به نام خداوند بخشنده مهربان

دانشگاه صنعتی شریف
مدرس: مهدی خرازی

تمرین سری دوم*
امنیت شبکه

دانشکده‌ی مهندسی کامپیوتر
پاییز ۱۳۹۷

مقدمه

هدف این تمرین آشنایی با **آسیب‌پذیری‌های عمومی وب** و نحوه exploit کردن و بهره‌برداری از آن‌ها است. شما در این تمرین با تعدادی از حمله‌های پایه‌ای **SQL Injection** و **Code Injection** آشنا شده و نحوه جلوگیری از آن‌ها را بررسی خواهید کرد.

در انجام این تمرین با چالش‌های متنوعی برای **یافتن آسیب‌پذیری‌های موجود** و نحوه استفاده از آن‌ها روبه‌رو خواهیدشد. برای استفاده از آن‌ها به کمک **برنامه‌نویسی** ابزارهای ساده‌ای طراحی خواهید کرد و از دانش **رمزنگاری**[۱] خود نیز بهره خواهیدگرفت. مهارت‌هایی که از این تمرین کسب می‌کنید، برای شرکت در مسابقات واقعی CTF[۲] و نیز برنامه‌نویسی امن‌تر وب مفید خواهندبود.

تمرین به صورت دو وب‌سایت Sample است که روی یک ماشین مجازی اجرا می‌شوند. برای انجام تمرین، باید اطلاعات مشخصی را از این سرورها **استخراج** کرده، برای حل یکی از آنها **اصلاحاتی** پیشنهاد دهید و در نهایت **گزارشی کوتاه** در مورد عملکرد خود و نحوه دستیابی به اطلاعات یادشده بنویسید.

* با تشکر از ایمان اکبری، سید هومن شاهرخی، امیرپاشا قابوسی و سولماز سلیمی.

[۱] Cryptography
[۲] Capture the Flag

سوال اول ـ Cross-Site Scripting (XSS)

۱. پیش‌زمینه

حملات XSS یکی از رایج‌ترین حملات وب هستند و بخش عظیمی از آسیب‌پذیری‌های وب در دهه اخیر مربوط به این حملات بوده‌اند. در این قسمت از تمرین شما با یک حالت ساده و سرراست از این نوع آسیب‌پذیری‌ها روبرو هستید و بایستی بر اساس دانش خود از این حملات، دسترسی خود را به وب‌سایت قربانی افزایش داده و از این طریق پرچم[۳] را که رشته‌ای از حروف است، پیدا کنید.

اطلاعات این قسمت یک بازبینی ساده از مطالب تدریس شده در کلاس و اسلایدهای درس است، بنابراین اگر از قبل به این مطالب تسلط دارید می‌توانید از این بخش عبور کرده و مستقیماً قسمت ۲ را مطالعه نمایید.

۱.۱. تعریف XSS

بطور عمومی Cross-Site Scripting به دسته‌ای از حملات اطلاق می‌شود که در آن حمله‌کننده اسکریپت‌های بدافزار خود را به صفحات تولید شده توسط یک اپلیکیشن وب تزریق می‌کند تا بتواند آن‌ها را روی دستگاه کاربران آن اجرا کند.

وب‌سایت قربانی غالباً وب‌سایتی مورد اعتماد است که به تنهایی ضرری برای کاربران ندارد، اما حمله‌کننده از طریق آن اسکریپت‌های سمت کارخواه[۴] خود را روی سیستم کاربران اجرا می‌کند. هدف حمله‌کننده می‌تواند استخراج اطلاعات از سیستم کاربران، دور زدن قوانین دسترسی همانند Same-Origin Policy و مانند آن باشد.

برای مثال به سناریوی زیر به عنوان یک نمونه کلاسیک و بسیار ساده XSS توجه کنید:

۱. در یک انجمن اینترنتی[۵] کاربران می‌توانند به طور دلخواه متن های خود را آپلود کنند، و اپلیکیشن وب هرآنچه کاربر به عنوان پست منتشر کند را داخل صفحه HTML قرار می‌دهد.

۲. حمله‌کننده در نوشتهٔ خود یک تگ `<script>` قرار می‌دهد که کدهای جاوااسکریپت موردنظر او را اجرا می‌کند. این اسکریپت Session ID کاربر را به طور خودکار برای یک سرور که حمله‌کننده آن را در اختیار دارد، ارسال می‌کند.

