ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ

Факултет Компютърни системи и технологии

Катедра Компютърни системи

маг. инж. Любен Асенов Николов

МЕТОДИ И СРЕДСТВА ЗА ОСИГУРЯВАНЕ НА ЗАЩИТА НА УЕБ ПРИЛОЖЕНИЯ

А В Т О Р Е Ф Е Р А Т

на дисертация за придобиване на образователна и научна степен
"ДОКТОР"

Област: 5. Технически науки
Професионално направление: 5.3 „Комуникационна и компютърна техника“
Научна специалност: „Компютърни системи, комплекси и мрежи“

Научни ръководители:

Доц. д-р инж. Аделина Алексиева-Петрова

СОФИЯ, 2026 г.

Дисертационният труд е обсъден и насочен за защита от Катедрения съвет на катедра „Компютърни системи“ към Факултет Компютърни системи и технологии на ТУ-София на редовно заседание, проведено на 20.10.2025 г.

Публичната защита на дисертационния труд ще се състои на 27.01.2026 г. от 15:00 часа в Конферентната зала на БИЦ на Технически университет – София на открито заседание на научното жури, определено със заповед № ОЖ-5.3-59/30.10.2025г. на Ректора на ТУ-София в състав:

1. Проф.Д-р Милена Кирилова Лазарова-Мицева – председател
2. Проф.Д-р Георги Илинчев Попов – научен секретар
3. Проф.Д-р Нина Василевна Синягина
4. Проф.Д-р Атанас Велков Атанасов
5. Проф.Д-р Мариана Евстатиева Горанова

Рецензенти:

1. Проф.Д-р Милена Кирилова Лазарова-Мицева
2. Проф.Д-р Нина Василевна Синягина

Материалите по защитата са на разположение на интересуващите се в канцеларията на Факултет Компютърни системи и технологии на ТУ-София, блок №1, кабинет № 1443-А.

Дисертантът е редовен докторант към катедра „Компютърни системи“ на Факултет Компютърни системи и технологии. Изследванията по дисертационната разработка са направени от автора, като някои от тях са подкрепени от научноизследователски проекти.

Автор: маг. инж. Любен Николов

Заглавие: Методи и средства за осигуряване на защита на уеб приложения

Тираж: 30 броя

Отпечатано в ИПК на Технически университет – София

I. ОБЩА ХАРАКТЕРИСТИКА НА ДИСЕРТАЦИОННИЯ ТРУД

Актуалност на проблема

Преминаването към дистанционна работа от вкъщи се превръща в „новата норма“ по време на пандемията. От началото на пандемията възниквалите киберзаплахи се увеличават, което довежда с до 600% увеличение на киберпрестъпността. Киберпрестъпниците използват различни канали за стартирането на хакерски атаки, вариращи от традиционни измами с атаки тип фишинг по имейл до сложни тактики като вмъкване на злонамерен скрипт в уеб страници Cross Site Scripting (XSS) и SQL инжектиране, целящи извличането на чувствителни данни.

В настоящия момент рисковете за киберсигурността са по-критични от всякога. Броят на жертвите, поради "източване" на данни следствие на хакерски атаки се увеличава с 210% през третото тримесечие на 2022 г. в сравнение с второто тримесечие на същата година. Уведомленията за нарушаване на данните ескалират от 19 през 2019 г. до 617 през 2022 г.

С развитието на информационните технологии и нарастващата им роля в съвременния дигитален свят, сигурността на софтуерните системи става все по-важна. Ускореното внедряване на нови технологии като облачни услуги, интернет на нещата (IoT), машинно обучение и изкуствен интелект поставя под заплаха критичните информационни инфраструктури и личните данни. Защитата на информацията и сигурността на уеб приложенията изискват внедряването на подходи и инструменти, които могат да идентифицират и смекчат заплахите по време на всички етапи на разработката и внедряването на софтуер.

Докато една от най-слабите точки в позицията на киберсигурността в една организация е крайният потребител, което е породено от липса на обучения за осведоменост в киберсигурността, друг фактор възниква в началото на цикъла от разработка на код, когато приложенията се разработват още в най-ранния цикъл от разработване на софтуер. А именно, лошо проектирани приложения без въведени механизми за сигурност, на по-късен етап оставят задни врати или пролуки, от които злонамерените потребители могат да се възползват и да експлоатират.

Моделирането на заплахите е методология, която помага да се идентифицират, анализират и оценяват потенциалните рискове в системата, още по време на етапите на софтуерно проектиране. Въпреки значителните му

предимства, този процес често е ръчен, което го прави трудоемък и податлив на човешки грешки. От друга страна, SAST и DAST предлагат систематичен подход към идентифицирането на уязвимости. SAST анализира изходния код на системата, за да открие слабости в структурата или логиката на кода, докато DAST симулира реални атаки върху работещото приложение, за да открие недостатъци в изпълнението. С увеличаващата се сложност на софтуерните системи и киберзаплахите е критично важно да се разработят автоматизирани решения, които интегрират моделирането на заплахите със статично и динамично тестване. Такива решения могат да намалят времето за откриване и валидиране на уязвимости, като същевременно увеличат точността и намалят зависимостта от човешки фактор.

