دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر

پایان‌نامه کارشناسی

گرایش شبکه و هوش مصنوعی

طراحی و پیاده‌سازی نرم‌افزار تحلیل برنامه‌های وب به منظور کشف آسیب‌پذیری‌های اسکریپت‌نویسی متقابل و تزریق

نگارش

طاهره فهیمی

استاد راهنما

دکتر حمیدرضا شهریاری

خرداد ۱۴۰۰

صفحه فرم ارزیابی و تصویب پایان نامه- فرم تأیید اعضاء کمیته دفاع

به نام خدا

تعهدنامه اصالت اثر

تاریخ:

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

اینجانب طاهره فهیمی متعهد می‌شوم که مطالب مندرج در این پایان‌نامه حاصل کار پژوهشی اینجانب تحت نظارت و راهنمایی اساتید دانشگاه صنعتی امیرکبیر بوده و به دستاوردهای دیگران که در این پژوهش از آنها استفاده شده است مطابق مقررات و روال متعارف ارجاع و در فهرست منابع و مآخذ ذکر گردیده است. این پایان‌نامه قبلاً برای احراز هیچ مدرک هم‌سطح یا بالاتر ارائه نگردیده است.

در صورت اثبات تخلف در هر زمان، مدرک تحصیلی صادر شده توسط دانشگاه از درجه اعتبار ساقط بوده و دانشگاه حق پیگیری قانونی خواهد داشت.

کلیه نتایج و حقوق حاصل از این پایان‌نامه متعلق به دانشگاه صنعتی امیرکبیر می‌باشد. هرگونه استفاده از نتایج علمی و عملی، واگذاری اطلاعات به دیگران یا چاپ و تکثیر، نسخه‌برداری، ترجمه و اقتباس از این پایان نامه بدون موافقت کتبی دانشگاه صنعتی امیرکبیر ممنوع است.
نقل مطالب با ذکر مآخذ بلامانع است.

امضا

تقدیر و تشکر

با سپاسی بی‌پایان از والدین عزیز و خواهر و برادر مهربانم که هیچ گاه از کوچکترین حمایتی نسبت به من در راه رسیدن به اهدافم دریغ ننموده‌اند.

با تشکر فراوان از استاد بزرگوار جناب دکتر حمیدرضا شهریاری که در تمامی موارد حمایت خود را نشان دادند و از محضرشان بهره‌های فراوان بردم.

با تقدیر فراوان از استاد گرامی، جناب دکتر بابک صادقیان که زحمت داوری این پایان نامه را تقبل فرمودند.

با سپاس فراوان از جناب آقای سعید طوسی سعیدی که قبول زحمت کرده و در تمامی مراحل این پروژه بنده را یاری کردند.

و با تشکری خالصانه از تمامی دوستان عزیزم که در طول انجام این پروژه صمیمانه همراهی‌ام کردند.

چکیده

با گسترش روزافزون فضای وب، نیاز به ابزاری برای شناخت آسیب‌پذیری‌های آن به صورت خودکار و سریع بیش از پیش شده است. از طرفی، هوش مصنوعی و الگوریتم‌های یادگیری ماشین و یادگیری عمیق به تمام زمینه‌های تکنولوژی وارد شده‌اند و باعث بهبود عملکرد سیستم‌های موجود شده اند. از این رو، در این گزارش سعی در پیاده‌سازی و بهبود ابزاری به منظور تشخیص آسیب‌پذیری CSRF با استفاده از روش‌های یادگیری ماشین شده است.

در این روش، هسته اصلی سیستم با استفاده از یادگیری ماشین پیاده‌سازی شده است که می‌تواند هر درخواست وب را به دو دسته حساس به CSRF و غیرحساس دسته‌بندی می‌کند. سپس با در نظر گرفتن یک سری از قوانین ابتکاری، سیستم می‌تواند با دقت بالایی آسیب‌پذیری را در وبگاه تشخیص دهد. این سیستم به صورت جعبه سیاه عمل می‌کند و همین موضوع باعث افزایش سرعت شده است. از طرفی دیگر، این سیستم به دلیل عدم وابستگی به زبان برنامه‌نویسی وبگاه، یک گزینه مناسب برای کشف این آسیب-پذیری می‌باشد.

واژه‌های کلیدی:

آسیب‌پذیری، یادگیری ماشین، یادگیری عمیق، جعبه سیاه

فهرست مطالب

صفحه

فهرست اشکال

صفحه

فهرست جداول

صفحه

فصل اول – مقدمه

مقدمه

فناوری به سرعت در حال پیشرفت است و اینترنت در حال تبدیل شدن به یک فناوری خانگی است. روزانه بیش از یک میلیون کاربر جدید به اینترنت متصل می‌شوند [۱]. این نرخ رشد، نشان دهنده استقبال بسیار فراوان کاربران از سرویس‌های مبتنی بر وب است. برنامه‌ها و سرویس‌های وب، کاربران را قادر می‌سازد تا فعالیت‌های برخط خود را با سهولت فراوان انجام دهند. در این دوران، بیشتر خدمات به گونه‌ای ارائه می‌شوند که دیگر نگران تغییرات جغرافیایی یا نبود زیرساخت کافی برای گرفتن خدمت نیستیم. با رشد بسیار سریع اینترنت، امنیت این محیط بسیار پراهمیت‌تر از پیش شده است.

۱-۱- اهمیت موضوع پروژه و بیان مسئله

امروز هر سازمانی اطلاعات عظیمی را از طریق وب به مشتریان، کارمندان و عموم مردم ارائه می‌دهد. در حالی که اطلاعات چاپی قابلیت تغییر ندارند اما محتوی تحت وب به صورت پویا قرار دارد و می‌تواند به سرعت به روز شود به همین دلیل، وبگاه‌ها و وب سرورها در معرض تهدیدهای امنیتی مختلفی قرار دارند.

هرسرور متصل به یک شبکه برخط نه تنها از تهدیدات داخلی ناشی از سوءاستفاده کارمندان از منابع و شبکه در خطر است بلکه در معرض طیف وسیعی از تهدیدات خارجی نیز قرار دارد. [1] امنیت وب برای محافظت از اطلاعات اعم از دسترسی غیرمجاز، تخریب یا تغییر بسیار مهم است.

حملات زیادی در دنیای برنامه‌های وب وجود دارد. هریک از این حملات می‌تواند هدف خاصی داشته باشد و یا بخش خاصی از سیستم را مورد تهدید قرار دهد. تزریق[۱] یکی از شایع‌ترین حملات موجود در دنیای وب است.

بسیاری از زمان‌ها، تشخیص یک حمله می‌تواند مدت زمان زیادی طول بکشد از این رو پیدا کردن و رفع آسیب‌پذیری‌ها می‌تواند نقطه اصلی افزایش امنیت باشد. از طرفی ابزارهای موجود نیاز به تغییرات زیادی دارند تا بتوانند به صورت گسترده مورد استفاده قرار بگیرند همین امر باعث هزینه‌بر بودن تشخیص آسیب‌پذیری‌ها نیز می‌باشد.

---

[۱] Injection

۱-۲- مروری بر پیشینه موضوع

تاکنون سیستم‌های مختلفی برای تشخیص آسیب‌پذیری‌های وب ساخته شده است. این سیستم‌ها از منظر ورودی‌هایشان به صورت کلی به سه دسته زیر تقسیم می‌شوند.

۱- سیستم‌های تحلیل ایستا: سیستم‌هایی که تنها کد برنامه را برای تشخیص آسیب‌پذیری بررسی می‌کنند.

۲- سیستم‌های تحلیل پویا: سیستم‌هایی که هنگام اجرای برنامه عملکرد آن را به منظور تشخیص آسیب‌پذیری به کار می‌برند.

۳- سیستم‌های ترکیبی: سیستم‌هایی که از ترکیب هر دو روش بالا ساخته شده اند.

به دلیل پیچیدگی تشخیص آسیب‌پذیری های موجود در دنیای وب، عموما سیستم‌های تحلیل ایستا برای این مهم مطلوب نیستند، به همین دلیل در این گزارش از سیستمی مبتنی بر تحلیل پویا استفاده می کنیم.

۱-۳- هدف پروژه

در این پروژه سعی شده است که سیستمی خودکار به گونه‌ای پیاده‌سازی شود که بتواند آسیب‌پذیری‌های مربوط به جعل کردن درخواست‌های وب را به درستی تشخیص دهد. نکته اصلی این کار، استفاده از یادگیری ماشین برای افزایش توانایی تشخیص سیستم و بهره‌وری است. استفاده از یادگیری ماشین باعث می‌شود تا سیستم بتواند بدون هزینه بالایی وجود یاعدم وجود آسیب‌پذیری را تشخیص بدهد.

در این پروژه مدل‌های مختلفی از الگوریتم‌های یادگیری ماشین آزمون و پیاده‌سازی شده است و با توجه به محدودیت‌های موجود بهترین مدل برای این‌کار انتخاب شده است. سپس نتایج مربوط به تست کردن این ابزار بر روی سایت‌های مختلف نیز در ادامه ذکر شده است.

۱-۴- معرفی ساختار پایان‌نامه

در این فصل مقدمه‌ای از موضوع پروژه، کاربرد آن و هدف پروژه توضیح داده شد. در ادامه‌ی این رساله، دانش کسب شده در این پروژه و نحوه‌ی ساخت سیستم هدف در ۵ فصل توضیح داده می‌شود.

در فصل دوم به توضیح آسیب‌پذیری CSRF می‌پردازیم و انواع آن را بیان می‌کنیم. یکی از نکات مهم برای تحلیل یک آسیب پذیری، شناخت دقیق یک آسیب‌پذیری و سناریوهای مختلف برای بهره برداری از آن است که در این فصل به صورت مفصل به آن‌ها خواهیم پرداخت.

در فصل سوم به مفاهیم اصلی یادگیری ماشین می‌پردازیم و مدل‌های مختلف موجود در یادگیری ماشین را با یکدیگر مقایسه می‌کنیم تا بتوانیم به یک مدل مناسب برای ابزارمان دست پیدا کنیم در ادامه اهمیت استفاده از یادگیری ماشین برای رسیدن به اهداف مورد نظرمان را بیان می‌کنیم.

در فصل پایانی به نحوه پیاده‌سازی و چالش‌های موجود می‌پردازیم. نحوه ساخت ابزارو رویکرد خودمان برای حل موانع را نیز بررسی می‌کنیم.

فصل دوم – آسیب پذیری CSRF

قدم اول شناخت جنبه های مختلف آسیب پذیری و آگاهی از نحوه بهر برداری از آن می باشد. در این فصل ابتدا مفهوم اصلی CSRF بیان می شود سپس انواع مختلف آن را بررسی می کنیم. هم چنین شیوه های مختلف جلوگیری از CSRF را هم بررسی می کنیم. در انتهای فصل نیز هدف اصلی سیستم ساخته شده مان را بیان می کنیم.

۲-۱- آسیب‌پذیری CSRF

برای فهم بهتر این آسیب‌پذیری لازم است که در ابتدا یک سری پیش زمینه‌هایی را بدانیم.

هرگاه به یک دامنه[۱] خاص درخواست می‌فرستیم کوکی[۲] مربوط به آن دامنه به صورت خودکار توسط مرورگر ارسال می‌شود. این اقدام بدون اینکه مرورگر در نظر داشته باشد که دامنه درخواستی چه چیزی می‌باشد انجام می‌شود. برای مثال فرض کنیم در وبگاه hello. com آدرس وبگاه attacker. com قرار داشته باشد و فردی روی این آدرس کلیک کند. در این حالت فرد بدون اینکه بخواهد کوکی‌های مربوط به وبگاه hello. com توسط مرورگرش به درخواست مربوط به attacker. com پیوست شده و ارسال می‌شود.

مورد دیگری که باید به عنوان پیش زمینه بدانیم کنترل کننده‌ای به اسم قانون دسترسی بین‌دامنه[۳] است. این قانون باعث می‌شود که دو وبگاه تنها در حالتی که دامنه و پورت[۴] و طرح‌واره[۵] یکسانی داشته باشند بتوانند با یکدیگر تعامل برقرار کنند. [۱] برای مثال در عکس شماره۱، به دلیل وجود قانون مبدا یکسان دو سایت attacker. com و google. com نمی‌توانند با یکدیگر ارتباط بگیرند.

---

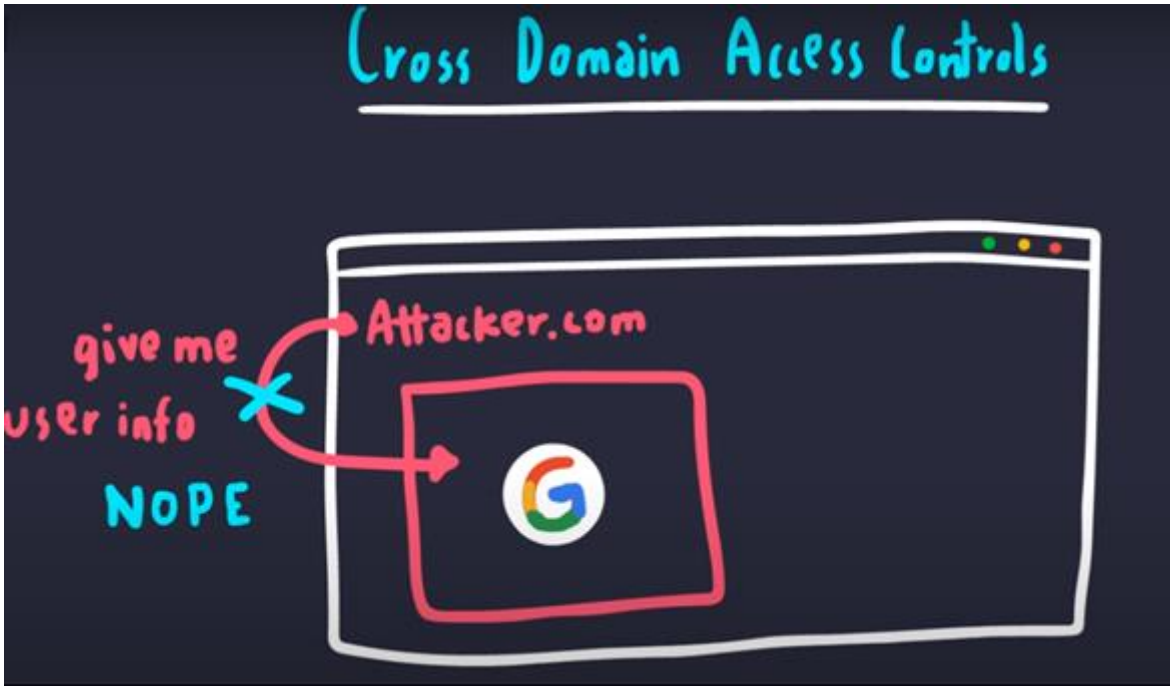
۱ Domain

۲ Cookie

۳ Cross Domain Access Control

۴ Port

۵ Schema



شکل ۱- قانون مبدا یکسان [2]

حمله CSRF زمانی رخ می‌دهد که یک وبگاه درخواستی را از طریق مرورگر به یک وبگاه دیگر بفرستند که این درخواست باعث تغییر در مقدار خاصی در آن وبگاه شود. به عبارت دیگر CSRF یک نوعی از حمله است و هنگامی اتفاق می‌افتد که یک وبگاه مخرب، ایمیل یا وبلاگ یا یک پیامی باعث تغیراتی ناخواسته در یک وبگاه مورد اعتماد شود که کاربر در آن وبگاه احراز هویت شده است.