۳. وب‌سایت یادشده نوشته‌های کاربران را بدون بررسی کافی منتشر کرده و کاراکترهای خاص آن‌ها را حذف نمی‌کند. در نتیجه تگ اسکریپت حمله‌کننده در صفحه HTML تمام بازدیدکنندگان قرار می‌گیرد.

۴. هر بازدیدکننده انجمن به محض باز کردن صفحه تزریق شده، به‌طور خودکار و حتی نامحسوس Session ID خود را برای وب‌سایت حمله‌کننده ارسال می‌کند.

---

[۳] flag
[۴] Client
[۵] Forum

اگر چه مثال بالا برای دادن دیدی کلی از مفهوم Cross-Site Scripting مناسب است، باید در نظر داشت که حملات XSS بر اساس نحوه ارسال به قربانی، اجرا و بازگشت به حمله‌کننده بسیار متنوع هستند و بازه بزرگی از حملات را شامل می‌شوند.

۲.۱. انواع حملات XSS

- **XSS Stored (Type-I)**: در حملات XSS نوع ۱ یا ذخیره‌شده، ورودی حمله‌کننده در سرور قربانی یعنی برای مثال در پایگاه داده، پیام‌های یک انجمن، log های بازدید، نظرات کاربران و ... ذخیره می‌شود و کاربر قربانی حمله، ورودی یادشده را به‌طور امن‌سازی نشده[۶] از وب‌سایت هدف حمله دریافت می‌کند.

  مثال زده شده در قسمت قبل نمونه‌ای از این دسته است.

- **XSS Reflected (Type-II)**: ورودی حمله‌کننده مستقیماً در وب‌سایت مورد حمله ذخیره نمی‌شود، بلکه در پیغام‌های خطا، نتایج جستجو و ... از وب‌سایت هدف حمله «بازتاب می‌شوند»، به این معنا که به شکلی (به غیر از ذخیره در خود وب‌سایت هدف حمله) برای کاربر قربانی ارسال شده و سپس او را به وب‌سایت هدف منتقل می‌کنند تا اسکریپت موردنظر را اجرا کند.

  این حملات از روش‌های مختلف همانند ایمیل، تبلیغات اینترنتی، سایت‌های دیگر و غیره به کاربران فرستاده می‌شوند. برای مثال فرض کنید در یک وب‌سایت هرگاه ورودی اشتباه باشد، آن را به همراه پیغام خطا در پایین صفحه چاپ می‌کند، اما قبل از نمایش آن ورودی را به طور درست امن‌سازی نمی‌کند.

  در این شرایط حمله‌کننده می‌تواند فرضاً به جای ورودی غلط، یک تگ `<script>` یا مانند آن را قرار دهد و سپس لینک صفحه آسیب‌پذیر با این ورودی را داخل یک تبلیغ در وب‌سایتی دیگر قرار دهد. کاربر قربانی به محض کلیک برروی تبلیغ، به وب‌سایت هدف منتقل شده و اسکریپت حمله‌کننده به عنوان ورودی غلط در پیغام خطای صفحه HTML قرار می‌گیرد و برای قربانی اجرا می‌شود.[۷] لفظ «بازتابی» به همین علت برای این دسته انتخاب شده‌است.

- **XSS DOM-Based (Type-۰)**: مربوط به زمانی است که کل مسیر حمله از منبع تا مقصد بدون گذشتن از داخل سرور و تماماً سمت کلاینت انجام شود، یعنی منبع اطلاعات و محل تخلیه آن همگی داخل مرورگر و ساختار DOM باشند. برای مثال منبع می‌تواند URL صفحه و مقصد می‌تواند یک دستور حساس همانند `document.write` باشد.

---

[۶] «امن‌سازی» اطلاعات به این معناست که قسمت‌هایی از ورودی که می‌توانند مشکل‌زا باشند به صورتی تبدیل شوند که دیگر خطری نداشته‌باشند (مثلا کاراکترهای ویژه و تگ های HTML به صورت escape شده تبدیل شوند) یا به‌طور کلی حذف شوند.

[۷] ممکن است بپرسید چرا اسکریپت یادشده را مستقیماً در همان تبلیغ قرار نمی‌دهند. توجه کنید که در این حالت، این اسکریپت داخل وب‌سایت هدف حمله اجرا می‌شود بنابراین مسائلی همانند Same-Origin Policy یا دسترسی به Cookie ها را می‌توان به این شکل دور زد.

۲. وظیفه شما

در ماشین مجازی داده شده، وب‌سایتی روی پورت `8083` اجرا می‌شود. شما بایستی براساس آنچه از Cross-Site Scripting آموخته‌اید، بتوانید سطح دسترسی خود به این وب‌سایت را ارتقا داده و پرچم را پیدا کنید. همانطور که گفته‌شد، پرچم یک رشته ساده است که در قسمت نکات در انتهای این سند قالب دقیق آن تعریف شده است.