Цел на дисертационния труд

Целта на дисертационния труд е да изследва практиките и методите, които се използват от злонамерени потребители в интернет за генерирането на уеб хакерски атаки. Настоящият труд има за цел да предложи и изследва подход за автоматизирано моделиране на заплахите, което включва интеграция на инструменти, заедно с ефективно внедряване на статични и динамични тестове на уязвимости за подобряване на сигурността в жизнения цикъл на разработка на софтуер за уеб приложения. Това позволява постигането на оптимална сигурност и надеждност на приложенията в съвременните динамични и разпределени среди.

Обект на изследване

Обект на изследване на дисертационния труд са подходи, методи и алгоритми за моделиране на заплахите, както и процесите за статично (SAST) и динамично (DAST) тестване на уеб приложения с цел идентифициране, оценка и минимизиране на рисковете от уязвимости. Изследването обхваща интеграцията на тези методи в жизнения цикъл на разработка на софтуер (SDLC) и тяхното влияние върху повишаването на сигурността и ефективността на софтуерните системи.

Предмет на изследване

Предмет на изследването са скоростта и ефективността на подходите за идентифициране на заплахи, както и възможностите за усъвършенстване на сигурността на уеб приложения чрез прилагане на методи за статичен и динамичен анализ на уязвимости. Акцентът е поставен върху усъвършенстването на тези техники за повишаване на точността,

автоматизацията и намаляване на времето за реакция при откриване на проблеми със сигурността.

Основна изследователска хипотеза

Основната изследователска хипотеза на изследването в дисертационния труд е: сигурността на уеб приложенията може да се повиши чрез интеграция на методи за моделиране на заплахите, статично (SAST) и динамично (DAST) тестване, с цел откриване и минимизиране на уязвимости на различни етапи от жизнения цикъл на разработка на софтуер (SDLC), както и значително намаляване на времето за целия процес – от началото на разработването до финалната фаза на внедряване на софтуер.

Задачи

За постигане на тази цел се поставят следните изследователски задачи:

1. Да бъдат проучени и изследвани подходи, методи и алгоритми за сигурност на данни, които се съдържат в уеб приложения, както и естеството на атаките и тяхното въздействие върху сигурността.

2. Да се изследва възможността за експлоатация на уеб приложения чрез тестване от софтуерни решения за киберсигурност, които се използват за оценка на риска и уязвимостта в уеб приложенията.

3. Да се изследват, разработят и интегрират методи за моделиране на заплахите, статично (SAST) и динамично (DAST) тестване на уязвимости, с цел подобряване на сигурността в жизнения цикъл на разработка на софтуер (SDLC) за уеб приложения.

4. Да се предложат методи и решения за сигурност, които имат за цел да осигурят интегритет и защита на данните срещу различни атаки.

Практическа приложимост

Предложените в дисертационния труд методи и подходи за моделиране на заплахи и статично (SAST) и динамично (DAST) тестване за уязвимости намират пряка практическа приложимост в процеса на разработка на уеб приложения и софтуерни системи, следващи принципите на DevSecOps. Чрез автоматизация на тези процеси и интеграцията им в жизнения цикъл на разработка на софтуер (SDLC) се осигурява ранно откриване и отстраняване на уязвимости, което води до значително намаляване на времето и разходите за тестване и повишаване на надеждността на крайния продукт.

Практическото приложение на предложената автоматизирана архитектура, използваща инструменти като Irius Risk и OWASP ZAP, позволява реално

внедряване в производствени DevSecOps тръбопроводи и непрекъсната интеграция. Това дава възможност на организациите да постигнат по-високо ниво на сигурност, по-бърз процес на разработка и намален риск от човешки грешки, като по този начин се повишава общата ефективност и устойчивост на софтуерните решения в динамични и разпределени среди.

Апробация

Апробацията на резултатите, получени в рамките на дисертационния труд е осъществена в серия от международни научни конференции и чрез публикация в международно научно списание. Една от публикациите е публикувана в международното научно списание WSEAS през 2025 г. Една от публикациите е докладвана и представена на 11-тата международна конференция „Computer Science 2023“, проведена в Созопол, България през 2023г. Друга част от публикациите са докладвани на международната конференция по приложна физика, симулация и изчислителни науки (International Conference on Applied Physics, Simulation and Computing), проведена в Рим, Италия през 2024 г., както и на международната конференция по софтуер, телекомуникации и компютърни мрежи(SoftCOM) през 2024 г. в Бол, Хърватска.

Публикации

В резултат на проведеното дисертационно изследване са публикувани общо четири научни статии, отразяващи основните идеи, резултати и приноси на разработката. Една от публикациите е отпечатана в международно научно списание, а останалите три са представени на международни научни конференции и публикувани в IEEE. Всички 4 статии са изготвени на английски език и са в съавторство с научния ръководител на докторанта.

Три от публикациите са индексирани в международната база данни Scopus, което потвърждава тяхната научна значимост и видимост в международното изследователско пространство. Две от тях са вече цитирани общо седем пъти от чуждестранни автори, включително в три статии, индексирани в Scopus.

Структура и обем на дисертационния труд

Дисертационният труд е с общ обем от 181 страници и е структуриран така, че логично и последователно да представя цялостния изследователски процес. Съдържа уводна част, в която се обосновава актуалността на темата и се формулират основните цели и задачи на изследването, последвана от четири тематични глави, насочени към решаването на поставените научни проблеми.