علت اینکه یک حمله CSRF اتفاق می‌افتد در پیش نیاز ابتدایی ذکر شده است. به عبارت دیگر چون یک درخواست‌های یک مرورگر به صورت خودکار همه کوکی‌های مربوط به یک وبگاه را همراه با آن درخواست می‌فرستد بنابراین اگر کاربر در یک وبگاه احرازهویت کرده باشد وبگاه درخواست واقعی و درخواست‌های مخرب را یکسان در نظر می‌گیرد. [۲]

شدت حمله‌ای که وابسته به این آسیب‌پذیری است وابسته به مقدار آسیب‌پذیری کشف شده در آن برنامه آسیب‌پذیر و مقدار دسترسی کاربر است. برای مثال اگر یک فردی بتواند در یک آسیب‌پذیری در ادمین یک سیستم پیدا کند می‌تواند با استفاده از این آسیب‌پذیری دسترسی‌های بسیار زیادی را پیدا کند. [۲]

۲-۲- آسیب پذیری های تزریق و XSS

حال که با مفهوم اصلی این آسیب‌پذیری آشنا شده‌ایم بهتر است که رابطه آن را با آسیب‌پذیری های تزریق و XSS بدانیم. برای این منظور ابتدا به بررسی دقیق این آسیب پذیری‌های می‌پردازیم.

۲-۲-۱- تزریق

حملات تزریق به صورت کلی به هر حمله‌ای گفته می‌شود که در آن داده‌ی مخرب به یک مفسر[۱] فرستاده شده و روی آن مفسر اجرا شود. این حمله می‌تواند از طریق پرس‌وجو[۲] با درخواست[۳] صورت گیرد. در نهایت این حمله می‌تواند به دسترسی بدون احراز هویت یا پردازش یک درخواست خارج از دسترسی منجر شود. این حمله در رتبه‌بندی اوسپ[۴] اولین رتبه را دارد. [۳] همین موضوع نشان دهنده اهمیت بالای شناسایی و تشخیص آسیب‌پذیری مربوط به این حمله است.

این نوع از حملات دسته‌بندی‌های فراوانی دارد اما آن چه که ما به عنوان حمله تزریق می‌شناسیم عموما همان حمله sql است. شکل شماره ۲ یک سناریو ساده از این حمله را نشان می‌دهد. در این شکل، حمله‌کننده یک دستور سمی که باعث تغییر یا افشای داده‌های موجود در پایگاه داده می‌شود را به سیستم وارد می‌کند. چون بررسی‌های امنیتی لازم در سیستم وجود ندارد، این دستور به پایگاه داده وارد شده و در آن‌جا پردازش می‌شود. این امر می‌تواند باعث خسارت‌های جبران ناپذیری شود.

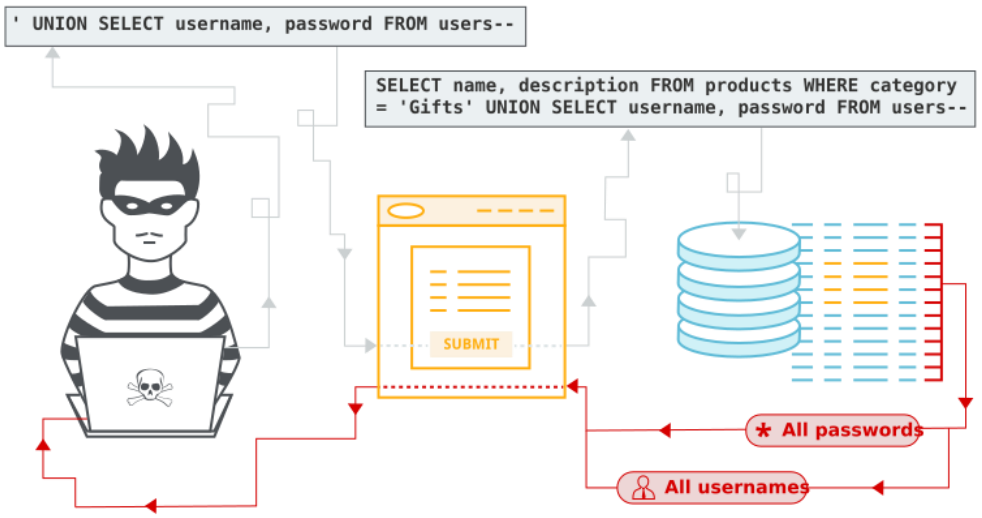
---

۱ Interpreter

۲ query

۳ Command

۴ Owasp Top Ten



شکل ۲ - سناریو حمله ی تزریق در پایگاه داده[3]

XSS - ۲-۲-۲

حمله XSS نیز زمانی اتفاق می‌افتد که در یک برنامه[۱] بدون احراز هویت بتوانیم یک اسکریپت[۲] خاصی را اجرا کنیم. این حمله می‌تواند منجر به ربودن نشست[۳]، یا تزریق داده‌های خراب یا هدایت‌مجدد[۴] صفحه به صفحه‌های شامل بدافزار شود. [۵]

در شکل شماره ۳ می‌توانیم یک سناریو از این حمله را ببینیم. در این حمله، حمله کننده در قسمتی از درخواستش، کد جاوااسکریپ[۵] مخربی را قرار داده است. به محض اینکه کاربر عادی روی آن کلیک کند،

---

۱ Application

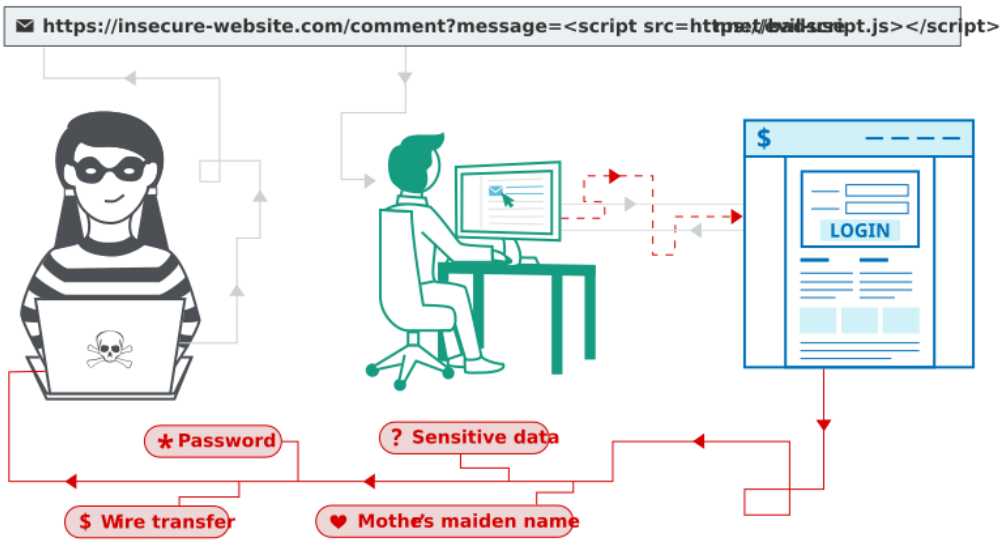
۲ Script

۳ Session

۴ Redirect

۵ Java Script

این کد اجرا می‌شود. از آن‌جایی که زبان جاوااسکریپت توانایی‌های زیادی دارد، این کد نیز می‌تواند بسیار مخرب باشد.



شکل ۳ – سناریوی حمله XSS [4]

همانطور که از تعاریف بالا می‌توان نتیجه گرفت حملات CSRF و XSS بسیار نزدیک به هم هستند به گونه‌ای که می‌توان حمله CSRF را به عنوان زیر بخشی از حملات XSS دانست. از طرفی دیگر حملات Injection به صورت کلی همان تزریقی است که منجر به نتایج مخرب می‌شود پس XSS و CSRF هر دو گونه‌ای از حملات تزریق هستند.[5]

به عبارت دیگر در یک حمله XSS حمله کننده دستورات جاوااسکریپت را در مرورگر قربانی اجرا می‌کند و در حمله CSRF یک سری فعالیت‌های ناخواسته‌ای را بدون اجازه کاربر اجرا می‌کند. در ادامه روش‌های مقابله با حملات CSRF را بررسی می‌کنیم و با در نظر گرفتن این روش‌ها بهتر می‌توانیم متوجه دسته‌بندی این حملات شویم.

۲-۳- روش های مقابله با حمله CSRF

به طور کلی این آسیب‌پذیری می‌تواند در تمامی برنامه‌های کامپیوتری وجود داشته باشد. ما در ابتدا پاسخ‌های گوناگون برای همه برنامه‌ها را ارائه می‌دهیم و در نهایت بر روی پاسخ‌های مربوط به برنامه‌های کاربردی (وبگاه ها) تمرکز می‌کنیم.

۱- اولین روش این است که در صورتی که ازچارچوب[۱] خاصی برای ساخت وبگاه‌مان استفاده می‌کنیم حتما بررسی کنیم که آن چارچوب در مقابل CSRF پیاده‌سازی‌های مقاوم‌سازی را انجام داده باشد. در صورتی که فریم‌ورک ما مقاوم‌سازی دربرابر CSRF را پیاده‌سازی نکرده بود، باید به ازای تمامی درخواستی‌هایی که حالتی را تغییر می‌دهند، باید csrf مهره[۲] به آن اضافه کنیم این مقدار را در پس‌زمینه[۳] بررسی کنیم.[6]

۲- برای نرم افزارهایی که حالت کاربر را نگهداری می‌کنند می‌توانیم از مهره الگوی سنکرون‌ساز[۴] استفاده کنیم.

۳- برای نرم افزارهایی که حالت کاربر را نگهداری نمی کنند می توانیم از کوکی چک‌کننده‌مجدد[۵] استفاده کنیم.

۴- برای صفحات وبگاه باید حداقل از یکی از روش های زیر برای امن سازی استفاده کرد:

الف) استفاده از ویژگی samesite در کوکی مربوط به جلسه

---

۱ Framework

۲ Token

۳ Background

۴ Synchronization pattern

۵ Doubled submit

ب) برای عملیات‌هایی که حساس هستند باید از حفاظت‌تقابل‌کاربر[۱] استفاده کرد

۵- از هیچ درخواست‌ای از نوع گرفتن[۲] برای تغییردادن حالت در سیستم‌مان استفاده نکنیم.

۲-۴- دسته بندی آسیب پذیری CSRF

این آسیب‌پذیری می‌تواند به دو دسته وجود داشته باشد. دسته اول آسیب‌پذیری سمت سرور است که نمونه آن در بالا معرفی شده است. دسته دیگر مربوط به آسیب‌پذیری در سمت کاربر است. این نوع از این آسیب‌پذیری اخیرا شناخته شده است. سپس در ادامه به یکی از اقداماتی که برای بهبود این آسیب‌پذیری انجام شده است اشاره می‌کنیم.

۲-۴-۱- بررسی آسیب پذیری در سمت کاربر

این آسیب‌پذیری برای اولین بار توسط سهیل خدایاری[۳] و جیانکارلو پلیگرو[۴] در مقاله‌ی [۷] بررسی شده است. در این آسیب پذیری، جزئی که آسیب‌پذیری می‌باشد مربوط به برنامه جاوااسکریپت است که در آن حمله کننده می‌تواند با تغییر در پارامترهای ورودی برنامه جاوااسکریپت، درخواست‌های متفاوتی تولید کند. برخلاف آسیب‌پذیری CSRF در سمت سرور، این دسته از آسیب‌پذیری نمی‌تواند با استفاده از مهره‌های ضد-csrft[۵] مقاوم‌سازی شود.

---

۱ User interaction based protection

۲ GET

۳ Soheil Khodayari

۴ Giancarlo Prllrgrio

۵ Anti-CSRF

شناخت یک آسیب‌پذیری جدید نیاز به بررسی هزاران صفحه جدید دارد که اینکار با استفاده از ابزارهای موجود ممکن نیست به همین دلیل این گروه ابزاری جدید را توسعه دادند. این ابزار از ساختار هیبریدی پیروی می‌کند. علت انتخاب این ساختار را می‌تواند در سه دلیل زیر خلاصه کرد:

۱- زبان جاوا اسکریپت به دلیل ماهیتی که دارد نمی تواند هیچ نقطه کانونی داشته باشد.
۲- قسمت های مختلف برنامه‌های جاوااسکریپت براساس رویدادهای مختلف اجرا می-شوند.
۳- استفاده از روش های استاتیک به تنهایی نمی تواند ساختار یک برنامه جاوااسکریپت را به درستی متوجه شود.

بدیهی است که تمام مواردی که در بالا ذکر شده است به این دلیل است که این آسیب‌پذیری از بخش برنامه جاوااسکریپت وبگاه ما ناشی می‌شود. درصورتی که این حمله‌کننده، یک حمله کننده مربوط به شبکه باشد می‌تواند پایه‌بار[۱] را در مرورگر مربوط به قربانی قرار دهد تا بتواند در شرایط دیگری بهره‌برداری کند. این آسیب‌پذیری می‌تواند در قسمت‌های مختلفی قرار داشته باشد مثل مخزن وبگاه[۲]، آدرس اینترنتی[۳]، پیام‌های ارسال‌شده، مدارک ارجاع‌شده، نام پنجره، ویژگی‌های HTML و یا کوکی ها. بهره-برداری از هر یک از این اسیب‌پذیری‌ها وابسته به توانایی حمله کننده و سایر شرایط موجود (مثل صفحه آسیب‌پذیری که از آن به عنوان طعمه استفاده می‌شود) وابسته است.

۲-۴-۲- سناریو بهره برداری از آسیب پذیری CSRF در سمت کاربر

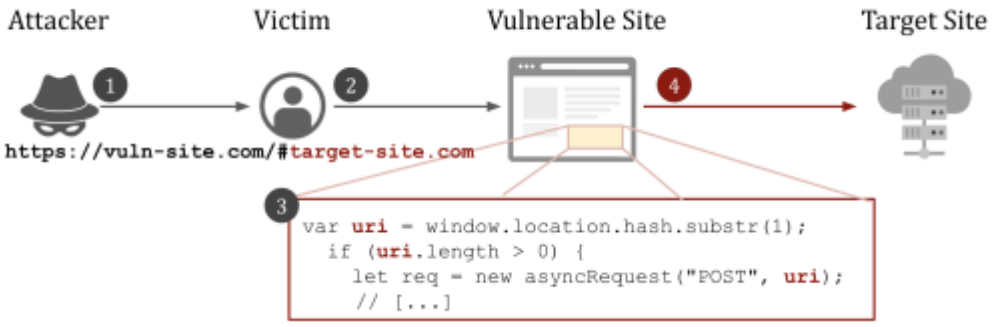
برای فهم بهتر این آسیب‌پذیری بهتر است که یک سناریو از شیوه بهره‌برداری از آن را ببینیم. در این سناریو، حمله‌کننده یک کاربر را گول می‌زند تا وی روی لینکی مخربی که مربوط به وبگاه حمله‌کننده یا وبگاهی که اسیب پذیر است کلیک کند. برای مثال شکل ۴ را در نظر بگیرید.

---

۱ Payload

۲ Web Storage

۳ URL



شکل ۴ - بهره برداری از آسیب پذیری csrf در سمت کاربر [7]

در مرحله اول حمله کننده آدرس اینترنتی وبگاهی آسیب‌پذیر را که در آن آدرس اینترنتی مربوط به وبگاه هدف قرار دارد را به گونه‌ای قرار می‌دهد تا کاربر روی آن کلیک کند (مثلا میتواند این آدرس اینترنتی را در صفحات وب پست کند تا کاربر روی آن کلیک کند یا آن را برای یک کاربر ایمیل کند) در مرحله دوم چون این آدرس اینترنتی مربوط به برنامه‌ای است که کاربر به آن اعتماد دارد (یعنی این وبگاه خودش باعث تخریب نیست و تنها مشکلی که دارد آسیب‌پذیر بودنش است) پس این وبگاه باز می‌شود و در مرحله سوم هنگام بارگزاری کامل صفحه، کد جاوااسکریپت موجود در این وبگاه، آدرس اینترنتی هدف را از آدرس اینترنتی صفحه استخراج می‌کند و یک درخواست HTTP را به وبگاه هدف ارسال می‌کند که همین کار باعث بوجود امدن یک آسیب‌پذیری امنیتی شده و حالت آن وبگاه هدف را تغییر می‌دهد.