به این منظور ابتدا تمامی کانال‌هایی را پیدا کنید که ممکن است بتوان از طریق آن‌ها اسکریپتی را تزریق کرد و سپس بررسی کنید که چنین کاری تا چه حد امکان‌پذیر است. دقت داشته باشید که هدف حمله شما سایر کاربران وب‌سایت هستند، بنابراین پس از یافتن کانال مناسب یک اسکریپت طراحی کنید که اطلاعات مفیدی را برای شما ارسال کند.

به همراه پرچم، تمامی مراحل حمله خود را از ابتدا تا یافتن پرچم ثبت کنید و در گزارش خود به طور کامل توضیح دهید. انتظار می‌رود اسکریپت تزریق شده خود را نیز حتماً در گزارش یادشده بیان کنید.

سوال دوم ـ SQL Injection

۱. پیش‌زمینه

این قسمت یادآوری اطلاعات موردنیاز برای انجام تمرین است. اگرچه مطالعه آن الزامی نیست و می‌توانید از زیرفصل دوم صورت تمرین را دنبال کنید، برای انجام ساده‌تر تمرین به‌شدت پیشنهاد می‌شود.[۸]

۱.۱. حمله‌های Injection

حمله‌های Injection یکی از فراگیرترین و پایه‌ای ترین انواع حمله‌های وب هستند و اساس بسیاری از حمله‌های پیچیده و مرکب را تشکیل می‌دهند. در فهرست مشهور ده مشکل امنیتی برتر OWASP حملات Injection در رتبه اول فهرست شده‌اند، به این معنا که یکی از بحرانی‌ترین خطرات اپلیکیشن‌های وب بوده و آسیب‌پذیری‌های مربوط به آن نباید در صنعت بیش از این تحمل شود.

Injection زمانی رخ می‌دهد که داده‌هایی غیرقابل اعتماد به صورت بخشی از یک دستور یا Query به نوعی مفسر[۹] ارسال می‌شوند. داده‌های ارسال شده توسط حمله‌کننده می‌توانند مفسر را فریب دهند تا دستوری ناخواسته را اجرا کند یا داده‌هایی را بدون اجازه صحیح به حمله‌کننده ارسال کند. اگر هدف حمله‌کننده پایگاه‌داده و مفسر SQL باشد، به این حمله SQL Injection می‌گویند.

---

[۸] بخش‌هایی از این قسمت از وب‌سایت شرکت Acunetix اقتباس شده‌است.

[۹] Interpreter

شکل ۱

OWASP Top 10 – 2013 (Previous)	OWASP Top 10 – 2017 (New)
A1 – Injection	A1 – Injection
A2 – Broken Authentication and Session Management	A2 – Broken Authentication and Session Management
A3 – Cross-Site Scripting (XSS)	A3 – Cross-Site Scripting (XSS)
A4 – Insecure Direct Object References - Merged with A7	A4 – Broken Access Control (Original category in 2003/2004)
A5 – Security Misconfiguration	A5 – Security Misconfiguration
A6 – Sensitive Data Exposure	A6 – Sensitive Data Exposure
A7 – Missing Function Level Access Control - Merged with A4	A7 – Insufficient Attack Protection (NEW)
A8 – Cross-Site Request Forgery (CSRF)	A8 – Cross-Site Request Forgery (CSRF)
A9 – Using Components with Known Vulnerabilities	A9 – Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards - Dropped	A10 – Underprotected APIs (NEW)

۲.۱. انواع SQL Injection

برای حمله SQL injection معمولاً نیاز به یک ورودی دارید که سرور از طریق آن، یک Query یا دستور SQL را می‌سازد و روی پایگاه‌داده اجرا می‌کند. در صورتی که ورودی را بتوان طوری دستکاری کرد که دستوری غیرمجاز یا ناخواسته اجرا شود، آسیب‌پذیری injection وجود دارد. حمله‌کننده پس از تشخیص ورودیِ ناامن و آسیب‌پذیری آن، ورودی را طوری تنظیم می‌کند که داده‌های موردنظر خود را از سرور استخراج کند یا اطلاعاتِ پایگاه‌داده را تغییر دهد.