В края на дисертационния труд са включени списък на основните приноси, публикациите по темата на дисертацията, сведения за участия в научноизследователски проекти, използваните литературни източници, както и речник на термините, списък на съкращенията, и пет приложения, съдържащи допълнителни резултати и материали, подпомагащи изложението.

В дисертацията са цитирани 114 литературни източника, в това число и 5 интернет ресурса. Дисертационният труд съдържа 47 фигури и 15 таблици, като номерацията им в автореферата напълно съответства на тази в основния труд.

II. СЪДЪРЖАНИЕ НА ДИСЕРТАЦИОННИЯ ТРУД

ГЛАВА 1. ОБЗОР НА МЕТОДИ И СРЕДСТВА ЗА ОСИГУРЯВАНЕ НА ЗАЩИТА НА УЕБ ПРИЛОЖЕНИЯ

Видове сканиране за сигурност

В днешния дигитален свят, където софтуерът е основа на почти всяка индустрия, сигурността на приложенията е от критично значение. Уязвимостите в софтуера могат да предизвикат кибератаки, които застрашават поверителността, интегритета и наличността на данните. Именно затова техниките за тестване на сигурността като статичното (SAST) и динамичното (DAST) сканиране за уязвимости са от съществено значение. Те помагат на разработчиците и екипите за сигурност да идентифицират и коригират потенциални рискове в различни етапи от жизнения цикъл на софтуера, осигурявайки по-надеждна защита срещу злонамерени атаки.

Статичното тестване на сигурността на приложенията (SAST) анализира изходния код на приложението по време на разработката на софтуер, за да открие уязвимости още в ранните етапи на жизнения цикъл при разработването на софтуер. От друга страна, динамичното тестване на сигурността на приложенията (DAST) се провежда върху работещо приложение, симулирайки атака от гледна точка на външен нападател, без достъп до изходния код. Докато SAST е ефективно решение за откриване на грешки в кода, DAST идентифицира уязвимости, които се проявяват само по време на изпълнение на работещо уеб приложение.

Динамично сканиране за сигурност (DAST)

Сигурността на уеб приложенията е от съществено значение и изисква редовно тестване. Съществуват множество техники и инструменти за тестване на сигурността, като OWASP (Open Web Application Security Project) предлага набор от насоки и инструменти, които помагат за откриването и коригирането на уязвимости. Основните методи за тестване включват Black Box (DAST) и White Box (SAST) тестване, които помагат на разработчиците и експертите по сигурност да идентифицират и поправят уязвимости в кода на приложенията.

Слабите контроли за сигурност на информационните активи представляват значителен проблем, създавайки уязвимости, които често се експлоатират от хакери. Изследванията показват, че приблизително 90% от интернет престъпленията включват уеб приложения, като атаките чрез инжектиране на база данни съставляват 47.06% от всички атаки. Други често срещани

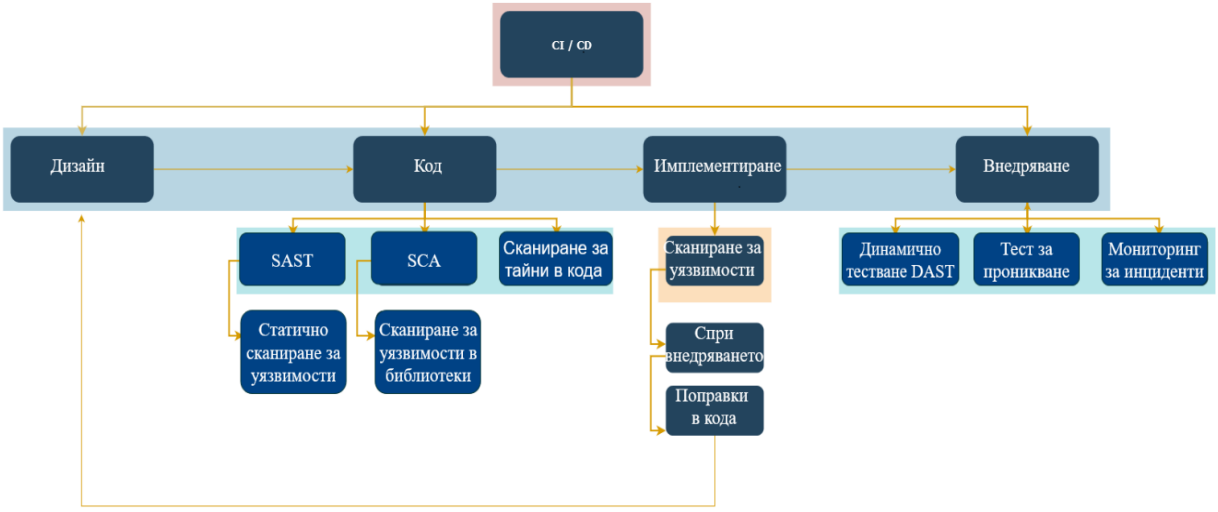
уязвимости, като SQL Injection и Cross-Site Scripting (XSS), допринасят за 35.33% от пробивите в сигурността, често в резултат на грешки в програмирането и неправилна конфигурация.