۲-۴-۳- روش جلوگیری از آسیب پذیری CSRF

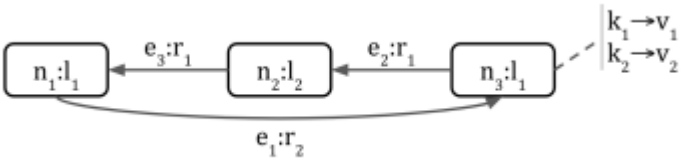
یکی از ابزارهایی که برای شناخت این آسیب پذیری استفاده می شود، چارچوب جاو[۱] است. این چارچوب برای شناسایی این آسیب پذیری از ساختار هیبریدی استفاده می کند.

روش های موجود عموما بر اساس شیوه های ایستا یا پویا هستند. روش های ایستا به دلیل اینکه تنها به صورت کد نگاه می کنند، توانایی پردازش خروجی سیستم به ازای ورودی های مختلف را ندارند برای

[۱] JAW

همین روش کارامدی محسوب نمی شوند. از طرفی دیگر روش های پویا هم به دلیل پیچیدگی سیستم به ازای رویداد های مختلف نمی تواند مناسب باشد برای همین بازدهی مناسبی ندارد.

برای رفع این مشکل می توانیم از روش ترکیبی این دودیدگاه استفاده کرد برای همین از روش گراف ویژگی هیبریدی استفاده می شود. در این گراف هر حالت نشان دهنده یک شرایط مختلف از سیستم است و هر یال نیز یک ساختار از کد است که باعث می شود سیستم از یک حالت به حالت دیگر تغییر کند. نود ها و یال ها می توانند داده هایی را تحت عنوان کلید و مقدار ذخیره کنند.



شکل ۵- نمونه گراف سیستم [7]

۲-۵- خلاصه

در این فصل به صورت دقیق با آسیب‌پذیری CSRF آشنا شدیم و فهمیدیم که حمله CSRF زمانی اتفاق می‌افتد که حمله کننده بتواند کاربر را فریب دهد به گونه‌ای که یک درخواست مخرب را بدون فهمیدن ارسال کند. در ادامه یک سناریو از چگونگی بهره برداری از این آسیب‌پذیری را دیدیم. سپس این آسیب‌پذیری را با اسیب‌پذیری‌های XSS و تزریق مقایسه کردیم و متوجه شدیم که آسیب‌پذیری CSRF می‌تواند جزو دسته‌ای از آسیب‌پذیری‌های تزریق باشد.

آسیب‌پذیری CSRF به دو دسته کلی آسیب‌پذیری سمت کاربر و سرور تقسیم می‌شود. سپس آسیب‌پذیری سمت کاربر را بررسی کردیم. این آسیب‌پذیری یکی از جدیدترین اسیب‌پذیری‌های موجود است و تاکنون مورد توجه زیادی قرار نگرفته است. یک سناریو از شیوه بهره برداری از این آسیب‌پذیری را دیدیم سپس یک نمونه از ابزارهای کارامدی که برای شناخت این آسیب‌پذیری وجود دارد را شناختیم و اصولی که برای ساخت این ابزار مورد استفاده قرار گرفته است را بررسی کردیم.

در نهایت ابزارهای موجود برای تشخیص این آسیب‌پذیری را بررسی کردیم و نقاط قوت و ضعف آن‌ها را شناسایی کردیم. در مجموع به این نتیجه رسیدیم که نیاز به یک ابزار کارامد که با حداقل وابستگی‌ها می‌تواند عمل کند نیاز اصلی ما خواهد بود.

فصل سوم – کشف آسیب پذیری با یادگیری ماشین

۳-۱- یادگیری ماشین چیست؟

یادگیری ماشین به روش‌هایی گفته می‌شود که در آن داده‌های زیادی که از قبل جمع‌آوری داده شده‌اند به یک مدل یادگیری ماشین داده می‌شود تا این مدل بتواند از روی این داده‌ها آموزش ببیند و برای داده‌هایی که تا بحال آن‌ها را ندیده است پاسخ‌هایی درست بدهد. به داده‌هایی که برای آموزش مدل استفاده می‌شود داده‌های اموزشی[۱] و برای داده‌هایی که برای ارزیابی کردن مدل استفاده می‌شود داده‌های تست[۲]گفته می‌شود. در ادامه انواع این داده‌ها و مدل‌ها را بررسی می‌کنیم.

۳-۲- انواع داده های ورودی در یک مدل

سه دسته اصلی داده برای آموزش یک شبکه خواهیم داشت:

۱- داده های بدون برچسب
۲- داده های بابرچسب
۳- داده های نیمه‌برچسب دار

اگر مسئله داده شده را یک مساله کلاس‌بندی[۱] در نظر بگیریم، داده‌های بدون برچسب داده‌هایی هستند که مقدار تعلق به کلاس مشخصی ندارند. این داده‌ها با استفاده از ویژگی‌هایشان برای دسته‌بندی استفاده می‌شوند یکی از ساده‌ترین دسته‌بند[۲]ها برای دسته‌بندی کردن این داده‌ها الگوریتم kمیانگین [۴] است. این الگوریتم برحسب فاصله‌ای که داده‌ها از یکدیگر دارند داده‌ها را در دسته‌های مختلفی قرار می‌دهد. داده‌های برچسب دار داده‌هایی هستند که به ازای هر داده، کلاسی که به آن تعلق دارد تحت عنوان یک ویژگی برای آن داده مشخص شده است. درباره این نوع از داده‌ها در ادامه بیشتر بحث خواهیم کرد. داده‌های نیمه برچسب دار نیز داده‌هایی هستند که بخشی از داده‌های موجود در مجموعه داده دارای

---

۱ Classification

۲ Classifier

برچسب بوده و بخش دیگری از داده‌ها بدون برچسب هستند. کار با این دسته از مجموعه داده‌ها سخت‌تر از بقیه مجموعه داده‌ها می‌باشد.

۳-۴- جمع آوری داده:

برای جمع آوری داده از مجموعه داده موجود در مقاله [۸]استفاده می‌کنیم. در این روش مجموعه داده بدست امده از تحلیل ۶۰ وبگاه محبوب ساخته شده است که دارای ۵۸۲۸ عنوان ضبط‌شده دارد که ۹۳۹ عدد از این درخواست‌ها آسیب‌پذیر هستند. از این داده‌ها تنها برای آموزش مدل استفاده میکنیم هر یک از داده‌ها نمایانگر یک لینک از صفحات موجود در وب است که با استفاده از ابزارهای موجود مانند Berp وجود آسیب‌پذیری در آن‌ها چک شده است. هر عضو از مجموعه داده دارای ۴۹ ویژگی است که اولین ویژگی مربوط به آسیب‌پذیر بودن یا نبودن آن دیتا می‌باشد، به عبارت دیگر داده‌های موجود در مجموعه داده‌مان همگی دارای برچسب می‌باشند.

مابقی ویژگی‌های موجود برای هر داده از روی ساختار مربوط به هر درخواست[۱] HTTP ساخته شده است. برای مثال، یکی از ویژگی های داده numOfParams است که تعداد پارامترهای موجود در یک درخواست را نشان می‌دهد.

۳-۴-۱- پیش پردازش داده ها

پیش‌پردازش[۲] روی داده‌ها فقط برای استخراج ویژگی‌های مختلف از درخواست و پاسخ‌های HTTP انجام شده و پیش‌پردازش خاص دیگری نیاز نداریم. تنها برای بهتر شدن کیفیت مدل و آموزش، از روش‌های کاهش ابعاد برای پیداکردن بعد بهینه برای آموزش استفاده می‌کنیم. برای این منظور از روش بیشترین احتمال مربع‌شی[۳] برای این‌کار استفاده می‌کنیم. بدین صورت که به ازای همه مقادیر { ۵، ۱۵، ۲۵، ۳۵،

---

[۱] request

[۲] Preprocessing

[۳] Chisqure test statistic

۴۵، ۴۹} بهترین ویژگی را بدست می‌آوریم و خروجی مدل را برای این ویژگی‌ها حساب می‌کنیم. بهترین تعداد ویژگی‌ها ۴۵ عدد است. ویژگی‌هایی که در داده‌ها کنار گذاشته می‌شوند عبارت‌اند از: changeInParams, PasswordInPath, PayInPath, ViewInParams

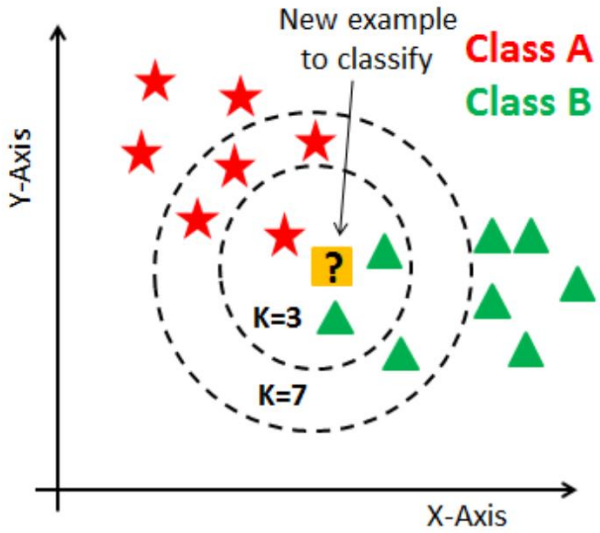
۳-۳- مدل سازی مساله

همان طور که از قسمت قبل متوجه شدیم، داده‌های ما به صورت برچسب دار هستند. حال باید دسته‌بندی مسائل موجود برای این دسته از مساله‌ها را بررسی کنیم و سعی کنیم مساله را به صورت یکی از این دسته از مسائل تبدیل کنیم.

به صورت کلی، الگوریتم‌های یادگیری ماشین برای داده‌های دارای برچسب به دو دسته زیر تقسیم می‌شود.

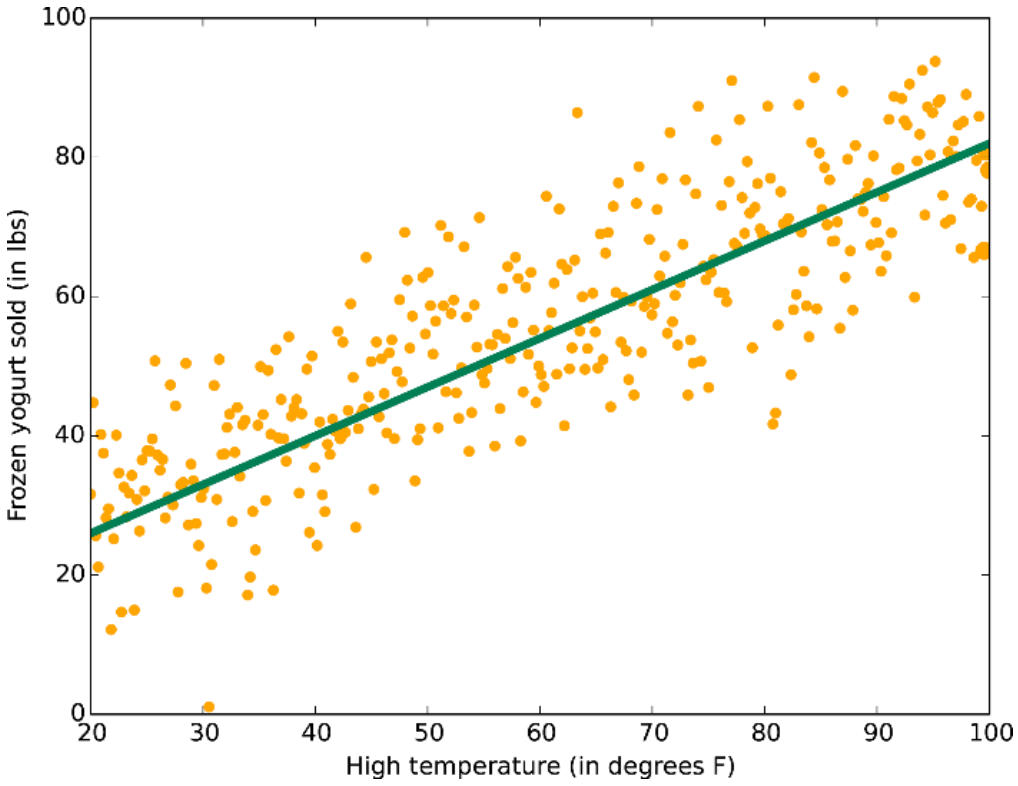
۱-الگوریتم‌های کلاس بندی: در این الگوریتم کلاس‌های مختلفی داریم که سعی میکنیم داده ورودی را به یکی از این کلاس‌ها مرتبط کنیم. برای مثال فرض کنید ویژگی‌های فیزیکی افراد مختلف (قد، وزن، فشار خون و غیره) را داریم و می‌خواهیم افراد را به دو گروه سالم و ناسالم

تقسیم کنیم. ساده‌ترین الگوریتم برای انجام این کار k نزدیک‌ترین همسایه می‌باشد. شکل شماره۶ نشان‌دهنده یک نمونه از این دسته‌بندی به ازای k های ۳و ۷ می‌باشد.



شکل ۶- تشخیص برچسب یک داده با استفاده از الگوریتم k نزدیک ترین همسایه[8]

۲- الگوریتم‌های رگرسیون: دراین الگوریتم‌ها سعی می‌کنیم تا خروجی مورد نظر برای هر داده را با توجه به داده‌هایی که تاکنون دیده‌ایم پیش بینی کنیم. برای مثال مقدار بارش باران در سال‌های متوالی را داریم و اکنون می‌خواهیم مقدار بارش باران برای امسال را پیش بینی کنیم. ساده‌ترین مدل رگرسیون، رگرسیون خطی می‌باشد که در شکل ۷ دیده می‌شود. در این شکل محتمل‌ترین مدل خطی موجود برای داده‌هایی که با نقاط نارنجی رنگ دیده می‌شوند رسم شده است.