براین اساس، حمله SQL Injection انواع زیادی دارد و می‌تواند بسیار ساده یا بسیار پیچیده باشد. حملات SQLi[۱۰] را می‌توان به سه دسته زیر تقسیم کرد:

۱. **In-band SQLi (Classic SQLi)**

حملات In-band SQL متداول‌ترین و به‌نوعی ساده‌ترین حملات SQLi محسوب می‌شوند. حمله In-band SQL مربوط به زمانی است که حمله‌کننده از کانال ارتباطی یکسانی برای **راه‌اندازی حمله** و **جمع‌آوری نتایج آن** استفاده می‌کند. دو نوع زیر، متداول‌ترین انواع حمله‌های In-band هستند:

(آ) **حمله براساس خطا**[۱۱] حمله SQL بر اساس خطا نوعی از حملات In-band است که در آن براساس پیغام‌های خطای throw شده توسط پایگاه داده ساختار و داده‌های آن استخراج می‌شود. در بعضی آسیب‌پذیری‌ها، یک حمله براساس خطا برای حمله‌کننده کافی است تا کل پایگاه داده را مرور کند.

[۱۰] کوتاه شدهٔ SQL Injection

[۱۱] Error-based SQLi

برای مثال، ممکن است در یک وب‌سایت پیغام‌های خطای دیباگ غیرفعال نشده باشند. به این‌ترتیب هنگام بروز خطا پیغام کامل آن در خروجی چاپ می‌شود. حمله‌کننده می‌تواند با ایجاد خطا به کمک این کانال ارتباطی داده‌های موردنظر خود را استخراج نماید.

(ب) **حمله براساس Union** [۱۲] به حملات SQLi گفته می‌شود که در آن‌ها به کمک کلیدواژه `UNION` زبان SQL نتایج دو یا چند دستور `SELECT` با یکدیگر ادغام می‌شود و نهایتاً نتیجه آن به صورت بخشی از پاسخ HTTP به حمله‌کننده برگردانده می‌شود.

۲. **Inferential SQLi (Blind SQLi)**

حمله‌های Inferential [۱۳] یا کور برخلاف حمله‌های In-band ممکن است برای Exploit کردن زمان بیشتری از حمله‌کننده بگیرند، اما به همان اندازه خطرناک هستند. در حمله‌های Inferential هیچ داده‌ای مستقیماً از طریق اپلیکیشن وب به حمله‌کننده منتقل نمی‌شود و حمله‌کننده نمی‌تواند نتیجه حمله خود را در همان کانال مشاهده کند (علت نامگذاری آن به «حملهٔ SQL Injection کور» همین است).

در عوض، در حمله «کور» حمله‌کننده ساختار پایگاه‌داده را با فرستادن ورودی‌های مهندسی شده، مشاهده پاسخ سرور و بررسی رفتار آن پس از دریافت ورودی بازسازی می‌کند.

حمله‌های کور عمدتاً در یکی از دو دسته **حملهٔ کور براساس خروجی دودویی** و **حملهٔ کور براساس زمان** جای می‌گیرند:

(آ) **حملهٔ کور براساس خروجی دودویی**

در این حمله‌ها یک درخواست SQL برای سرور ارسال می‌شود که اپلیکیشن وب را وادار می‌کند براساس `TRUE` یا `FALSE` بودن نتیجهٔ درخواست، رفتار متفاوتی از خود نشان دهد.

برای مثال، ممکن است در صورت `FALSE` بودن نتیجه درخواست، در پاسخ HTTP پیغام خاصی نمایش داده شود و این تنها کانال دریافت خروجی برای حمله‌کننده باشد. اگرچه این کانال نمی‌تواند اطلاعات را مستقیماً به حمله‌کننده بازگرداند، حمله‌کننده همچنان می‌تواند به صورت حساب‌شده اطلاعات موردنظر خود را از این کانال دودویی بازسازی نماید.

(ب) **حملهٔ کور براساس زمان**

اگر حتی یک کانال خروجی دودویی در پاسخ HTTP از سرور وجود نداشته باشد، همچنان آسیب‌پذیری‌های SQLi می‌توانند بسیار خطرناک باشند. حمله‌های کور براساس زمان، سرور را وادار می‌کنند در صورت `TRUE` یا `FALSE` بودن نتیجه یک عبارت SQL مقدار زمان مشخصی قبل از پاسخ دادن وقفه ایجاد کنند. به این ترتیب اگرچه در پاسخ HTTP حمله‌کننده فیلد خاصی وجود ندارد که به‌کمک آن نتیجه دستورات خود را مشاهده کند، همچنان می‌تواند براساس مدت اجرای دستور داده‌های دودویی خاصی را از سرور دریافت کرده و اطلاعات موردنظر خود را بازسازی کند.