Инструменти като OWASP ZAP са от съществено значение в процесите на тестване за проникване, предлагайки функции като прихващане на прокси сървъри, сканиране на портове, анализ на параметри и сравнения на сесии. Тези инструменти включват също усъвършенствани възможности като обработка на CSRF идентификатор и динамични SSL сертификати, подобрявайки способността им да идентифицират и смекчават уязвимостите. OWASP ZAP е високо оценен за своята ефективност при откриване на уязвимости, особено в областта на сигурността на уеб приложенията.

DAST проверява сигурността на работещото приложение, като симулира атаки в реална среда. Инструменти като OWASP ZAP и Burp Suite се използват широко за тестване на уеб приложения. Динамичното тестване за уязвимости се използва в по-късните етапи на жизнения цикъл за разработка на софтуер, най-често по време на тестване или преди внедряване в продукционна среда.

Модели DevOps и DevSecOps

DevOps включва използване на итеративна разработка на софтуер, автоматизация и използване на програмируеми и декларативни инфраструктури. Проблемите със сигурността в DevOps често произлизат от конфликти между различните цели на разработчиците и екипите за киберсигурност. Основната характеристика на DevOps е автоматизацията на много процеси за тестване и интеграция на софтуер, което позволява на организациите бързо и безпроблемно да създават нови версии на софтуер.



Фигура 4 SAST и DAST в DevSecOps

Във фигура 4 се наблюдава типичен поток и интеграция в така наречения тръбопровод (pipeline), където процеса за разработка на софтуер започва във фазата на дизайн на кода и през втората фаза код, когато кодът се проверява за уязвимости чрез статично сканиране за сигурност (SAST) и проверка за уязвимости в библиотеките (SCA – Software Composition Analysis), които се използват. В тази фаза екипите на разработчиците на софтуер и екипите по сигурност имат възможността да открият уязвимости, които могат да бъдат открити и разрешени още преди кодът да е компилиран. В крайната стъпка следва процес, където кодът автоматично се компилира и се интегрира в тестова или в работна среда. Това е финалната фаза, където динамичното тестване за сигурност на приложения има възможност да сканира готовия софтуерен продукт за уязвимости (DAST) и на база готовите резултати, екипите за сигурност имат възможност да поправят евентуални уеб заплахи, открити по време на фазата за внедряване на софтуера.

Появата на DevOps методологиите революционизира софтуерното разработване, обединявайки разработката и операциите за постигане на скорост и гъвкавост. Въпреки това, ускорението повдига значителни въпроси относно интегрирането на надеждни мерки за сигурност, което води до появата на DevSecOps като рамка за вграждане на сигурността във всички етапи на разработването на софтуер. DevSecOps измества сигурността "наляво" в жизнения цикъл на софтуерното разработване (SDLC), като гарантира, че уязвимостите се идентифицират и адресират рано, минимизирайки потенциалните рискове и разходи. Изследванията подчертават необходимостта от интегриране на сигурността в непрекъснати разработки чрез автоматизация за откриване на уязвимости в реално време и тестване, както и насърчаването на мислене, ориентирано към сигурност в рамките на екипите за разработка на софтуер.

Моделиране на заплахите

Интегрирането на най-добрите практики за разработване на сигурен софтуер в жизнения цикъл на разработка на софтуер е от съществено значение, за да се гарантира, че софтуерът остава функционален дори при зловредни атаки. Архитектурния анализ на риска и моделирането на заплахи, е ключов елемент в този процес. Моделът на риска служи като количествена рамка за оценка на вероятността и въздействието на заплахите върху активите и компонентите в една архитектура.

Няколко предишни изследвания илюстрират подхода STRIDE за моделиране на заплахи. Например, някои автори представят метод за моделиране на заплахи с цел оценка и справяне с ботнет атаки в сценарии за умен дом в контекста на Интернет на нещата (IoT). Подходът включва идентифициране на потенциални заплахи както на ниво разработка, така и на ниво приложение в умния дом, използвайки техниките за моделиране на заплахи STRIDE и VAST. Освен това, изследването свързва тези идентифицирани заплахи с потенциални ботнет атаки и предлага различни стратегии за разрешаване на всички идентифицирани заплахи.

Редица предходни изследвания подкрепят интеграцията на моделиране на заплахи в DevOps pipeline модели. Например, е представен прототип за непрекъснат анализ и управление на заплахите, който улеснява непрекъснатото моделиране и извличане на заплахи, интегрирайки този процес в конвейера за непрекъсната интеграция в рамките на GitLab. Чрез включване на анализа на заплахите в хранилището на изходния код, управлението на заплахите става непрекъсната задача, позволявайки по-точно наблюдение на напредъка в смекчаването на заплахите.

Изводи и заключения към първа глава

В съвременния дигитален свят, където сигурността на софтуерните системи е критичен аспект от разработката на приложения, интегрирането на методи за тестване на уязвимости като статичен анализ (SAST) и динамичен анализ на сигурността (DAST) е от съществено значение. Проведените изследвания подчертават важността на тези техники в рамките на DevSecOps практиките, които се стремят да въведат сигурността като неразделна част от жизнения цикъл на разработка на софтуера (SDLC).