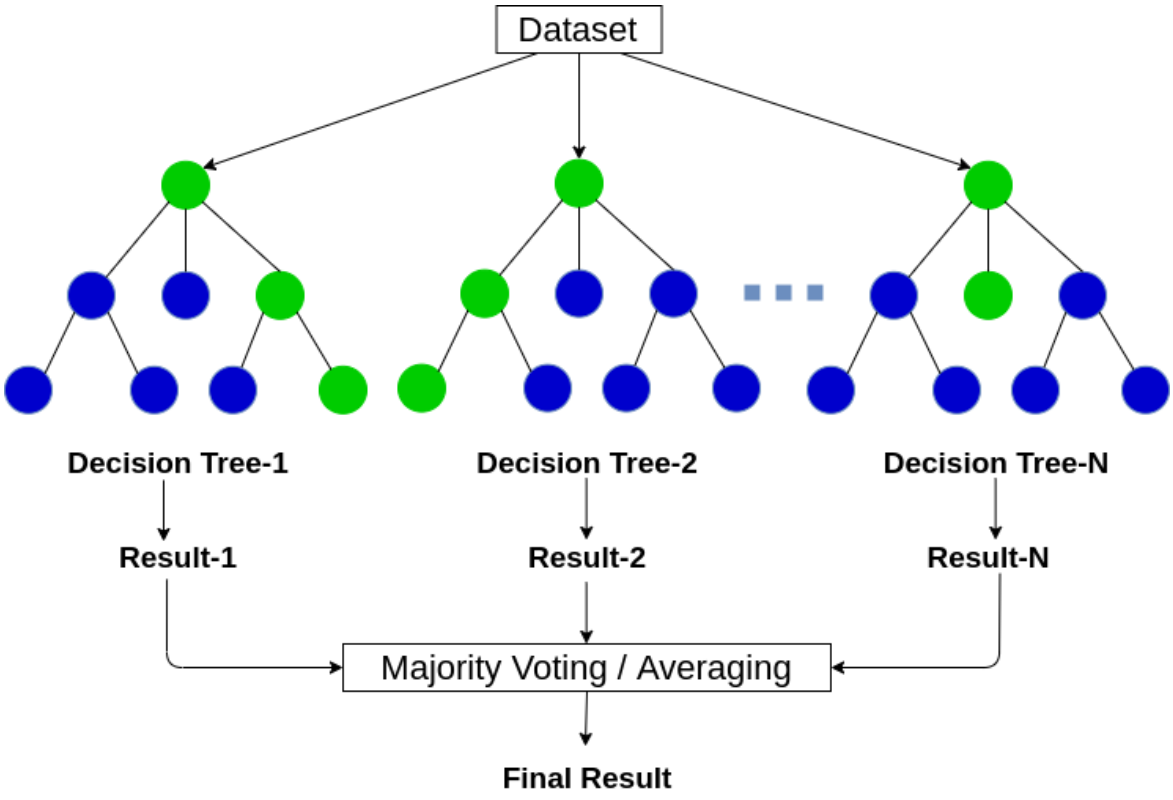
شکل ۷- رگرسیون خطی[9]

دسته دیگری از الگوریتم‌های یادگیری وجود دارند که در آن از یک مدل خاص برای حدس زدن مقدار یا برچسب داده استفاده نمی‌شود، بلکه در آن الگوریتم از چندین مدل برای حدس زدن خروجی استفاده می‌کنیم و سپس خروجی این مدل‌ها را ترکیب کرده و خروجی نهایی را بدست می‌آوریم. این دسته از الگوریتم‌ها انسامبل[۱] نامیده می‌شوند. یکی از ساده‌ترین و بهترین این الگوریتم ها، جنگل تصادفی است. این الگوریتم‌ها خود به دو دسته کیسه‌ای[۲] و تقویت‌شده[۳] تقسیم می‌شوند. شکل شماره ۷ یک نمونه از این الگوریتم را نشان می‌دهد که خروجی نهایی با استفاده از رای‌گیری مشخص شده است.

۱ Ensemble

۲ Bagging

۳ Boosting



شکل ۸- الگوریتم جنگل تصادفی[10]

اگر بخواهیم مساله پیدا کردن آسیب‌پذیری در صفحه وب را به صورت یک مساله دودویی نگاه کنیم می‌توانیم این مساله را به وجود یاعدم وجود آسیب‌پذیری تبدیل کنیم. به عبارت دیگر مساله ما به یک مساله کلاس‌بندی داده‌ها به کلاس باینری تبدیل شده است. برای مثال دو کلاس ۰ و ۱ داریم که کلاس ۰ نشان دهنده عدم وجود آسیب‌پذیری در یک داده و کلاس ۱ نشان دهنده وجود آسیب‌پذیری برای آن داده می‌باشد. در نهایت ما مساله وجود یاعدم وجود آسیب‌پذیری را به یک مساله کلاس‌بندی به دو کلاس تبدیل کردیم و از یادگیری نظارت‌شده برای یادگیری مدل استفاده می‌کنیم.

۳-۵- چرا از یادگیری ماشین استفاده کردیم؟

شاید برای پاسخ دادن به این سوال بهتر باشد که روش‌های موجود را بررسی کنیم و هدف مورد نظر را با هر یک از این روش‌ها در نظر بگیریم. رویکردهای مختلفی می‌تواند برای شناسایی آسیب‌پذیری CSRF

وجود داشته باشد. این رویکردها هر کدام می‌توانند مزایا و معایبی داشته باشند. در ادامه به دو دسته از ابزارهای موجود می‌پردازیم که می‌تواند آسیب‌پذیری CSRF را تشخیص دهد.

دیمن اولین ابزار خودکاری بود که برای تشخیص آسیب‌پذیری CSRF ساخته شده است. دیمن یک مدل براساس تست امنیتی چارچوب است که براساس پایش کردن در زمان اجرا ساخته شده است.[11] دیمن منبع باز بوده و بسیار کارآمد است اما به زبان php نوشته شده است برای همین وابسته به زبانی است که وبگاه با آن زبان نوشته شده است. علاوه بر آن باید به منبع کد صفحه برای تحلیل پویا دسترسی داشت.

از طرفی چون ممکن است به منبع کد صفحه دسترسی نداشته باشیم و در حال حاضر هم وبگاه‌های زیادی هستند که با تکنولوژی‌های مختلف توسعه داده می‌شوند استفاده از ابزار تحلیل جعبه‌سیاه[1] بسیار مناسب است. ابزارهای تحلیل جعبه سیاه آن دسته از ابزارهایی هستند که تمامی سیستم را به صورت جعبه‌ای که محتوی آن مخفی است در نظر می‌گیرند سپس سعی می‌کنند با استفاده از خروجی‌ها به ازای ورودی‌های مختلف سیستم را تحلیل کنند.

استفاده از پویشگر[2]ها برای تشخیص اسیب‌پذیری‌ها یکی از رایج‌ترین و شایع‌ترین ابزارها هستند اما تحقیقات قبلی نشان می‌دهد که پویشگرها ابزارهای کارامدی برای اینکار نیستند زیرا برای اینکه یک پویشگر بتواند خروجی مناسبی برای هر سیستمی داشته باشد باید برای آن سیستم طراحی شده باشد که این امر با توجه به رشد روزافزون وبگاه‌های اینترنتی غیرممکن است. برخی از پویشگرهای تجاری هم نمی‌توانند آسیب‌پذیری‌های csrf را تشخیص دهند زیرا تشخیص اینکه این آسیب‌پذیری دقیقا در کدام فرم قرار دارد کار سختی است. [1]

از میان راهکارهای متفاوتی که برای پیاده‌سازی سیستم‌مان به منظور تشخیص آسیب‌پذیری CSRF داشتیم، راهکار استفاده از یادگیری ماشین را استفاده می‌کنیم. این روش به نسبت روش‌های دیگر جامع‌تر

[1] Black Box

[2] Scanner

و فراگیرتر است زیرا وابسته به پلتفرم خاصی نیست. از طرفی دیگر این روش می‌تواند بسیاری از آسیب‌پذیری‌هایی که ممکن است توسط ابزارهای پویشگر شناسایی نشود را بشناسد.

۳-۶- انتخاب مدل

نکته قابل توجهی که وجود دارد انتخاب مدل مناسب با در نظر گرفتن شرایط و ویژگی های داده های موجود در مجموعه داده می باشد. در بخش قبلی ویژگی های موجود در مجموعه داده را بررسی کردیم حال با در نظر گرفتن این ویژگی ها سعی می کنیم تا بهترین مدل را برای یادگیری انتخاب کنیم. از مدل های زیر به منظور پیدا کردن بهترین مدل استفاده می کنیم:

۱- Logestic Regression(LR)
۲- Supported vector machine(SVM)
۳- Decision tree(DT)
۴- Random Forest(RF)
۵- Gradient Boosted Decision(GBD)

به این دلیل که اندازه مجموعه داده کوچک می باشد پس احتمالا مدل های انسامبل[۱] و درخت تصمیم می تواند خروجی مناسبی داده باشد. بهترین خروجی ناشی از جنگل تصادفی است. اگرچه خروجی های مدل Gradient Boosted Decision با جنگل تصادفی تقریبا یکسان است اما چون این الگوریتم زمان بیشتری برای پیشبینی دارد پس از الگوریتم جنگل تصادفی استفاده می کنیم.

۳-۶-۱- بدست آوردن پارامترهای مدل

پس از این که مدل RF را به عنوان بهترین مدل شناسایی کردیم حال باید مقادیر فراپارامتر[۲] مربوط به آن را مشخص کنیم. برای این کار مدل را با مقادیر مختلف فراپارامتر تست می‌کنیم و سعی می‌کنیم که بهترین مقادیر را پیدا کنیم.

---

[۱] Ensemble

[۲] Hyper-parameter

مدل را با تعداد ۱۰۰ و ۵۰۰ و ۱۰۰۰ درخت تست می‌کنیم و از خروجی‌های تست متوجه می‌شویم که بهترین مقدار برای تعداد درختان ۵۰۰ درخت می‌باشد. در صورتی که تعداد درختان زیادتر شود یادگیری مدل طولانی‌تر خواهد شد علاوه بر آن تجربه به ما نشان می‌دهد که در صورتی که تعداد درختان دو برابر یا بیشتر از این حد شود خروجی به همین مقدار افزایش نخواهد یافت برای همین استفاده از ۵۰۰ درخت برای اینکار مناسب است.

۳-۶-۲- نحوه آموزش مدل و خروجی مدل

۹۰ درصد از داده‌ها را برای آموزش و ۱۰ درصد را برای تست نگه می‌داریم. چون کلاس‌های داده نامتعادل هستند (۱۶ درصد از داده‌ها حساس و ۸۴ درصد از داده‌ها مربوط به کلاس غیرحساس هستند) به همین منظور برای ارزیابی مدل‌هایمان از روش stratified random sampling استفاده می‌کنیم. با این روش ما این تضمین را داریم که در هر دو بخش تست و آموزش از یک توزیع یکسانی برای کلاس‌های مختلف برخوردار هستیم. این کار باعث می‌شود تا بایاس[۱] انتخاب‌هایمان کاهش پیدا کند. [1] این نکته را باید خاطر نشان کرد که مدلی که در نهایت استفاده می‌کنیم را روی تمامی داده‌ها آموزش داده‌ایم.

۳-۷- خلاصه

در این فصل به روش های یادگیری ماشین پرداختیم و هم پوشانی آن را با مساله خودمان در نظر گرفتیم. سپس سعی کردیم که ازمیان مدل های مختلف، مدلی را به عنوان مدل یادگیری ماشین انتخاب کنیم که بهترین کارایی را باتوجه به داده های موجود در مجموعه داده داشته باشد. مدل ها را با داده ها آموزش داده و خروجی آن ها را برای داده های تست بررسی کردیم.

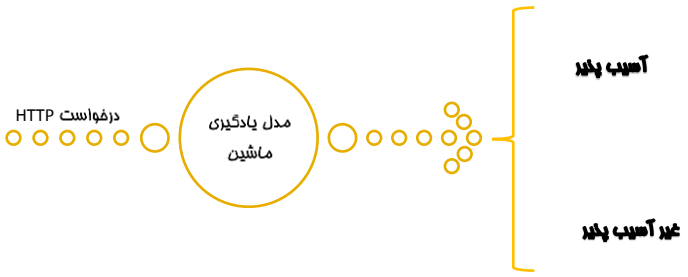
[۱] Bias

فصل چهارم - پیاده سازی و چالش ها

در این فصل ساختار سیستم کشف آسیب‌پذیری و نحوه پیاده‌سازی آن را بیان می‌کنیم. سپس چالش‌های موجود برای پیاده‌سازی را بررسی می‌کنیم و معیارهای لازم برای ارزیابی را بیان می‌کنیم و راهکارهای یافته شده برای کاهش خطا را بیان می‌کنیم در نهایت مزایای نرم‌افزار را نسبت به سایر مدل‌های مشابه بیان می‌کنیم.

۴-۱- ساختمان پوینده

برنامه اصلی ما از دو قسمت اصلی تشکیل شده است. قسمت اول مربوط به شیوه پیاده‌سازی مدل یادگیری ماشین (مطابق شکل ۱۰) و بخش دوم مربوط به شبکه و نحوه ارسال و دریافت درخواست‌ها می‌باشد. پیاده‌سازی با استفاده از زبان پایتون[۱] انجام شده است. این زبان یکی از قوی‌ترین و رایج‌ترین زبان‌های برنامه‌نویسی است که کتابخانه‌های مناسبی برای اینکار دارد. برای پیاده‌سازی قسمت اول از کتابخانه سایکت لرن استفاده می‌کنیم. سایکت لرن[۲] یک کتابخانه یادگیری ماشین برای زبان پایتون است که الگوریتم‌های کلاس بندی، رگرسیون و خوشه‌بندی را می‌تواند پیاده‌سازی کند بنابراین تمام مدل‌های پیشنهاد داده شده در فصل قبلی را می‌توان با استفاده از این کتابخانه پیاده‌سازی کرد. کدهای مربوط به پیاده‌سازی مدل یادگیری ماشین در قسمت پیوست قرار داده شده است.



شکل ۹- ساختار مدل یادگیری ماشین برای کشف آسیب پذیری

[۱] Python

[۲] Scikit-learn

قسمت دوم سیستم، وظیفه ارسال درخواست‌ها در نشست‌های متفاوت و حدس آسیب‌پذیری CSRF را دارد. برای قسمت دوم می‌توانیم از کتابخانه‌های متفاوتی استفاده کنیم اما برای اینکه کاربر خود بتواند احراز هویت را انجام دهد از کتابخانه سلنیوم وایر[۱] استفاده می‌کنیم. این کتابخانه یک کتابخانه گسترش یافته از کتابخانه سلنیوم[۲] است که می‌تواند اجازه دسترسی به درخواست‌ها و پاسخ‌های ارسال شده توسط مرورگر را به ما می‌دهد. برای اینکار ابتدا برنامه یک درایور[۳] مربوط به مرورگر مورد نظر ما را تولید می‌کند سپس درخواست‌های کاربر می‌تواند از دو طریق، API مربوط به کتابخانه و یا به صوت دستی توسط خود کاربر می‌تواند ارسال شود. [۱۰] این کتابخانه قابلیت‌های زیادی دارد که هنگام پیاده‌سازی هر مورد به آن‌ها اشاره خواهیم کرد.

شاید برای فهم برنامه بهتر است با استفاده از نمودار کلاس[۴] مربوط به این سیستم را بررسی می‌کنیم. ساختار اصلی برنامه از چهار کلاس موجود در شکل ۱۰ ساخته شده است. کلاس اصلی که مسئول مدیریت سیستم را برعهده دارد پایمیچ[۵] است. این کلاس با سه کلاس مدل[۶]، حساسیت[۷] و حدسCSRF[۸] در ارتباط است. کلاس اول مسئول ساخت و آموزش مدل بوده، کلاس دوم هر درخواست را به دو دسته حساس و غیرحساس تقسیم‌بندی می‌کند و کلاس آخر نیز در نهایت همه دستورات حساس را بررسی کرده و آن دسته از دستوراتی که دارای آسیب‌پذیری CSRF هستند را به عنوان خروجی بر می‌گرداند.