---

[۱۲] Union-based SQLi

[۱۳] ترجمهٔ «استنباطی» برای آن پیشنهاد شده است که در این‌جا از آن صرف‌نظر کردیم.

۳. Out-of-band SQLi

این حملات به اندازه حملات قبلی فراگیر نیستند، زیرا براساس ویژگی‌های خاص پایگاه داده و اپلیکیشن وب عمل می‌کنند و تنها در شرایط خاصی قابل اعمال بوده و ممکن است بهره‌گیری از آن‌ها بسیار پیچیده باشد.

در این حمله‌ها، حمله‌کننده نمی‌تواند از کانال یکسانی داده‌های دستورات خود را اجرا کند و سیگنالی براساس آن‌ها دریافت کند. یعنی همانند حمله‌های کور امکان دریافت مستقیم داده‌های موردنظر وجود ندارد اما به دلایلی (برای مثال عدم‌ثبات در پاسخ‌های سرور) حمله‌های براساس زمان نیز امکان‌پذیر نباشد.

در تکنیک‌های Out-of-Band، حمله‌کننده بر توانایی سرور برای انجام درخواست‌های HTTP یا DNS و امثال آن‌ها تکیه می‌کند. برای مثال دستور `XP_DIRTREE` در Microsoft SQL Server می‌تواند درخواست های DNS را اجرا کند و به این‌ترتیب حمله کننده با کنترل DNS مقصد می‌تواند یک کانال ارتباطی بین پایگاه داده و خود ایجاد کند. به صورت مشابه دستور `UTL_HTTP` در پایگاه‌داده‌های Oracle برای ارسال دستورهای HTTP استفاده می‌شود. دقت کنید این تکنیک‌ها به شدت وابسته به پیکربندی و تنظیمات و شرایط سرورها هستند و به همین دلیل شانس کمتری برای موفقیت دارند.

۳.۱. پیشگیری از حمله

بسترهای امروزی توسعه وب مکانیزم‌های بسیاری برای جلوگیری از حمله‌های SQLi دارند. بسیاری از آن ها ابزارهای کارآمدی برای ساختن دستورات SQL و ارتباط با پایگاه‌داده ارائه می‌کنند که می‌تواند شانس حمله‌های SQLi را به‌شدت کاهش دهد. اما هم‌چنان سهل‌انگاری‌های برنامه‌نویسی می‌توانند منجر شوند آسیب‌پذیری‌های جدی در اپلیکیشن‌های وب وجود داشته باشد.

به‌طور کلی، راه‌حل صحیح مقابله با SQLi ساختن Query ها با ابزارهایی است که ورودی کاربر را برای حمله‌های گوناگون مورد بررسی قرارداده و اطمینان حاصل می‌کنند که Query ایجاد شده در هنگام اجرا هیچ عملکرد ناخواسته‌ای انجام نخواهدداد.

فیلترکردن کلیدواژه‌های SQL ، کاراکترهای خاص مانند Single-Quote و کاراکترهای White-Space از جمله فیلترهای ابتدایی برای ورودی‌های نامطمئن هستند. بنابراین حمله‌کننده در مرحلهٔ اول همواره بررسی می‌کنند سرور با چه کیفیتی ورودی‌ها را مورد بررسی قرارداده و آیا تمامی آسیب‌پذیری‌های ممکن را در فیلتر خود لحاظ کرده است یا خیر.

۲. پیش‌زمینه رمزنگاری

در جریان این تمرین، با یک چالش رمزنگاری ساده نیز مواجه خواهید شد. در ادامه، پیش‌زمینه کوتاهی ارائه شده که هرآنچه برای حل این تمرین در رابطه با رمزنگاری موردنیاز است، در آن بیان می‌شود. پیشنهاد می‌شود این قسمت را به طور کامل مطالعه کنید.

۱.۲. رمزنگاریِ متقارن

الگوریتم‌های رمزنگاریِ کلید متقارن[۱۴] یا به صورت ساده رمزنگاری متقارن دسته‌ای از الگوریتم‌های رمزنگاری هستند که از کلید یکسانی برای رمزنگاریِ متنِ ساده [۱۵] و رمزگشایی عبارت رمزشده [۱۶] استفاده می‌کنند. رمزنگاری متقارن به دلیل سادگی الگوریتم، عدم نیاز به توان محاسباتی زیاد برای تولید کلیدِ امن، و در نهایت سطح امنیت بالایی که ارائه می‌کنند، روش‌های رمزنگاری بسیار ارزش‌مندی هستند. با این وجود دو چالش جدی در این الگوریتم‌ها وجود دارد که مربوط به نحوه‌ی توزیع[۱۷] و مدیریت کلید[۱۸] می‌شود.