Проучванията в Глава 1 също така акцентират върху моделирането на заплахите, което представлява методология за идентифициране, анализ и класификация на потенциални рискове в информационните системи. Разгледаните методи, като STRIDE, PASTA и дървета на атаките, които предоставят аналитична рамка за оценка на заплахите, като същевременно подпомагат процеса на проектиране на сигурни архитектури. Въведените инструменти като Microsoft Threat Modeling Tool (MTMT), OWASP Threat Dragon и Pytm осигуряват ефективност и автоматизация на процесите за моделиране, което значително намалява времето и ресурсите, необходими за идентифициране на уязвимости. Освен това, тяхната интеграция в DevSecOps тръбопроводите позволява ранно откриване на потенциални рискове и заплахи,

както и тяхното ефективно смекчаване чрез автоматизирано генериране на тестови случаи за сигурност.

Разгледаните изследвания показват, че DevSecOps методологиите предоставят значителни предимства при управлението на сигурността в софтуерната разработка, като въвеждат принципа на "shift-left" сигурността, позволявайки ранно откриване и адресиране на уязвимости. Въпреки това, предизвикателства като фалшиви положителни резултати (False Positives) при SAST, ограниченото покритие на динамичните тестове при DAST, както и липсата на стандартизирани подходи за моделиране на заплахите остават отворени проблеми, изискващи допълнителни изследвания. Въз основа на направените анализи може да се заключи, че интегрирането на SAST, DAST и моделирането на заплахи в DevSecOps среда значително подобрява сигурността на софтуерните системи, като намалява риска от експлоатация на уязвимости и улеснява процеса на проактивно управление на сигурността. Въпреки това, ефективността на тези техники зависи от автоматизацията на тестовете, културните промени в организациите и внедряването на съвременни технологии като машинно обучение и изкуствен интелект за адаптивно откриване на заплахи. Моделирането на заплахите играе ключова роля в този процес, като осигурява методологична рамка за идентифициране на потенциални заплахи и оценка на рисковете, което позволява по-добро прогнозиране и предотвратяване на потенциални атаки. Включването на методологии като STRIDE, дървета на атаките и PASTA в DevSecOps тръбопроводите може значително да подобри реакцията срещу заплахи чрез автоматизирано генериране на тестови сценарии и симулиране на възможни пътища за компрометиране на системите. Бъдещите насоки за изследване включват разработване на по-прецизни механизми за статичен анализ, подобряване на динамичните тестове, както и автоматизиране на моделирането на заплахи с цел интеграция в CI/CD тръбопроводите за по-ефективно и гъвкаво управление на киберсигурността.

ГЛАВА 2. МОДЕЛИРАНЕ НА ЗАПЛАХИ ВЪЗ ОСНОВА НА РАМКАТА STRIDE ЗА УЕБ ПРИЛОЖЕНИЯ

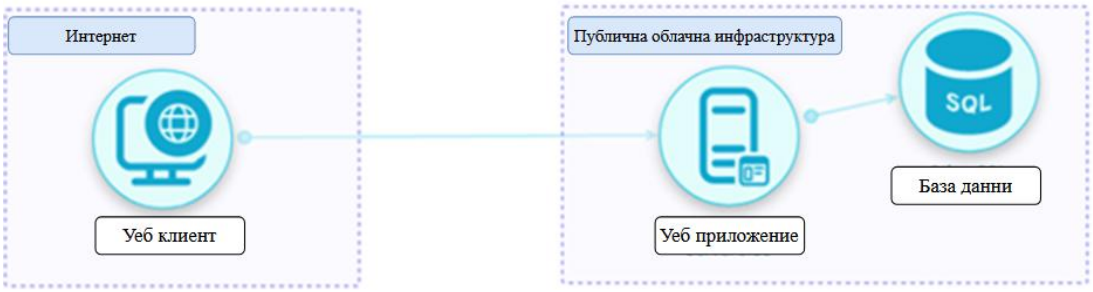
Основната цел на моделирането на заплахи е проактивно да идентифицира, класифицира и опише кибер заплахите чрез рамката STRIDE, с цел повишаване на видимостта на потенциалните атаки, които могат да съществуват в една

сървърна екосистема. Този подход подпомага развитието на устойчивост чрез предвиждане на проблеми в една архитектура, свързани със сигурността.

В дисертацията се разглеждат четири основни инструмента за моделиране на заплахи: Irius Risk, Microsoft Threat Modeling Tool, OWASP Threat Dragon и Trike. Анализът се основава на моделиране на заплахи върху диаграма от високо ниво, която включва уеб клиент, уеб сървър и база данни, свързани в публична инфраструктура. За всеки инструмент са идентифицирани и документирани потенциалните заплахи, произтичащи от архитектурата и комуникационните потоци, както и механизмите за противодействие, които целят минимизиране на риска и повишаване на сигурността на системата.

Решения за моделиране на заплахите

Основната цел на този етап от методологията DevSecOps всъщност е да се хванат всякакви уязвимости по време на етапа на проектиране на приложението, още преди да започне самият процес на разработка. Моделирането на заплахите спомага за идентифициране на потенциални уязвимости в инфраструктурата и в самото приложение, които могат да представляват заплаха за организацията както от техническа, така и от бизнес перспектива в най-ранния етап на SDLC. Разбира се, IriusRisk също предоставя пълен списък с контрамерки за уязвимостите, споменати по-рано. Тези контрамерки могат да бъдат взети под внимание, когато приложенията всъщност се разработват.