---

۱ Seleniumwire

۲ Selenium

۳ driver

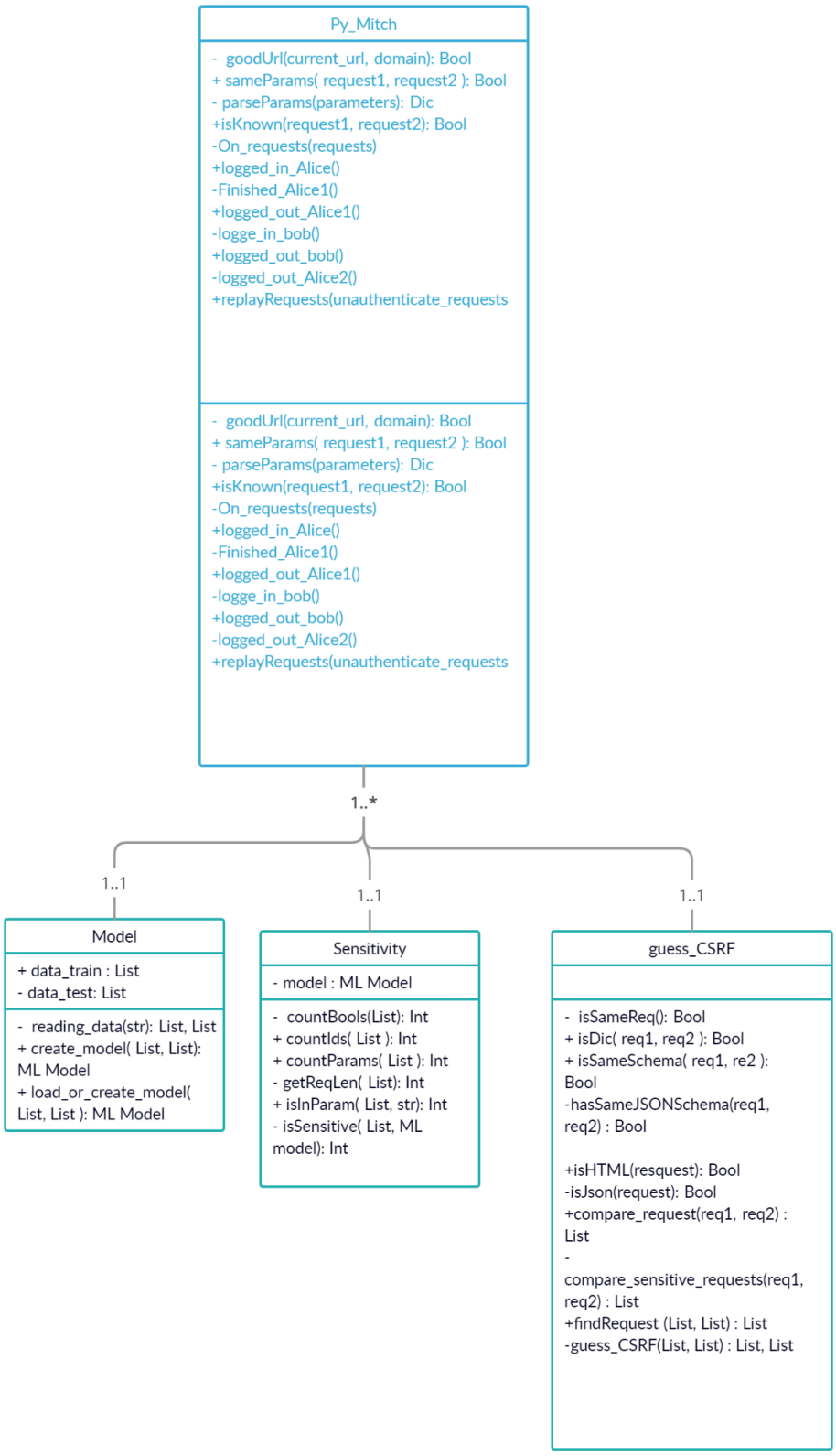
۴ Class Diagram

۵ Py_Mitch

۶ Model

۷ Sensitivity

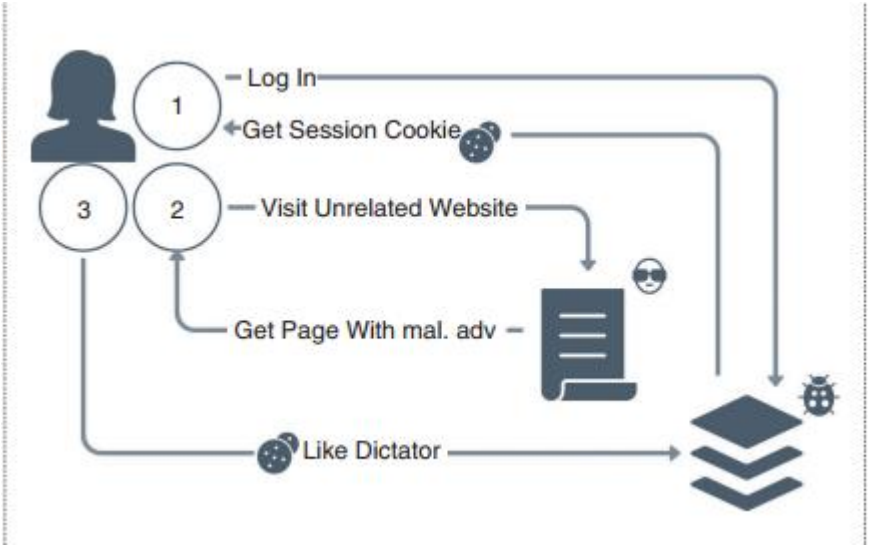
۸ Guess_CSRF



شکل ۱۰- نمودار کلاس سیستم

۴-۲- ساختار حمله CSRF

ساختار یک حمله csrf در شکل ۱۱ نشان داده شده است.



شکل ۱۱- یک سناریو از حمله CSRF [1]

سناریو حمله ی شکل شماره ۱۱ به صورت زیر است:

۱- کاربر دریک وبگاه اسیب پذیر احرازهویت می کند در این حالت یک جلسه احراز هویت شده از طریق کوکی ها بین برنامه وب و مرورگر برقرار می شود.

۲- از طریق همین مرورگر کاربر یک سایت مخرب را بازدید می کند. این سایت در خواست های cross site- را با استفاده از مرورگر و کوکی های کاربر برای سایت آسیب پذیر ارسال می کند.

۳- چون در خواست ها شامل کوکی های کاربر برای آن وبگاه هستند برای همین این درخواست ها به عنوان درخواست هایی از سمت کاربر در نظر گرفته می شود.

در این حالت آن وبگاه توانسته است که چک های امنیتی را دور بزند.

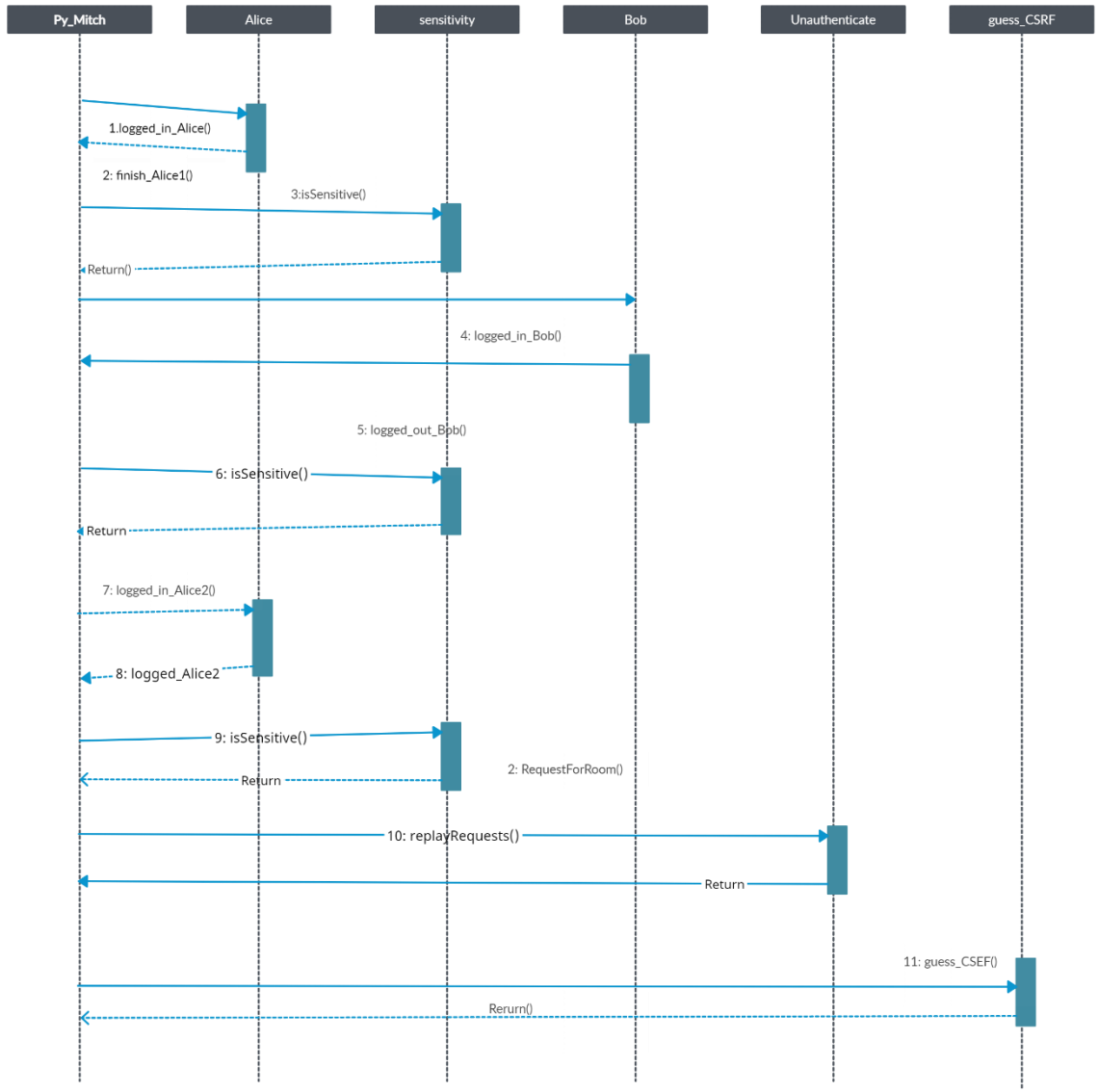
۴-۳- سناریو کارکرد برنامه

برای فهم بهتر کارکرد برنامه از نمودار توالی وقایع در شکل ۱۲ استفاده می‌کنیم. برای پیاده‌سازی این سناریو باید از دو جلسه مربوط به دو کاربر متفاوت استفاده کنیم. کاربر اول که همان حمله کننده می‌باشد را آلیس[۱] نامیده و کاربر دوم را که کاربری عادی می‌باشد باب[۲] می‌نامیم. سناریو اصلی‌ای که ما از آن استفاده می‌کنیم مانند آن است که حمله کننده (آلیس) درخواست‌های http حساس را از طریق جلسه مربوط به مرورگر قربانی جعل و ارسال می‌کند به همین دلیل داشتن دو اکانت برای این سناریو الزامی است. از این رو در این سناریو اگر درخواست‌های مربوط به الیس دارای ویژگی‌ای باشند که مربوط به الیس باشد در این صورت csrf امکان‌پذیر نیست.

به ازای درخواست‌هایی از الیس که حساس هستند، برنامه محتوی پاسخ‌های این درخواست‌ها را نگهداری می‌کند سپس این درخواست‌ها را در سشن مربوط به باب ارسال می‌کند و پاسخ‌های آن‌ها را نیز به دست می‌اورد. در نهایت پاسخ‌های مربوط به الیس و باب را با یکدیگر مقایسه می‌شوند اگر پاسخ‌ها یکسان بودند به این معنی است که الیس توانسته است یک درخواست به داخل درخواست‌های باب جعل کند از این رو برنامه می‌تواند این موقعیت را به عنوان یک آسیب‌پذیری گزارش کند.

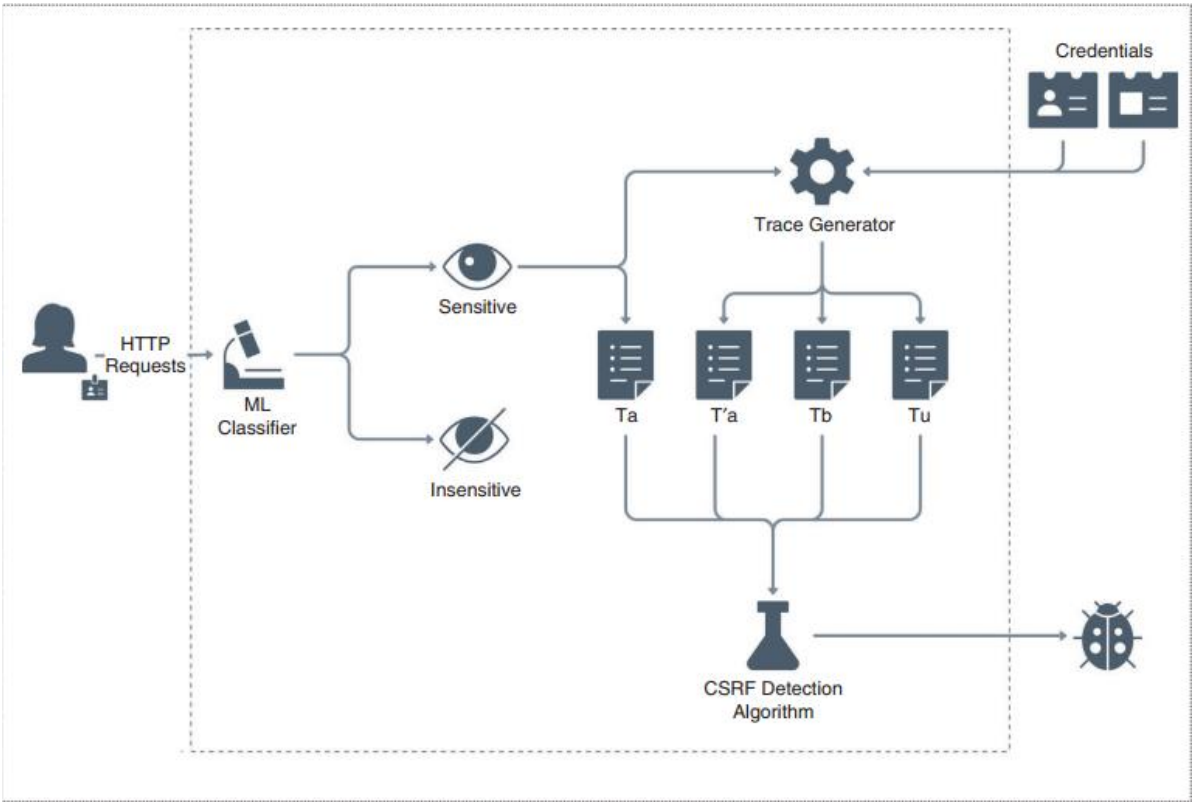
[۱] Alice

[۲] Bob



شکل ۱۲- نمودار توالی سیستم

حال توانستیم مراحل مختلف برنامه را شرح دهیم. در ادامه توضیح می‌دهیم که در هر مرحله چگونه درخواست‌های حساس که می‌توانند آسیب‌پذیر باشند را از سایر درخواست‌های تشخیص می‌دهیم. شکل۱۳، روش کارکرد برنامه را نشان می‌دهد.



شکل ۱۳- روش کارکرد برنامه [1]

درخواست ورودی ابتدا پردازش می‌شود و به فرمت ورودی مناسب برای مدل یادگیری ماشین تبدیل می‌شود. سپس مدل این درخواست را بررسی می‌کند و تشخیص می‌دهد که آیا این درخواست آسیب‌پذیر می‌باشد یا نه. در صورتی که درخواست حساس باشد، در دنبال کننده قرار می‌گیرد تا همین درخواست به ازای حالت‌های دیگر از قبیل حالت بدون احراز هویت (Tu[۱])، باب (Tb[۲]) و آلیس مرحله دوم (T'a[۳]) محاسبه شود. سپس تمامی حالت‌های تولید شده به ازای این درخواست به الگوریتم تشخیص CSRF داده می‌شود.