۲.۲. AES چیست؟

الگوریتم رمزنگاری متقارن AES[۱۹] در سال ۲۰۰۱ توسط موسسه‌ی استانداردهای ملی آمریکا معرفی شد و با توجه به ویژگی‌های قابل توجهی که داشت به سرعت استفاده از آن فراگیر شد. AES یک الگوریتم رمزنگاری بلوکی است که در صورت دریافت هر رشته‌ای، آن را به بلوک‌هایی با اندازه‌ی ۱۲۸ بیت تبدیل می‌کند. این بلوک‌های ۱۲۸ بیتی در دو مرحله‌ی رمزنگاری و رمزگشایی اندازه‌ی ثابتی دارند. بر خلاف اندازه‌ی ثابت بلوک‌ها، کلیدهای رمزنگاری می‌توانند ۱۲۸، ۱۹۲ و ۲۵۶ بیتی باشند. از آن‌جایی که اجرای الگوریتم AES به صورت تکرار مراحل است، تغییر اندازه‌ی کلید در واقع تعداد مراحل را به ترتیب به ۱۰، ۱۲ و ۱۴ تغییر می‌دهد.

هر مرحله از اجرای الگوریتم AES نیازمند اجرای چهار تابع مشابه است که در مرحله‌ی آخر یکی از توابع اجرا نمی‌شود. هم‌چنین در رمزگشایی یک پیام همه‌ی مراحل دقیقاً مانند رمزنگاری متن ساده به صورت معکوس اجرا می‌شود. مراحل مختلف رمزنگاری و رمزگشایی در شکل ۲ نمایش داده شده است.

۳.۲. انواع مدهای رمزنگاری AES

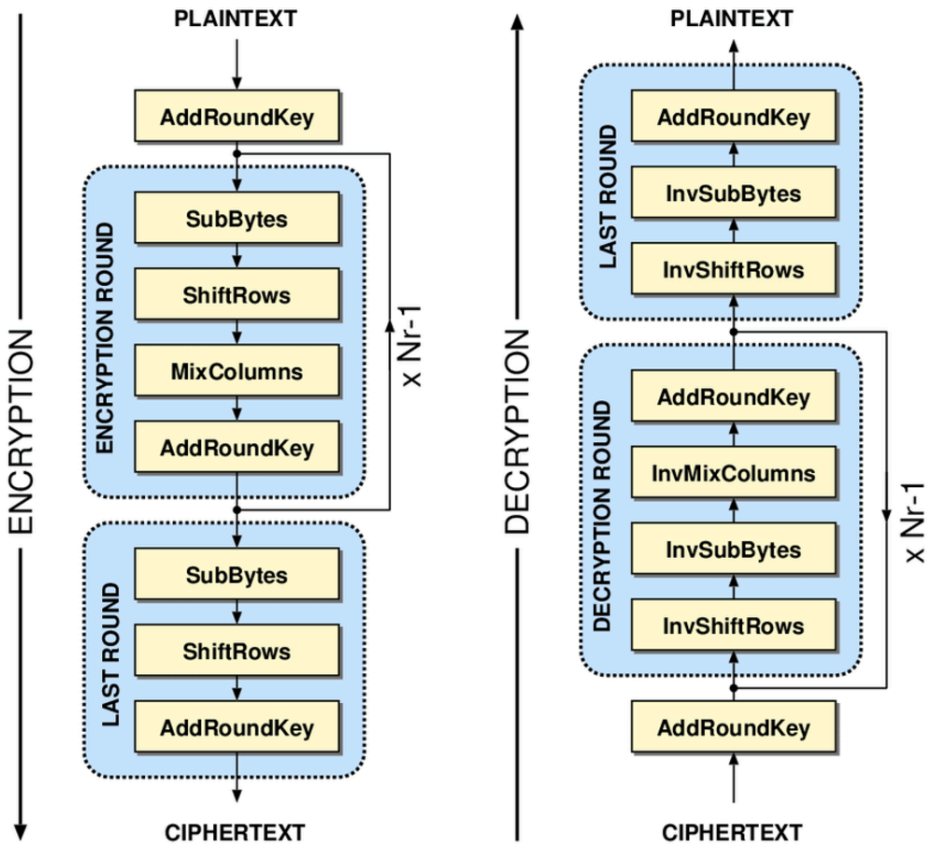
همه‌ی الگوریتم‌های رمزنگاری بلوکی، می‌توانند در مدهای مختلفی استفاده شوند. در ادامه دو مورد از پرکاربردترین مدهای رمزنگاری بلوکی با نام‌های ECB، CBC و CTR معرفی شده‌اند.