Фигура 10 Irius Risk диаграма за моделиране на заплахите

Създадена е диаграма на високо ниво на архитектурата (Фиг. 10), която помогна за проучването на потенциалните заплахи, които могат да съществуват във връзка с изобразения модел. Списъкът с потенциални заплахи и мерки за противовъздействие на уязвимостите са предоставени от IriusRisk и за двата компонента.

Резултати и дискусии

Трите инструмента (Irius Risk, Microsoft Threat Modeling Tool и OWASP Threat Dragon), които са разгледани до този момент, служат за моделиране на заплахи, но се различават по отношение на функциите си, потребителската аудитория и областите на фокус. Irius Risk и MTMT са по-подробни и може да са подходящи за по-широк спектър на софтуерна разработка и системи, докато OWASP Threat Dragon специално се насочва към сигурността на уеб приложенията и подчертава сътрудничеството в софтуерната общност благодарение на своята функция с отворен код.

Това включва заплахи като фалшифициране на данни, кражба на самоличност, манипулиране на данни, разкриване на информация, отказ на услуга (недостъпност на системата) и повишаване на привилегиите, съгласно рамката STRIDE. Освен това, те адресират десетте най-големи потенциални уязвимости на уеб приложенията според OWASP. Чрез интегрирането на тези модели на заплахи, организациите могат проактивно да осигурят своите системи срещу различни кибер-заплахи, подобрявайки цялостната си сигурност.

Изводи и заключения от втора глава

Всеки инструмент има своите уникални силни и слаби страни. Изборът на правилния инструмент зависи от фактори като удобство на използване, изисквания за интеграция и специфичния фокус за моделиране на заплахи. Като заключение за всеки инструмент за моделиране на заплахи в уеб-базирана архитектура, могат да се направят следните изводи:

- Trike предлага удобен за потребителя интерфейс с възможност за създаване на персонализирани шаблони. Инструментът е лесен за използване, тъй като поддържа множество рамки за потенциални заплахи и противодействия, в допълнение към STRIDE и OWASP Top 10. Това го прави добър кандидат, ако системни или мрежови архитекти трябва да оценят различните слоеве на OSI модела.

- Irius Risk: Предоставя напреднало моделиране на заплахи, управление на риска и съответствие с регулаторни изисквания. Могат да се създават диаграми, които генерират потенциални заплахи и противодействия в реално време. Това е особено ценно за екипи за изследвания и разработки, които следват подхода на Secure Development Lifecycle при разработката на софтуер. Irius Risk поддържа множество рамки за моделиране на заплахи, а ако дадена

рамка не е налична, инженерите по сигурността могат да създават персонализирани рамки, съобразени с нуждите на организацията. Инструментът е възможно да се интегрира в DevOps цикъла и ако се генерират множество потенциални заплахи, той може да бъде конфигуриран така, че да блокира напредъка на разработката в тръбопровода.

- Microsoft Threat Modeling Tool: Поддържа интеграция в екосистемата на Microsoft, като предлага възможности за моделиране на заплахи на корпоративно ниво. Инструментът поддържа само моделиране на заплахи чрез STRIDE и OWASP Top 10 за продуктите на Microsoft.

- OWASP Threat Dragon: Инструмент с отворен код, разработен основно за поддръжка на моделиране на заплахи по OWASP Top 10, използвайки STRIDE. Фокусиран е единствено върху тестването на сигурността на уеб приложения и не може да се използва като основа за моделиране на заплахи в други области, като сигурност на операционни системи или мрежови оценки.

ГЛАВА 3. МОДЕЛ ЗА ИНТЕГРИРАНЕ НА МОДЕЛИРАНЕТО НА ЗАПЛАХИ В DEVSECOPS ТРЪБОПРОВОД ЗА ПОДОБРЕНО РАЗРАБОТВАНЕ НА СОФТУЕР

Действие на DevSecOps тръбопровод

Моделирането на заплахи и откриването на уязвимости са ключови теми, които подчертават значението на проактивното идентифициране и смекчаване на заплахите, особено в AWS среди. Друго изследване представя CoReTM, рамка за съвместно моделиране на заплахи, която позволява на междудисциплинарни екипи систематично да адресират уязвимости. Тези открития подкрепят подхода „shift left“ за ранно интегриране на сигурността в процесите на разработка.

Моделиране на заплахите като методологичен компонент на сигурността в DevSecOps практиките.

Действителният модел на DevSecOps е показан на фиг. 15, който бе създаден в сътрудничество с екип от разработчици в една голяма софтуерна компания. Оценяване на текущия DevOps модел и подобряването му чрез включване на киберсигурност в неговата основа е задължителен процес.