نمودار مربوط به مواردااستفاده[۴]به صورت شکل ۱۴ خواهد بود.

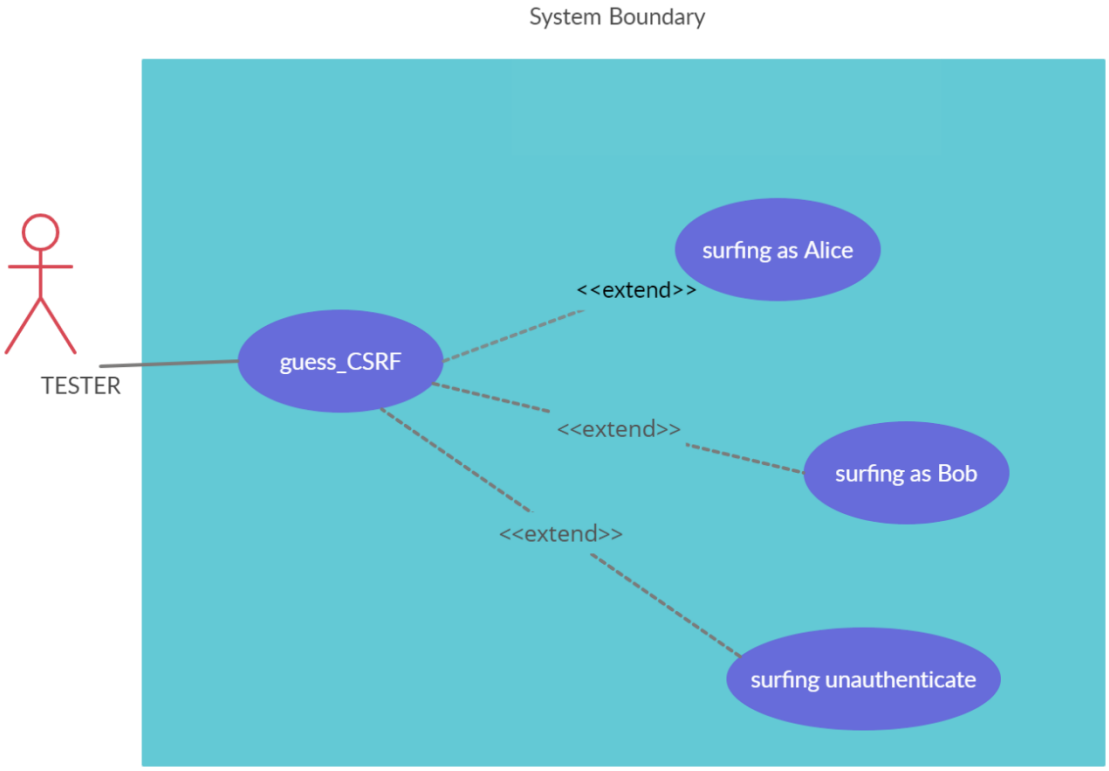
---

۱ Unauthenticated Trace

۲ Bob Trace

۳ Alice Trace again

۴ Use Case Diagram



شکل ۱۴- نمودار موردکاربرد سیستم

۴-۴- نکات پیاده سازی

برای استفاده از برنامه کافیست نیازمندی های برنامه را از طریق فایل requirement.txt و با استفاده از pip نصب کنیم. از دستور زیر می توانیم برای نصب نیازمندی ها استفاده کنیم.

Pip install -r requirement.txt

سپس لینک هایی از برنامه را که می خواهیم وجود یا عدم وجود آسیب پذیری در آن ها چک شود را به عنوان ورودی به برنامه می دهیم. در نهایت برنامه یک خروجی از لینک هایی که می توانند آسیب پذیر باشند را به عنوان خروجی برمی‌گرداند.

۴-۵- روش های ارزیابی

شاید بهتر باشد برای ارزیابی برنامه خروجی‌های ممکن آن را بررسی کنیم. جدول ۱ می‌تواند حالت‌های ممکن برای خروجی برنامه و خروجی واقعی را نشان دهد. منظور از خروجی واقعی، برچسب واقعی مربوط به یک داده است. برای مثال ممکن است برچسب یک داده ۱ باشد (از فصل دوم به خاطر داریم که برچسب ۱ نشان دهنده آسیب‌پذیر بودن آن داده است) ولی خروجی برنامه برای آن داده صفر باشد در این حالت خروجی مثبت کاذب است.

جدول ۱- حالت های ممکن خروجی سیستم

	کلاس های پیش بینی شده		
کلاس واقعی داده		برچسب کلاس =۱	برچسب کلاس = ۰
	برچسب کلاس = ۱	مثبت واقعی[۱]	منفی کاذب[۲]
	برچسب کلاس = ۰	مثبت کاذب[۳]	منفی واقعی[۴]

سبرای ارزیابی کارایی برنامه از معیار F1[۵] استفاده می کنیم. این معیار ترکیبی از معیارهای Percision و recall است. به عبارت دیگر این نرخ می تواند مقدار منفی کاذب و مثبت کاذب را به خوبی نمایش دهد.

۱ True Positive

۲ False Negative

۳ False Positive

۴ True Negative

۵ F1-measure

نکته دیگر این است از آن جایی که کلاس داده ها نامتعادل[۱] هستند برای همین استفاده از این معیار می تواند برای ارزیابی مدل ما مناسب باشد.

$$F1 = \frac{2 \times \text{Percision} \times \text{R}ecall}{\text{Percision} + Recall} = \frac{\text{TP}}{\text{TP} + \frac{1}{2}(FP + FN)}$$

خروجی مدل ما به صورت زیر خواهد بود:

system	Precision	recall	F1
RF	0.78	0.67	0.72

۴-۶- چالش های پیاده سازی

یکی از بزرگترین چالش‌ها برای ساخت این ابزار تشخیص درخواست‌های HTTP به صورت درست بوده است. در میان روش‌های ابتکاری[۲]که قبلا وجود داشته‌اند [۱] مقدار مثبت کاذب بسیار زیاد بوده است، به همین منظور از شرطهایی برای شناسایی آسیب‌پذیری برای لینک‌ها استفاده می‌شود. این شرطها باعث می‌شود که نرخ مثبت کاذب بسیار کاهش پیدا کند. این شرطها را در ادامه ذکر می‌کنیم.

۱- در خواست حساس: یک درخواست HTTP را وقتی حساس محسوب می‌کنیم اگر و تنها اگر الف) باعث تغییرات امنیتی مربوط در حالت برنامه وب در حال پردازش آن بشود. ب) و درون یک محتوی[۳] احراز هویت شده مربوط به یک کاربر بشود.

۲- یک درخواست وقتی حساس و قابل دسترس در نظر گرفته می‌شود که اگر و تنها اگر حفره امنیتی مربوط به آن تغییر حالت پس از اجرای پردازش پاسخ مربوط به درخواست درون مرورگر قابل نمایش باشد

---

۱ Imbalanced Data

۲ heuristic

۳ Context

مثال‌های مربوط به این دسته بسیار زیاد هستند و عموما در مقابل حملات csrf مقاوم‌سازی نمی‌شوند. مثل آیکون[1] های موجود در شبکه‌های اجتماعی و گزینه‌های مربوط به تبادل اطلاعات بین برنامه‌های وب.

چالش دیگری که برای پیاده‌سازی این برنامه داشتیم احراز هویت برای هر وبگاه بود. در ابتدا تلاش شده است که این قسمت به صورت خودکار پیاده‌سازی شود به عبارت دیگر برنامه به صورت خودکار کاربر را برای آن وبگاه احراز هویت کند اما پیاده‌سازی این سیستم می‌تواند از نظر امنیتی چالش بزرگی داشته باشد. علاوه بر آن در این صورت دسترسی نگهداری از رمز عبور کاربر برای سایت‌های مختلف به برنامه داده می‌شود که از منظر امنیتی می‌تواند مخاطره‌آمیز باشد بدین منظور احراز هویت برای هر کاربر به صورت دستی و توسط خود کاربر انجام شود و سایر قسمت‌های دیگر برنامه به صورت خودکار انجام می‌شود.

۴-۷- ویژگی های نرم افزار

نرم‌افزار خروجی ما نرم افزاری قابل اجرا بر روی تمامی سیستم‌ها می‌باشد. نرم افزاری که توانستیم پیاده‌سازی کنیم به صورت خودکار عمل می‌کند، به عبارت دیگر تمامی قسمت‌های نرم‌افزار بجز احزار هویت توسط وبگاه، به صوت خودکار اجرا می‌شود. علاوه بر آن مدلی که استفاده می‌کند در برابر حملات adversarial نیز مقاوم است. ویژگی اصلی این نرم‌افزار عدم وابستگی به سیستم عامل خاص است.

۴-۸- خلاصه

در این فصل با ابزارهای موجود برای پیاده‌سازی مدل آشنا شدیم و پس از آن سناریو اصلی پیاده‌سازی سیستم را بررسی کردیم و نیازمندی‌ها را متناسب با سناریو بررسی کردیم. چالش‌های موجود هنگام پیاده‌سازی را رفع و بررسی کردیم و در نهایت ویژگی‌های سیستم پیاده‌سازی شده را با سیستم‌های قبلی بررسی کردیم.

[1] Icon

منابع و مراجع

[1] S. Calzavara, M. Conti, R. Focardi, A. Rabitti and G. Tolomei, "Machine Learning for Web Vulnerability Detection: The Case of Cross-Site Request Forgery," in IEEE Security & Privacy, vol. 18, no. 3, pp. 8-16, May-June 2020, doi: 10.1109/MSEC.2019.2961649.

[2] PwnFunction, "Cross-Site Request Forgery (CSRF) Explained," , 2019.

[3] P. team, "SQL Injection," PortSwigger, 2021. [Online]. Available: https://portswigger.net/web-security/sql-injection. [Accessed 2021].

[4] P. team, "cross site scripting," web security academy, 2021. [Online]. Available: https://portswigger.net/web-security/cross-site-scripting. [Accessed 2021].

[5] D. Caissy, "Owasp Top 10 - 2017," July 2017. [Online]. Available: https://owasp.org/www-pdf-archive//OWASP_LA_New_OWASP_Top_10_David_Caissy_2017_07.pdf. [Accessed July 2017].

[6] S. Calzavara, M. Conti, R. Focardi, A. Rabitti and G. Tolomei, "Mitch: A Machine Learning Approach to the Black-Box Detection of CSRF Vulnerabilities," 2019 IEEE European Symposium on Security and Privacy (EuroS&P), 2019, pp. 528-543, doi: 10.1109/EuroSP.2019.00045.

[7] G. P. Soheil Khodayari, "JAW: Studying Client-side CSRF with Hybrid Property Graphs," CISPA Helmholtz Center, 2021.

[8] R. Shah, "Introduction to k-Nearest Neighbors (kNN) Algorithm," ai.plainenglish, 2021. [Online]. Available: https://ai.plainenglish.io/introduction-to-k-nearest-neighbors-knn-algorithm-e8617a448fa8.

[9] R. Bhatia, "analyticsindiamag," 19 09 2017. [Online]. Available: https://analyticsindiamag.com/top-6-regression-algorithms-used-data-mining-applications-industry/.

[10] H. Ampadu, "Random Forest understanding," AI Pool, 01 04 2021. [Online]. Available: https://ai-pool.com/a/s/random-forests-understanding.

[11] Giancarlo Pellegrino, Martin Johns, Simon Koch, Michael Backes, Christian Rossow, "Deemon: Detecting CSRF with Dynamic Analysis and Property Graph," *arXiv*, 2017. [Online]. Available: https://arxiv.org/pdf/1708.08786.

[12] O. C. S. Team, "Cross-Site Request Forgery Prevention Cheat Sheet," GitHub, [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html.

[13] O. team, "Injection," Owasp, 2017. [Online]. Available: https://owasp.org/www-project-top-ten/2017/A1_2017-Injection. [Accessed 2017].

[14] KirstenS, "XSS," Owasp , [Online]. Available: https://owasp.org/www-community/attacks/xss. [Accessed 25 Apr 2021].

[15] N. P. S. H. R. D. P. A. J. C. Ian Goodfellow, "adversarial example research," 24 Febrary 2017. [Online]. Available: https://openai.com/blog/adversarial-example-research/.

[16] W. Keeling, "Selenium-Wire," PyPi, 2 Aug 2018. [Online]. Available: https://pypi.org/project/selenium-wire/. [Accessed 2021 Jun 2021].

[17] Acunetix, "the Acunetix Web Application," Acunetix, USA, 2019.

[18] ilmoi, "will my Machine Learning system be attacked?," towards data science, 15 07 2019. [Online]. Available: https://towardsdatascience.com/will-my-machine-learning-be-attacked-6295707625d8.

[19] N. P. S. H. R. D. ,. P. A. J. C. Ian Goodfellow, "Attacking Machine Learning with adversarial examples," 24 2 2017. [Online]. Available: https://openai.com/blog/adversarial-example-research/. [Accessed 24 2 2017].

[20] S. B. M. K. L. B. Mahmood Sharif, "Accessorize to a Crime: Real and Stealthy Attacks on state-Of-Art Face Recognition," vol. 15, p. 10, 2016.

[21] Jonasguan, "cleverhans-lab," cleverhans, 11 2 2021. [Online]. Available: https://github.com/cleverhans-lab/cleverhans. [Accessed 21 1 2016].

پیوست‌ها

پیوست الف- کدهای بخش اصلی برنامه.