- ECB

در این مد از رمزنگاری، هر بلوک از داده (توجه کنید که اندازه‌ی بلوک در رمزنگاری AES همیشه مقدار ۱۲۸ بیت است) به صورت مجزا رمز می‌شود و سپس با کنار هم قرار دادنِ عبارتِ رمزشده‌ی هر بلوک، پیام رمزشده‌ی نهایی ساخته می‌شود. اگرچه این مد از رمزنگاری بسیار ساده است و نیاز به محاسبات کمی دارد،

[۱۴] Symmetric-key algorithm
[۱۵] Plain text
[۱۶] Cipher text
[۱۷] Key Distribution
[۱۸] Key Management
[۱۹] Advanced Encryption Standard

شکل ۲: مراحل مختلف رمزنگاری و رمزگشایی در الگوریتم AES (منبع تصویر)



اما از آن‌جایی که کلید به ازای همه‌ی بلوک‌ها یکسان است، انتظار می‌رود اگر دو بلوک حاوی پیام یکسانی باشند، پیام رمزشده نیز یکسان شود؛ به همین دلیل حملات زیادی روی این مد از رمزنگاری تعریف می‌شود.

- CBC

این مد از رمزنگاری، بسیار پرکاربرد است و تلاش اصلی آن این است که ضعف اصلی مد ECB را حل کند. به همین دلیل برای رمزنگاری هر بلوک از داده، قبل از این‌که عملیات اصلی رمزنگاری آغاز شود، داده با خروجی مرحله‌ی قبل (عبارت رمزشده از بلوک قبلی) XOR می‌شود. به این ترتیب اگر دو بلوک داده‌ی یکسانی داشته باشند، خروجی رمزشده‌ی آن‌ها برابر نیست. هم‌چنین برای این‌که بلوک اول رمزگذاری شود، یک مقدار ۱۲۸ بیتی تصادفی که معمولاً با عنوان IV شناخته می‌شود، تولید می‌شود و داده‌ی بلوک اول با این عبارت XOR می‌شود.

- CTR

این مد از رمزنگاری تلاش می‌کند روش‌های رمزنگاری بلوکی را تبدیل به رمزنگاری جریانی[۲۰] نماید. در واقع برای شروعِ رمزنگاری هر بلوک لازم نیست منتظر خروجی بلوک قبلی باشیم. در این مد از رمزنگاری، با

[۲۰] stream cipher

دریافت یک کلید و خروجی مرحله‌ی قبل، تابع رمزنگاری فراخوانی می‌شود، سپس خروجی این تابع با مقدار داده‌ی بلوک XOR می‌شود تا عبارت رمزشده‌ی بلوک را تولید کند و هم‌چنین خروجی آن به مرحله‌ی بعدی ارسال می‌شود.

مد CTR امکان محاسبات موازی را فراهم می‌کند و به همین دلیل، در رایانه‌های امروزی می‌تواند سرعتی چندین برابر مد CBC داشته باشد. هم‌چنین یک ویژگی بسیار جالب روش‌های جریانی این است که چون در مرحله‌ی آخر بلوک داده با مقداری XOR می‌شود، تغییر هر یک بیت در مقدار رمزشده منجر به تغییر همان بیت در عبارت متن ساده می‌شود.

۳. وظیفهٔ شما

همانند قسمت قبل، در این قسمت نیز شما با یک وب‌سایت ناآشنا روبرو هستید که کد آن را نیز در اختیار ندارید و قصد دارید به کمک حمله‌های SQLi به آن حمله‌کنید. به صورت مشابه در این قسمت تمرین نیز هدف شما به دست‌آوردن پرچمی است که در جایی از این سرور پنهان شده است.

پس از راه‌اندازی ماشین مجازی داده‌شده، روی پورت `8008` آن یک سرور وب بالا می‌آید که هدف حمله شماست. [۲۱] شما باید آن را به‌طور کامل بررسی کرده، آسیب‌پذیری‌های آن را تشخیص دهید و با حمله‌های SQLi حساب‌شده پرچم را که در جایی از سرور نهفته است، پیدا کنید.

به این منظور، شما به دانش **رمزنگاری** خود نیز نیاز دارید. در داخل وب‌سایت سرنخ‌هایی وجود دارد که شما را برای پیش‌برد حمله راهنمایی می‌کند.