Фигура 15 DevSecOps модел

Предложеният модел е автоматизиран с помощта Дженкинс, който е реализиран да предприеме следните стъпки:

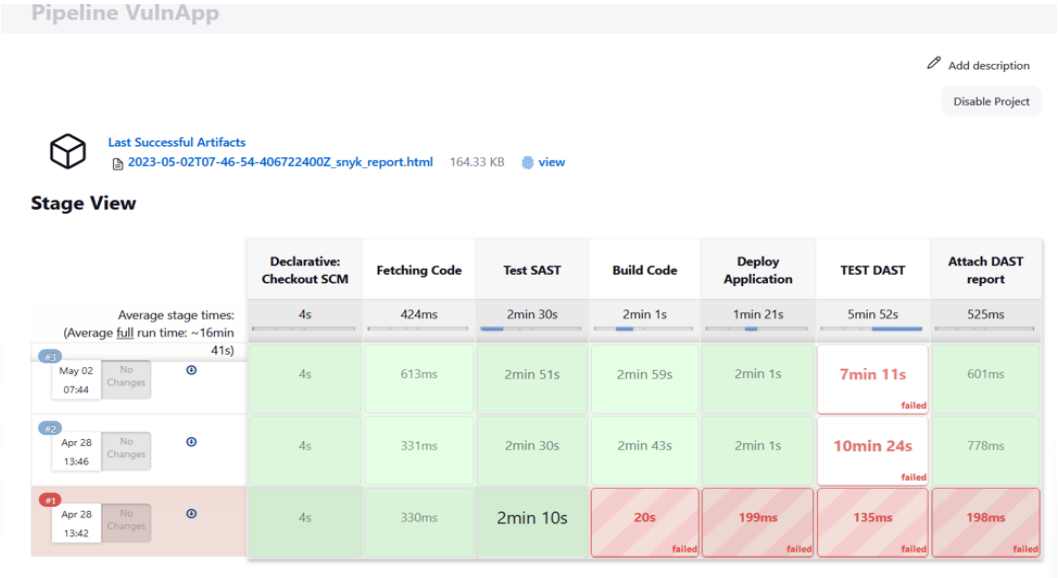
1) По време на етапа „Fetching code“, Дженкинс се свързва с отдалеченото частно хранилище на Github и изтегля пълния изходен код локално на Windows виртуалната машина.

2) Последван от етапа „Execute SAST“, се стартира инструмента за статично сканиране на приложения с отворен код - Snyk, за да се сканира изтегления програмен код от хранилището. Генерира се отчет. По време на този етап Snyk може да се инструктира да спре DevOps етапа въз основа на броя на откритите уязвимости и тяхната сериозност в програмния код

3) Етапът „Build application“ компилира Java уеб приложението Spring Petclinic с помощта на инструмента за компилиране Maven. Очакваният изходен файл трябва да се експортира като файлово разширение Jar.

4) Етап „Deploy application“ стартира приложението Petclinic.

Финализира се последния етап „TEST DAST“, за да се извърши динамично тестване на сигурността спрямо работещото приложение. По време на тази фаза се стартира така наречения DAST инструмент Owasp ZAP, използвайки докер контейнер, който има за цел да сканира приложението за уязвимости Spring Petclinic. Когато сканирането е готово, Owasp ZAP генерира доклад с намерените уязвимости, който може да се анализира, за да се направи проверка на това какви проблеми са открити по време на сканирането.



Фигура 17 DevSecOps модел в Jenkins

Файлът на Дженкинс е програмиран с помощта на езика за програмиране Groovy, който е поддържаният от Дженкинс език за програмиране за автоматизация. Веднъж зареден в Дженкинс, се изпълнява всяка част от функциите на етапите, започвайки от „Извличане на код“ до последния етап „TEST DAST“. Всеки път, когато се задейства нова компилация и Дженкинс се опита да извлече код от частното хранилище на GitHub, по дизайн той винаги изтегля Jenkinsfile, който зарежда инструкциите, които стартират модела на DevSecOps тръбопровода. На фиг. 17 е показан крайният резултат от DevSecOps модела изобразен като етапи от различните стъпки.

За да се потвърди предложеният модел, се използва примерно приложение като начална точка, за да се демонстрира как статичното и динамично тестване на сигурността на приложението може да се интегрира в етапите на DevSecOps тръбопроводния модел. За тази цел екипът за разработка създаде частно хранилище на GitHub, което е създадено и клонирано от официалното хранилище на Spring Petclinic в GitHub. Клонираното частно хранилище на GitHub е точно копие на оригиналното хранилище.

Рамка за интегриране на моделирането на заплахи в DevOps за подобрено разработване на софтуер

Предложеният модел на DevSecOps процес на разработка се валидира чрез реализиране на рамка, създадена в сътрудничество с един от екипите за разработка в рамките на отдела за изследвания и развитие на компания за

разработка на софтуер. Направена е оценка за подобряване на настоящият модел DevOps чрез включване на сигурността в процес на разработка.

С развитието на DevOps практиките, необходимостта от интегриране на мерки за сигурност става все по-актуална. Традиционните оценки на сигурността често не успяват да отговорят на бързите цикли за разработка на софтуер, характерни за DevOps.

На база предложения модел се реализира рамка за вграждане на моделиране на заплахи в Jenkins тръбопровода, последвана от интегриране на данните от потенциални заплахи в база данни и тяхното използване за автоматизирани проверки за уязвимости с помощта на инструменти за статично и динамично тестване на сигурността на приложенията. Целта е да се подобри сигурността на приложенията, без да се забавя скоростта на разработката на софтуер. Чрез прилагането на този подход организациите могат да повишат нивото на сигурност на своите приложения, като проактивно идентифицират и смекчават потенциалните заплахи през целия жизнен цикъл на разработката.