جدول الف- ۱: کلاس اصلی برنامه.

```
from seleniumwire import webdriver
from urllib.parse import urlparse
import multiprocessing

# inner imports :
from sensitivity import *
from model import *
from guess_csrfs import *


class Py_Mitch:
    """This class contain the main program
        inputs:
            browser_is : introduce the type of the browser to be used by
Mitch
            url : this is the url of the login page
            domain : domain of the web page
    """
    def __init__(self, browser_is="chrome", url="https://darmankade.com",
domain="darmankade.com"):
        self.log_sensitivity = {}
        self.phase = 0
        if browser_is == "chrome":
            self.driver = webdriver.Chrome()
        elif browser_is == "firefox":
            self.driver = webdriver.Firefox()
        else:
            self.driver = webdriver.Safari()
        self.main_domain = domain
        self.driver.get(url)
        self.lock = multiprocessing.Lock()

        # urls that mitch search for vulnerabilities in them
        self.search_urls = []
        self.active_collector = []
        self.alice1_requests = []
        self.sensitive_requests = []
        self.main_sensitive_req = []
        self.bob_requests = []
```

```
        self.candidates = []
        self.null_collector = []
        self.unauth_requests = []
        self.collected_sensitive_requests = 0
        self.collected_total_request = 0
        self.classifier = load_or_create_model()
```

جدول الف- ۲: استخراج داده های ورودی از درخواست ها و ارسال این ورودی ها به مدل.

```
def goodUrl(self, current_url, domain):
    # s = urlparse(request_url)
    u = urlparse(current_url)
    isGood = True

    if not u.scheme.startswith('http'):
        isGood = False
    if u.path.endswith('/chrome/newtab'):
        isGood = False
    if domain not in current_url:
        isGood = False
    return isGood


def sameParams(self, a, b):
    flag = True
    if len(a.keys()) != len(b.keys()):
        flag = False
    else:
        for k in a.keys():
            if k not in b.keys():
                flag = False
    return flag

def compareReq(self, a, b):
    return a['method'] == b['method'] and a['url'] == b['url'] and
self.sameParams(a, b)

def isKnown(self, r, gs):
    flag = False
    if gs == []:
        return False
    if not gs:
        flag = True

    for g in gs:
        if self.compareReq(g, r):
            flag = True
    return flag

def parseParams(self, param):
    p = {}
    for k in param.keys():
        if p.get(k) is not None:
            p[k] = p.get(k)
```

```
    return p


def on_requests(self, arr):
    for request in self.driver.requests:
        req = {}
        # visiting url is the url of the main frame from the browser
        isMainFrame = False
        # check for the main frame
        if request.response is not None and
request.response.headers['Content-Type'] is not None:
            if 'html' in request.response.headers['Content-Type'] and
self.main_domain in request.url:
                isMainFrame = True
                # print("we have a main frame in : ", request)

        if isMainFrame and self.goodUrl(request.url, self.main_domain):
            req['method'] = request.method
            o = urlparse(request.url)
            req['url'] = o.scheme + "://" + o.hostname + "//" + o.path
            req['reqId'] = request.id
            req['response'] = {}
            # trying to take the body --> taking body from the driver
            try:
                req['response']['body'] = self.driver.page_source
                # req['response']['body'] =
request.response.body.decode('utf-8')
            except:
                print("except in body decoding")
            req['params'] = self.parseParams(o._asdict())
            if request.method == "POST":
                if request.body.decode('utf-8') != '':
                    postBody = {}
                    data = request.body.decode('utf-8')
                    if data.startswith("{"):
                        data = data.replace('{', '')
                        data = data.replace('}', '')
                        data = data.split(',')
                        for d in data:
                            k, v = d.split(':')
                            postBody[k[0].replace('\'', '')] =
v[0].replace('\'', '')
                    else:
                        # check this part .................????????????????/
                        # print("data is : ", data)
                        data = data.replace("%5B", '[')
                        data = data.replace("%5D", ']')
                        temp = data.split("&")
                        for t in temp:
                            d = t.split("=")
                            postBody[d[0]] = [d[1]]
                    for k in postBody.keys():
                        req['params'][k] = postBody[k]
                    # here remains --- line 140 to 160
```

```
            # print("req is: ", req)

            sen = False
            sensitivity = isSensitive(req, self.classifier)
            if isinstance(sensitivity, int):
                if sensitivity == 1:
                    sen = True
                    # print("sensitive Request in : ", req['url'])
            else:
                if sensitivity != 'n':
                    sen = True
                #     print("sensitive Request in : ", req['url'])
                # else:
                #     print("here we have trouble : ", sensitivity)

            if sen and (not self.isKnown(req, arr)):
                req['headers'] = request.headers
                self.main_sensitive_req.append(request)
                # add code line 179 to 185
                req['response']['status'] = request.response.status_code
                if request.response.headers is not None:
                    headers = {}
                    for k in request.response.headers.keys():
                        headers[k] = request.response.headers[k]
                    if headers != {}:
                        req['response']['headers'] = headers

                arr.append(req)
                # print("sensitive request is added", req)
                self.collected_sensitive_requests += 1
            # add to total requests
            self.collected_total_request += 1
        del self.driver.requests
```

جدول الف- ۳: توابع فاز اول

۱- احراز هویت حمله کننده و ارسال درخواست ها

۲- ۲- اطمینان از پایان یافتن جلسه حمله کننده

۳- ۳- خروج حمله کننده

```
def logged_in_Alice(self):
    self.lock.acquire()
    # here we need to add another phase for starting Alice
    print("Alice logged in ")
    # self.active_collector = self.sensitive_requests
    del self.driver.requests
    self.phase = 0
    self.call_url(self.search_urls, self.sensitive_requests)
    print("alice sensitive requests are : ", self.sensitive_requests)
    self.lock.release()
```

```
def finished_Alice1(self):
    self.lock.acquire()
    print("Alice run finished, preparing CSRF test forms...")
    print("Please logout from the current session and notify the extension")
    input()
    self.phase = 1
    self.lock.release()

def logged_out_Alice1(self):
    print("Alice logged out, please login as Bob and notify the extension")
    self.lock.acquire()
    input()
    self.phase = 2
    self.lock.release()
```

جدول الف- ۴: توابع کاربر ساده (باب[۶۸])

۱- احراز هویت باب و ارسال درخواست ها

۲- خروج باب

```
def logged_in_bob(self):
    self.lock.acquire()
    print("Logged in as Bob, testing sensitive requests...")
    del self.driver.requests
    # self.active_collector = self.bob_requests
    # self.replayRequests(self.bob_requests)
    self.call_url(self.search_urls, self.bob_requests)
    print("bob requests are: ", self.bob_requests)
    print("...please logout from Bob's account and notify the extension")
    input()
    self.phase = 3
    self.lock.release()

def logged_out_bob(self):
    print("Logged out as Bob, please login as Alice again and notify the
extension")
    input()
    self.active_collector =[]
    self.phase = 4
```

جدول الف- ۵: ارسال درخواست های حمله کننده در فاز دوم

```
def logged_in_Alice2(self):
    print("Logged in as Alice again, testing sensitive requests...")
```

[۶۸] Bob

```
    self.lock.acquire()
    del self.driver.requests
    # self.active_collector = self.alice1_requests
    # self.replayRequests(self.alice1_requests)
    self.call_url(self.search_urls, self.alice1_requests)
    print("...please logout from Alice's account and notify the extension")
    input()
    self.phase = 5
    self.lock.release()
```

جدول الف- ۶: ارسال درخواست ها بدون هیچ احراز هویت

```
def replayRequests(self, unauth_requests):
    import requests
    session = requests.Session()
    for r in self.main_sensitive_req:
        req = {}
        req['method'] = r.method
        o = urlparse(r.url)
        req['url'] = o.scheme + "://" + o.hostname + "//" + o.path
        req['reqId'] = r.id
        req['params'] = r.params
        if r.method.upper() == "POST":
            if r.body.decode('utf-8') != '':
                postBody = {}
                data = r.body.decode('utf-8')
                if data.startswith("{"):
                    data = data.replace('{', '')
                    data = data.replace('}', '')
                    data = data.split(',')
                    for d in data:
                        k, v = d.split(':')
                        postBody[k[0].replace('\'', '')] = v[0].replace('\'',
'')
                else:
                    # print("data is : ", data)
                    data = data.replace("%5B", '[')
                    data = data.replace("%5D", ']')
                    temp = data.split("&")
                    for t in temp:
                        d = t.split("=")
                        postBody[d[0]] = [d[1]]
                for k in postBody.keys():
                    req['params'][k] = postBody[k]
        if r.method.upper() == "POST":
            res = session.request(method=r.method, url=r.url, params=r.body,
headers={"Content-type": "application/x-www-form-urlencoded; charset=UTF-8"})
            # session.request(method=r.method, url=r.url,
data=data.encode("utf-8"),
            #                 headers={"Content-Type": "application/json;
charset=UTF-8"})
```

```
            # session.request(method=r.method, url=r.url,
data=json.dumps(postBody),
            #                 headers={"Content-Type": "application/json;
charset=UTF-8"})

        else:
            res = session.po
        # res = session.get(r['url'], data=r['params'], headers=r['headers'])
        req['response'] = {}
        req['response']['body'] = res.text

        req['response']['status'] = res.status_code
        if res.headers is not None:
            headers = {}
            for k in res.headers.keys():
                headers[k] = res.headers[k]
            if headers != {}:
                req['response']['headers'] = headers

        unauth_requests.append(req)


def logged_out_Alice2(self):
    print("Logged out as Alice, testing unauth sensitive requests...")
    self.lock.acquire()
    # self.active_collector = self.unauth_requests
    del self.driver.requests
    self.replayRequests(self.unauth_requests)
    print("all data collected")
    self.phase = 6
    self.lock.release()
```

جدول الف- ۷: اجرای برنامه و نتیجه گیری در فاز پایانی

```
# we change the guessCSRFs function and removes tellCSRFs
def make_conclusion(self):
    print("making conclusion")
    candidates, resulting_candidates = guessCSRFs(self.sensitive_requests,
self.alice1_requests, self.bob_requests, self.unauth_requests)
    print("search for possible CSRFs finished, please expand the array
presented here to see candidates:")
    print(candidates)
    print("resulting candidates are :")
    print(resulting_candidates)
    # results_url = tellCSRFs(self.sensitive_requests, self.alice1_requests,
self.bob_requests, self.unauth_requests)

def log(self):
    print("collected sensitive url ", self.collected_sensitive_requests)
    print("collected total requests", self.collected_total_request)
    print("size of the arrays: alice1, bob, alice2, unauth")
    print(len(self.sensitive_requests), "    ", len(self.bob_requests), "
```

Abstract

```
", len(self.alice1_requests),
          "           ", len(self.unauth_requests))


if __name__ == "__main__":
    login_url = input("enter the login page of the website:")
    domain = login_url.replace("https://", "")
    domain = domain.replace("/login", "")

    mitch = Py_Mitch("firefox", login_url, domain)
    n = int(input("enter numbers of urls you wants to search in : "))
    for i in range(n):
        mitch.search_urls.append(input())
    print("mitch created")

    print("if you logged in print enter: ")
    input()
    mitch.logged_in_Alice()
    mitch.finished_Alice1()
    mitch.logged_out_Alice1()
    mitch.logged_in_bob()
    mitch.logged_out_bob()
    mitch.logged_in_Alice2()
    mitch.logged_out_Alice2()

    print(mitch.collected_total_request, "    ",
mitch.collected_sensitive_requests)
    mitch.make_conclusion()
    # print("mitch main sensitive requests: ", mitch.main_sensitive_req)
    print("a complete log is in : ")
    mitch.log()

    mitch.driver.close()
```

پیوست ب- کدهای محاسبه ساخت مدل یادگیری ماشین

جدول ب- ۱: افزودن کتابخانه‌های مورد نیاز.

```
import csv
import numpy as np
import joblib
from sklearn.ensemble import RandomForestClassifier
from sklearn_porter import Porter
```

جدول ب- ۲ : خواندن داده‌ها از فایل.

```
def reading_data(filename):
    x = []
    y = []
    column_names = []
    with open(filename) as csv_file:
        csv_reader = csv.reader(csv_file, delimiter=',')
        line_count = 0
        for row in csv_reader:
            if line_count == 0:
                column_names = [a for a in row]
                line_count += 1
                # print(column_names)
            else:
                y.append(row[1])
                x.append([float(row[i]) for i in range(len(row)) if i != 1])
                line_count += 1
    return column_names, np.array(x), np.array(y)
```

جدول ب- ۳: ساخت مدل جدید یا بارگزاری مدل ساخته شده.

```
def create_model():
    column_names_temp, x, y = reading_data("dataset/features_matrix.csv")
    column_names_temp.remove("flag")

    # remove first column
    column_names_temp.remove("reqId")
    x = np.delete(x, obj=0, axis=1)

    model = RandomForestClassifier(500)
```

```
    model.fit(x, y)
    return model

def load_or_create_model():
    # check the exisitng of the model in the direstory or create that
    model = None
    model = joblib.load('estimator.pkl')
    if model is not None:
        return model
    else:
        return create_model()
```

جدول ب- ۴: تبدیل مدل به فایل js

```
def port_to_js(model, filename):
    porter = Porter(model, language='js')
    output = porter.export(embed_data=True)
    file = open(filename, "w")
    file.write(output)
    file.close()
```

جدول ب- ۵: ساخت مدل مقاوم نسبت به حمله adversarial.

```
def encode_to_one_hot(arr):
    from numpy import argmax
    from sklearn.preprocessing import LabelEncoder
    from sklearn.preprocessing import OneHotEncoder
    # integer encode
    label_encoder = LabelEncoder()
    integer_encoded = label_encoder.fit_transform(arr)
    print(integer_encoded)
    # binary encode
    onehot_encoder = OneHotEncoder(sparse=False)
    integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
    onehot_encoded = onehot_encoder.fit_transform(integer_encoded)
    print(onehot_encoded)
    # invert first example
    inverted = label_encoder.inverse_transform([argmax(onehot_encoded[0,
:])])
    print(inverted)
    return onehot_encoded

# using ch2 to extract 45 features
from sklearn.feature_selection import chi2
```

Abstract

```
# reading files
column_names_temp, x_temp, y_temp =
reading_data("dataset/features_matrix.csv")
column_names_temp.remove("flag")
# column_names_temp.remove("reqId")
# remove first column
# x_temp = np.delete(x_temp, obj=0, axis=1)
ind = column_names_temp.index("changeInParams")
column_names_temp.remove("changeInParams")
x_temp = np.delete(x_temp, obj=ind, axis=1)

ind = column_names_temp.index("passwordInPath")
column_names_temp.remove("passwordInPath")
x_temp = np.delete(x_temp, obj=ind, axis=1)

ind = column_names_temp.index("payInPath")
column_names_temp.remove("payInPath")
x_temp = np.delete(x_temp, obj=ind, axis=1)

ind = column_names_temp.index("viewInParams")
column_names_temp.remove("viewInParams")
x_temp = np.delete(x_temp, obj=ind, axis=1)


# we want maximum of chi_res[0] and minimum of chi_res[1]
chi_res = chi2(x_temp, y_temp)
index_min = np.argsort(chi_res[1])
# find removing indeces
removing = index_min[len(index_min)-4:]
for i in range(len(removing)):
    column_names_temp.remove(column_names_temp[removing[i]])
    x_temp = np.delete(x_temp, obj=i, axis=1)



"""error ---> we should use 10 fold nested cross validation with 10
percentage for test-----------------------------------------------------
https://machinelearningmastery.com/nested-cross-validation-for-machine-
learning-with-python/
"""
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.1)
# y_train = y_train.reshape((-1, 1))
# y_test = y_test.reshape((-1, 1))

# save the start time
start_time = time.time()

from sklearn.ensemble import RandomForestClassifier
# create the model
model = RandomForestClassifier(n_estimators=500)



from art.estimators.classification.scikitlearn import
ScikitlearnRandomForestClassifier
```

```
# Create the ART classifier
classifier = ScikitlearnRandomForestClassifier(model=model)

onehot_encoded = encode_to_one_hot(y_train)

# twos = np.repeat(2, len(y_train))
# combined = np.vstack((y_train, twos)).T

# Train the ART classifier
classifier.fit(x_train, onehot_encoded)

# Evaluate the ART classifier on benign test examples
predictions = classifier.predict(x_test)

test_encoded = encode_to_one_hot(y_test)
accuracy = np.sum(np.argmax(predictions, axis=1) == np.argmax(test_encoded,
axis=1)) / len(y_test)
print("Accuracy on benign test examples: {}%".format(accuracy * 100))

#save the model
classifier.save("adversial_rf_model_9145", "./")
```

جدول ب- ۶: ساختن وکتورهای حمله.

```
# create an attack
from art.attacks.evasion import BoundaryAttack
attack = BoundaryAttack(classifier)
attack_data = attack.generate(x_test, test_encoded)
np.savetxt("attackVector632.csv", attack_data, delimiter=",")

# now check the attack data on the robust model
robust_predict = classifier.predict(attack_data)

# create a simple classifier
simple_model = RandomForestClassifier(n_estimators=500).fit(x_train, y_train)
# check prediction of attack data in simple model
simple_predict = simple_model.predict(attack_data)

from numpy import argmax
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
label_encoder.fit(y_test)
t = 0
for i in range(len(simple_predict)):
    inverted = label_encoder.inverse_transform([argmax(robust_predict[i,
:])])
    if inverted == y_test[i]:
        t += 1

f = 0
```

```
for i in range(len(simple_predict)):
    if simple_predict[i] == y_test[i]:
        f += 1
```