پس از به‌دست آوردن پرچم از طریقی که در محل یافتن پرچم توضیح داده می‌شود کد سرور را دریافت کنید و آن را طوری تغییر دهید که آسیب‌پذیری‌های مورد استفاده شما دیگر در آن وجود نداشته باشد. نحوه ثبت و ارسال کد در همین پیوند پس از وارد کردن پرچم توضیح داده شده‌است.

۱.۳. تحویل‌دادنی‌ها

تحویل‌دادنی شما در این تمرین عبارتند از:

- **مستند**: شما باید تک‌تک مراحل انجام حمله، استراتژی های استفاده شده، نوع حملات انجام شده و نتایج آن‌ها را در مستند خود فهرست کنید.
- **کُدهای مورداستفاده در حمله**: برای انجام کامل این تمرین شما به برنامه‌نویسی نیاز دارید. پیشنهاد می‌شود به کمک مرورگر یا درخواست‌های HTTP دستی آسیب‌پذیری‌ها را بررسی کرده و برای استخراج داده‌ها ابزارهای خود را طراحی کنید. کدهای خود را نیز *به همراه مستند* خود به آدرس مشخص شده ارسال کنید.

[۲۱] Virtual Box به شما این امکان را می‌دهد که پورت های ماشین مجازی را به دستگاه میزبان forward کنید.
نحوهٔ forward کردن در این لینک توضیح داده شده است. پس از انجام این کار می‌توانید با مرورگر وب خود در دستگاه میزبان آدرس `localhost:8008/` را باز کرده و مانند یک وب‌سایت عادی آن را مرور کنید.

- **پرچم**: پس از انجام کامل حمله‌ها شما به پرچم دست خواهید یافت و باید آن را در مستند خود به عنوان مدرک انجام تمرین قرار دهید.
- **کُد اصلاح شده**: پس از وارد کردن پرچم در وب‌سایت داده‌شده، شما به کد سرور دست خواهید یافت. پس از آن می‌بایست با حداقل تغییر ممکن آن را در برابر حمله SQLi ای که انجام داده‌اید امن کرده و آسیب‌پذیری آن را از بین ببرید. در صفحه دریافت کد سرور، نحوه ارسال کد نیز توضیح داده شده است.

نکات

۱. پرچم در هر دو تمرین، دارای فرمت زیر است:

`FLAG{<90 BYTE STRING OF THESE CHARS: 0-9 _ $ # % a-z A-Z>}`

۲. تمام برنامه‌ی شما باید توسط خود شما نوشته شده باشد. فرستادن کل یا قسمتی از برنامه‌تان برای افراد دیگر، یا استفاده از کل یا قسمتی از برنامه‌ی فرد دیگری، حتی با ذکر منبع، تقلب محسوب می‌شود.

۳. ارسال پاسخ و راهنمایی در گروه‌های تلگرام و سایر منابع عمومی به منزله تقلب محسوب خواهدشد.

۴. از به اشتراک گذاشتن پرچم با سایر دانشجویان خودداری کنید. تشخیص اشتراک‌گذاری پرچم به نمره منفی صد برای طرفین منجر خواهدشد.

۵. در صورتی‌که هر مشکل یا پرسشی داشتید که فکر می‌کنید پاسخ آن برای همه مفید خواهدبود، آن را در فهرست پستی (میلینگ لیست) ارسال نمایید.

۶. از فرستادن جواب تمرین به فهرست پستی خودداری کنید.

۷. شما برای ارسال نهایی تمرین نیاز دارید تا مانند تمرین اول یک پوشه به نام hw2 در مخزن خود ایجاد نمایید. تمام فعالیت‌های خود را در گزارش به صورت کامل بنویسید. به ازای هر بخش شما باید یک عبارت پرچم ارسال کنید. هم‌چنین برای وصله کردن کد سرور در سوال SQLi یک پوشه به اسم `server-code` ایجاد نمایید. سپس کدی که از سمت سرور به دست آورده‌اید را در این پوشه کپی کنید و اصلاحات لازم را انجام دهید. در نهایت همه‌ی کدهای مورد استفاده‌ی خود را که در گزارش به آن‌ها اشاره کرده‌اید در پوشه‌ای به اسم `codes` قرار دهید.

۸. همه‌ی پرونده‌های لازم را با همان نامی که در مستند تمرین ذکر شده است، با دستورهای زیر ارسال کنید (فرض شده مخزن خود را در مسیر home قرار داده‌اید):

```
cd ~/ce442-971-student_id/hw2
git status
git add - -all
git commit -m "Finished my second assignment"
git push origin master
```