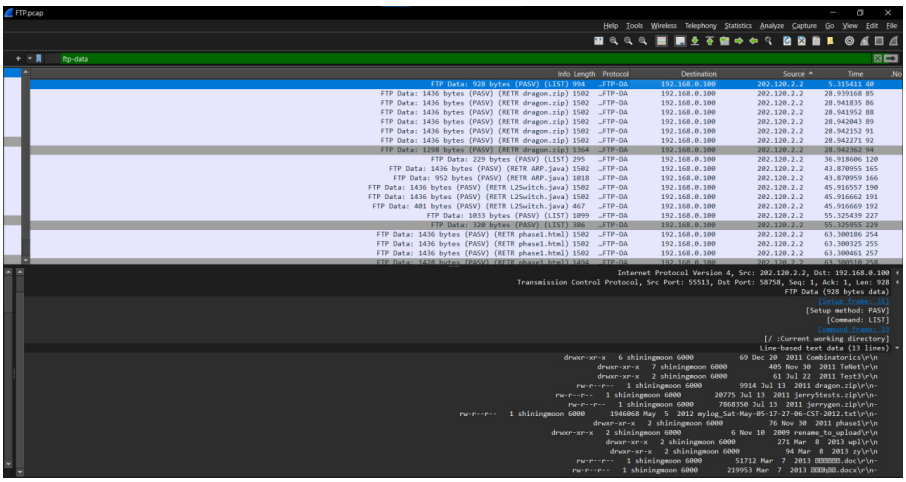
خروجی وایرشارک بدین شکل می‌شود:



شکل ۲۰: فایل‌های دانلود شده توسط کاربر از این سرور

لیست فایل‌ها:

phase1.html ، l2switch.java ، arp.java ، dragon.zip



شکل ۲۱: لیستی از فایل‌ها که سمت سرور قرار دارند

برای مثال ۲ فایلی که سمت سرور وجود دارد ولی کاربر دانلود نکرده:

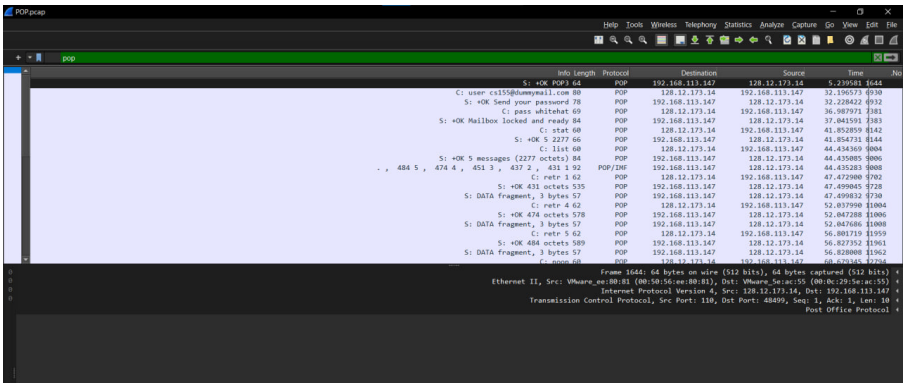
mylog_Sat-May-05-17-27-06-CST-2012.txt ، jerrygen.zip

ترافیک POP

برای این بخش، در ابتدا این فیلتر را در وایرشارک اعمال می‌کنیم:

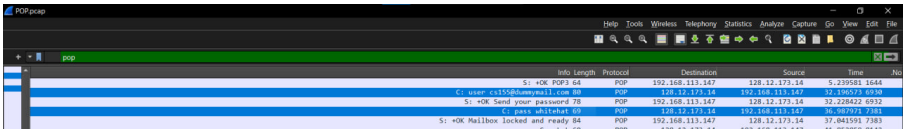
pop

خروجی وایرشارک بدین شکل می‌شود:



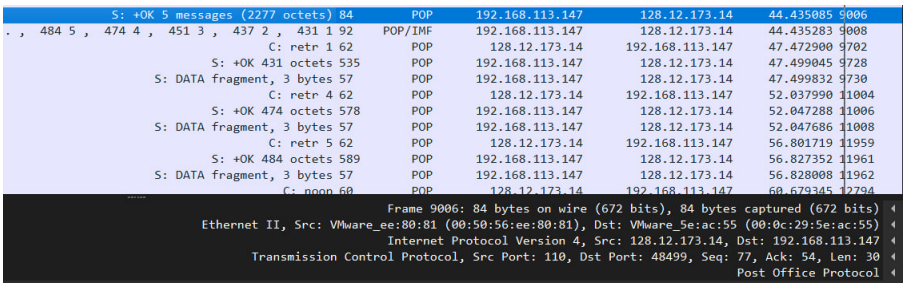
شکل ۲۲: خروجی Wireshark

پس اطلاعات مدنظر ما بدین شکل است:



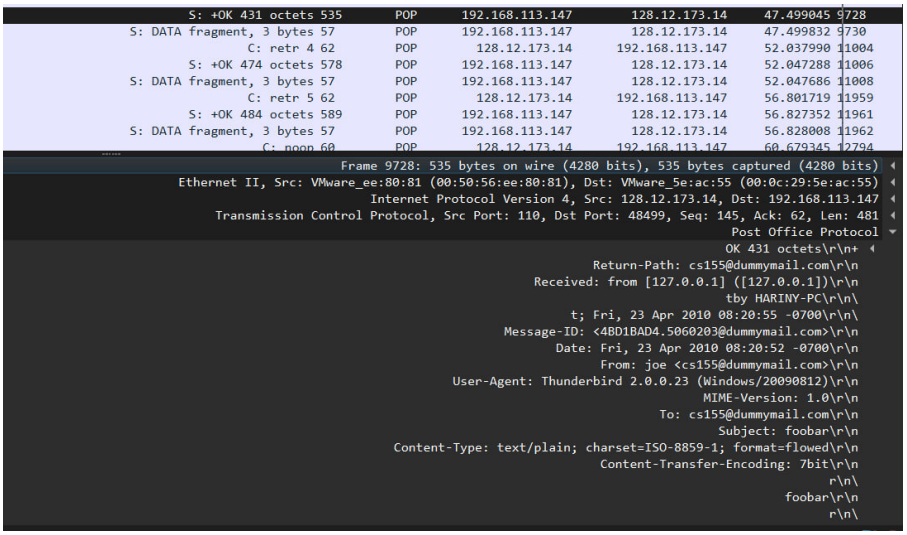
شکل ۲۳: نام کاربری cs155@dummymail.com و پسورد whitehat

در بخش بعدی می‌بینیم که ۵ پیام در صندوق پستی کاربر وجود دارد.



شکل ۲۴: ۵ پیام

محتویات یک ایمیل دل‌خواه نیز بدین شکل است:



شکل ۲۵: ایمیل دل‌خواه

تاریخ: Fri, 23 Apr 2010 08:20:52 -0700

از: joe <cs155@dummymail.com>

به: cs155@dummymail.com

موضوع: foobar

محتوا: foobar

همچنین با توجه به تصویر داریم که فیلد user-agent برابر است با

User-Agent: Thunderbird 2.0.0.23 (Windows/20090812)

پس شخص از ویندوز استفاده می‌کرده است و کلاینت ایمیل او نیز Thunderbird بوده است.