جدول ب- ۷: تبدیل مدل مقاوم به خروجی js

```
# load the model again
import pickle
loaded_model = pickle.load(open("adversial_rf_model_9145.pickle", 'rb'))

from sklearn_porter import Porter
porter = Porter(loaded_model, language='js')
output = porter.export(embed_data=True)
file = open("robustclassifier.js", "w")
file.write(output)
# Out[6]: 24092671
file.close()
```

پیوست پ- کدهای تشخیص CSRF

جدول پ- ۱: افزودن توابع مورد نیاز.

```
import json
import datetime
from json_schema_infer import *
```

جدول پ- ۲: محاسبه ویژگی ها از روی داده ها و مقایسه درخواست‌ها

```
def isSameReq(base, test):
    if base['url'] != test['url']:
        return False
    for k in base['params'].keys():
        if not (k in test['params']):
            return False

    for p in test['params'].keys():
        if not (p in base['params']):
            return False
    return True
```

Abstract

```
# remove the type in here .....................????????????????//// check
this later---> object is a dict in python
def isDic(v):
    return isinstance(v, dict) and (not (v is None)) and (not isinstance(v,
list)) and (not isinstance(v, datetime.date))
    # return type(v) == 'object' and (not (v is None)) and (not isinstance(v,
list)) and (not isinstance(v, datetime.date))
    # return (not (v is None)) and (not isinstance(v, list)) and (not
isinstance(v, datetime.date))


def isSameSchema(sA, sB):
    for k in sA.keys():
        if not (k in sB):
            return False
        else:
            if isDic(sA[k]) and isDic(sB[k]) and (not isSameSchema(sA[k],
sB[k])):
                return False
            elif sA[k] != sB[k]:
                return False

    return True


def hasSameJSONSchema(a, b):
    sA = getSchema(a)
    sB = getSchema(b)

    ret = isSameSchema(sA, sB)
    return ret


def isHTML(s):
    if "</html>" in s['body'].lower():
        return True
    if 'Content-Type' in s['headers'] and "text/html" in
s['headers']['Content-Type']:
        return True
    return False


def isJson(s):
    try:
        j = json.dump(s['body'])
        if isinstance(j, int):
            return False
    except:
        return False

    return True
```

Abstract

```
def compare_requests(rA, rB):
    result = {
        'url': rA['url'],
        'params': rA['params'],
        'overall': 'same',
        'method': {},
        'status': {},
        'body': {'ans': 'same'}
    }
    statusA = rA['response']['status']
    statusB = rB['response']['status']

    if statusA == statusB:
        result['status']['ans'] = 'same'
    else:
        result['status']['ans'] = 'different'
        result['overall'] = 'different'

    result['status']['valueA'] = statusA
    result['status']['valueB'] = statusB

    # checking the body type
    if isHTML(rA['response']):
        result['body']['typeA'] = 'html'
    elif isJson(rA['response']):
        result['body']['typeA'] = 'json'
    else:
        result['body']['typeA'] = 'plaintext'

    if isHTML(rB['response']):
        result['body']['typeB'] = 'html'
    elif isJson(rB['response']):
        result['body']['typeB'] = 'json'
    else:
        result['body']['typeB'] = 'plaintext'

    min_length = min(len(rA['response']['body']),
len(rB['response']['body']))
    max_length = max(len(rA['response']['body']),
len(rB['response']['body']))

    result['body']['ratio'] = (min_length + 1) / (1.0 * max_length + 1)

    if result['body']['typeA'] == 'JSON' and result['body']['typeB'] ==
'JSON':
        json_a = json.dump(rA['response']['body'])
        json_b = json.dump(rB['response']['body'])
        if hasSameJSONSchema(json_a, json_b):
            result['body']['ans'] = 'same'
        else:
            result['body']['ans'] = 'different'
            result['overall'] = 'different'
    elif result['body']['typeA'] == 'html' and result['body']['typeB'] ==
'html':
        if result['body']['ratio'] < 0.99:
```

Abstract

```
            result['body']['ans'] = 'different'
            result['overall'] = 'different'

    elif result['body']['typeA'] == 'plaintext' and result['body']['typeB']
== 'plaintext':
        if rA['response']['body'] != rB['response']['body']:
            result['body']['ans'] = 'different'
            result['overall'] = 'different'

    else:
        if result['body']['typeA'] != result['body']['typeB']:
            result['body']['ans'] = 'different'
            result['overall'] = 'different'

    result['body']['valueA'] = rA['response']['body']
    result['body']['valueB'] = rB['response']['body']

    print("result in guess CSRF is : ", result)
    return result


def compare_sensitive_requests(runA, runB):
    results = []
    for rA in runA:
        found = False
        for rB in runB:
            if isSameReq(rA, rB):
                found = True
                # compare two equal request
                results.append(compare_requests(rA, rB))
        if not found:
            print("couldn't find request: ", rA['url'])

    return results

def isSameEndPoint(base, test):
    if base['url'] != test['url']:
        return False

    for k in base['params'].keys():
        if not (k in test['params']):
            return False

    for p in test['params'].keys():
        if not (p in base['params']):
            return False

    return True


def findRequest(needle, haystack):
    for r in haystack:
        if isSameEndPoint(needle, r):
            return r
```

```
    # print("!!!!no matching endpoint found for ", needle['url'])
    return False

def guessCSRFs(alice, alice1, bob, unauth):
    print("comparing traces...")
    alice_vs_unauth = compare_sensitive_requests(alice, unauth)
    alice_vs_alice1 = compare_sensitive_requests(alice, alice1)
    alice_vs_bob = compare_sensitive_requests(alice, bob)
    candidates = []
    print("comparison analysis ...")
    print("confirming sensitivity ... ")

    for r in alice_vs_unauth:
        print("checking ... ", r['url'])
        if r['overall'] == 'different':
            print("candidate added")
            candidates.append(r)

    resulting_candidates = []
    print("confirming reachability ... ")
    for c in candidates:
        print("checking :", c['url'])
        r_avb = findRequest(c, alice_vs_bob)
        r_ava1 = findRequest(c, alice_vs_alice1)

        if r_avb['overall'] == ' different' and r_ava1['overall'] ==
'different':
            continue
        resulting_candidates.append(c)

    return candidates, resulting_candidates
```

جدول پ- ۳: توابع مورد نیاز برای پردازش درخواست های json

```
def typeArray(val):
    r = []
    for v in val:
        v_type = type(v)
        if not (type in r):
            r.append(v_type)
    return {'type': r}


def typeValue(val):
    if isinstance(val, list):
        return {'type': 'array', 'items': typeArray(val)}
    # check if it is an object or not!!!!!
    if isinstance(val, dict):
        properties = getProperties(val)
        return {'type': 'object', 'properties': properties, 'required':
```

Abstract

```
properties.keys()}

    if isinstance(val, int):
        return {'type': 'integer'}

    return {'type': type(val)}

def getProperties(j):
    print("!!!!!!!!!!!!!!!!!! this instance is not checked yet")
    k = j.keys()
    for name in k:
        j[name] = typeValue(j[name])
    return j


def getSchema(json_object):
    schema = {}

    # print("!!!!!!!!!!!!!!!!!!! getSchema in json_schema_infer is not
completed.....")

    schema['$schema'] = 'http://json-schema.org/schema#'
    schema['title'] = 'JSON inferred schema'
    schema['description'] = 'JSON inferred schema'
    schema['type'] = 'object'
    try:
        schema['properties'] = getProperties(json_object)
        schema['required'] = schema['properties'].keys()
    except:
        schema['properties'] = {}
        schema['required'] = schema['properties'].keys()

    return schema
```

پیوست ت- کدهای محاسبه حساسیت.

شکل ت- ۱: توابع مورد نیاز برای ساخت داده ورودی مدل از روی درخواست.

```
# sensitivity.py
import numpy as np

"""why does it use all 52 features not 45 ones of them??????????????"""
def countParams(req):
    return len(req['params'])

def countBools(req):
    numberBools = 0
    for p in req['params'].keys():
        # print(req['params'][p])
        if req['params'][p] == 'True' or req['params'][p] == 'False' or
req['params'][p] == '1' or req['params'][p] == '0':
            numberBools += 1
    # print("number of bools are : ", numberBools)
    return numberBools


import re
def countIds(req):
    numOfIds = 0
    prog = re.compile('^[0-9]{14}|[0-9\-a-fA-F]{20,}$')
    for p in req['params'].keys():
        if req['params'][p] is not None:
            if prog.match(req['params'][p][0]) is not None:
                numOfIds += 1
    return numOfIds


def countBlobs(req):
    numOfBlobs = 0
    prog = re.compile('^[^\s]{20,}$')
    for p in req['params'].keys():
        if req['params'][p]:
            if prog.match(req['params'][p][0]) != None:
                numOfBlobs += 1
    return numOfBlobs

# length of the params keys and values
def getReqLen(req):
    l = 0
    for p in req['params'].keys():
        if req['params'][p]:
            l = l + len(p) + len(req['params'][p])
    return l

# changed the concept of this method .... used new implementation
```

Abstract

```
def isInPath(req, k):
    if k.lower() in req['url'].lower():
        return 1
    else:
        return 0

# changed the concept of this method .... used new implementation
def isInParam(req, k):
    for p in req['params'].keys():
        if k.lower() in p.lower():
            return 1
    return 0
```

شکل ت- ۲: تابع اصلی محاسبه حساسیت.

```
def isSensitive(req, classifier):
    fetureVector = []

    if req['method'].upper() == "PUT" or req['method'].upper == "DELETE":
        return True
    if req['method'].upper() == "OPTIONS":
        return False
    # numberOfParams
    fetureVector.append(countParams(req))
    # numberOfBools
    fetureVector.append(countBools(req))
    # numberOfIds
    fetureVector.append(countIds(req))
    # numberOfBlobs
    fetureVector.append(countBlobs(req))
    # reqLen
    fetureVector.append(getReqLen(req))

    keywords = ['create', 'add', 'set', 'delete', 'update', 'remove',
                'friend', 'setting', 'password', 'token', 'change', 'action',
                'pay', 'login', 'logout', 'post', 'comment', 'follow',
'subscribe', 'sign', 'view']
    # ??????????????????????????.....................................
    for k in keywords:
        fetureVector.append(isInPath(req, k))
        fetureVector.append(isInParam(req, k))

    methods = ['PUT', 'DELETE', 'POST', 'GET', 'OPTIONS']
    for m in methods:
        if m == req['method'].upper():
            fetureVector.append(1)
        else:
            fetureVector.append(0)

    # now run javascript model here
    sensitivity = classifier.predict([np.array(fetureVector)])
    return sensitivity[0]
```

Abstract

With the ever-expanding web space, the need for a tool to identify vulnerabilities automatically and quickly has increased. On the other hand, artificial intelligence and machine learning and deep learning algorithms have entered all areas of technology and have improved the performance of existing systems. Therefore, in this report, an attempt has been made to implement and improve a tool to detect CSRF vulnerabilities using machine learning methods.

In this way of detection, the main core of the system is based on Machine learning that can classify the HTTP requests into two groups, Sensitive and unsensitive.Then we consider some heuristic rules to recognize the vulnerability in the website. System is looking the whole website as a black-box, so It is not depend on the programming language.

Key Words:

Vulnerability, Machine learning, deep learning, Black-Box system.

Amirkabir University of Technology
(Tehran Polytechnic)

Department of Computer Engineering

Final Project For BSc

Design and implementation of web application analysis software to detect cross-site-scripting and injection vulnerabilities

By
Tahera Fahimi

Supervisor
Dr.Hamid Reza Shahriari

Jun.2021

References

[1] S. Calzavara, M. Conti, R. Focardi, A. Rabitti and G. Tolomei, "Machine Learning for Web Vulnerability Detection: The Case of Cross-Site Request Forgery," in IEEE Security & Privacy, vol. 18, no. 3, pp. 8-16, May-June 2020, doi: 10.1109/MSEC.2019.2961649.

[2] PwnFunction, "Cross-Site Request Forgery (CSRF) Explained," , 2019.

[3] P. team, "SQL Injection," PortSwigger, 2021. [Online]. Available: https://portswigger.net/web-security/sql-injection. [Accessed 2021].

[4] P. team, "cross site scripting," web security academy, 2021. [Online]. Available: https://portswigger.net/web-security/cross-site-scripting. [Accessed 2021].

[5] D. Caissy, "Owasp Top 10 - 2017," July 2017. [Online]. Available: https://owasp.org/www-pdf-archive//OWASP_LA_New_OWASP_Top_10_David_Caissy_2017_07.pdf. [Accessed July 2017].

[6] S. Calzavara, M. Conti, R. Focardi, A. Rabitti and G. Tolomei, "Mitch: A Machine Learning Approach to the Black-Box Detection of CSRF Vulnerabilities," 2019 IEEE European Symposium on Security and Privacy (EuroS&P), 2019, pp. 528-543, doi: 10.1109/EuroSP.2019.00045.

[7] G. P. Soheil Khodayari, "JAW: Studying Client-side CSRF with Hybrid Property Graphs," CISPA Helmholtz Center, 2021.

[8] R. Shah, "Introduction to k-Nearest Neighbors (kNN) Algorithm," ai.plainenglish, 2021. [Online]. Available: https://ai.plainenglish.io/introduction-to-k-nearest-neighbors-knn-algorithm-e8617a448fa8.

[9] R. Bhatia, "analyticsindiamag," 19 09 2017. [Online]. Available: https://analyticsindiamag.com/top-6-regression-algorithms-used-data-mining-applications-industry/.

[10] H. Ampadu, "Random Forest understanding," AI Pool, 01 04 2021. [Online]. Available: https://ai-pool.com/a/s/random-forests-understanding.

[11] Giancarlo Pellegrino, Martin Johns, Simon Koch, Michael Backes, Christian Rossow, "Deemon: Detecting CSRF with Dynamic Analysis and Property Graph," *arXiv*, 2017. [Online]. Available: https://arxiv.org/pdf/1708.08786.

[12] O. C. S. Team, "Cross-Site Request Forgery Prevention Cheat Sheet," GitHub, [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html.

[13] O. team, "Injection," Owasp, 2017. [Online]. Available: https://owasp.org/www-project-top-ten/2017/A1_2017-Injection. [Accessed 2017].

[14] KirstenS, "XSS," Owasp , [Online]. Available: https://owasp.org/www-community/attacks/xss. [Accessed 25 Apr 2021].

[15] N. P. S. H. R. D. P. A. J. C. Ian Goodfellow, "adversarial example research," 24 Febrary 2017. [Online]. Available: https://openai.com/blog/adversarial-example-research/.

[16] W. Keeling, "Selenium-Wire," PyPi, 2 Aug 2018. [Online]. Available: https://pypi.org/project/selenium-wire/. [Accessed 2021 Jun 2021].

[17] Acunetix, "the Acunetix Web Application," Acunetix, USA, 2019.

[18] ilmoi, "will my Machine Learning system be attacked?," towards data science, 15 07 2019. [Online]. Available: https://towardsdatascience.com/will-my-machine-learning-be-attacked-6295707625d8.

[19] N. P. S. H. R. D. ,. P. A. J. C. Ian Goodfellow, "Attacking Machine Learning with adversarial examples," 24 2 2017. [Online]. Available: https://openai.com/blog/adversarial-example-research/. [Accessed 24 2 2017].

[20] S. B. M. K. L. B. Mahmood Sharif, "Accessorize to a Crime: Real and Stealthy Attacks on state-Of-Art Face Recognition," vol. 15, p. 10, 2016.

[21] Jonasguan, "cleverhans-lab," cleverhans, 11 2 2021. [Online]. Available: https://github.com/cleverhans-lab/cleverhans. [Accessed 21 1 2016].