Чрез прилагането на рамка за интегриране на моделирането на заплахи в DevOps конвейер се цели да се сравнят идентифицираните предизвикателства, свързани с нея.

Фигура 31 илюстрира интеграцията на моделирането на заплахи в рамката на Jenkins DevOps конвейера.

Фигура 31 Интеграция на моделирането на заплахи в Jenkins DevOps конвейер.

Процесът е разделен на няколко етапа:

- *Моделиране на заплахите:* В етапа на **моделирането на заплахи**, екипите по сигурността и разработчиците работят в тясно сътрудничество, за да идентифицират потенциални заплахи и уязвимости в архитектурата на системата. Тази фаза е обозначена в диаграмата с кутията "Моделиране на заплахи". По време на този етап, на базата на идентифицираните високорискови

потенциални заплахи за сигурност на софтуера, може да се вземе решение тръбопроводът да бъде спрян и да се изиска от разработчиците и архитектите да обмислят възможните решения или противодействия въз основа на намерените заплахи. В случай, че екипът по разработка приеме риска, може да се разреши продължаването на тръбопровода към следващата фаза.

- *Извличане на данните от моделиране на заплахите*: Резултатите от етапа на **моделиране на заплахи** се извличат в структуриран формат, който може да се обработи допълнително. Тази стъпка гарантира, че данните за заплахите са готови за интеграция с базата данни.

- *Интеграция с база данни*: Структурираните данни за заплахи се въвеждат в база данни, което позволява ефективно извършване на заявки и анализ на потенциалните уязвимости, които биват съхранени в базата. Тази интеграция е от съществено значение за централизирането на информацията, свързана с идентифицираните заплахи, и за осигуряване на бърз достъп до нея.

- *Динамично сканиране на заплахите:* Използвайки интегрираните данни за заплахи от базата данни, се провеждат автоматизирани сканирания за уязвимости с цел идентифициране и адресиране на потенциални проблеми със сигурността. Тази фаза се фокусира върху сканирането на конкретни области, които са идентифицирани като високорискови по време на етапа на моделиране на заплахи.

- *Анализ и доклад*: Резултатите от сканиранията за уязвимости се анализират и се извеждат в доклади. Тези доклади играят важна роля за екипите по сигурност и разработчиците, като им помагат да разберат идентифицираните уязвимости и да предприемат подходящи мерки за смекчаване на рисковете.

- *Дженкинс DevOps тръбопровод*: Целият процес е интегриран в Jenkins конвейера, което осигурява автоматично провеждане на оценки за сигурността при всеки цикъл за компилирането и изграждането на софтуер. Тази автоматизация е от ключово значение за интегрирането на сигурността в DevOps практиките, тъй като позволява непрекъснато наблюдение и оценка на уязвимостите по време на целия жизнен цикъл на разработка на софтуер.

Тази рамка гарантира, че съображенията за сигурност са внедрени през целия жизнен цикъл от разработката на софтуер, като помага проактивно да се идентифицират и смекчават потенциални заплахи. Чрез интегрирането на оценки за сигурността в ранните етапи от разработката на софтуер, екипите

могат да установят уязвимости и рискове, преди те да бъдат експлоатирани от хакери.

Изводи и заключения към трета глава

Представеното в трета глава на дисертационния труд изследване предоставя стабилна рамка за интегриране на моделирането на заплахи в Jenkins DevOps тръбопровода. Автоматизирането на откриването на заплахи и използването на структурирани данни в база данни за сканиране на уязвимости позволява на организациите да управляват проактивно рисковете за сигурността, като осигуряват разработването на сигурни софтуерни системи в среда на непрекъснато внедряване.

Чрез вграждане на процесите по сигурност в CI/CD тръбопроводи, тази рамка решава често срещания конфликт между нуждите от бърза разработка и строгите изисквания за сигурност. Тя осигурява мащабируемост и ефективен начин за поддържане на високи стандарти за сигурност, без да се компрометира скоростта на разработката. Непрекъснатото актуализиране на моделите на заплахи и интегрирането на техните данни в централна база за насочено сканиране гарантира, че практиките за сигурност се развиват паралелно с приложението, като се адресират нововъзникващите заплахи.

Моделирането на заплахите е ключов аспект на DevSecOps, който позволява на организациите да идентифицират и смекчават потенциални рискове още в началните етапи на разработка. Настоящото изследване демонстрира, че използването на автоматизирани методи за моделиране на заплахи, комбинирани с динамично и статично сканиране, осигурява ефективен механизъм за превенция срещу кибер атаки. Интеграцията на тези процеси в Jenkins CI/CD тръбопровода позволява създаването на автоматизирани механизми за сигурност, които намаляват вероятността от компрометиране на приложенията.

Представеният DecSecOps модел може да се разшири чрез използване на усъвършенствани методи за анализ на заплахи, като машинно обучение и изкуствен интелект, за да се повиши ефективността на предсказване и превенция на атаки. Допълнително, интегрирането на Runtime Application Self-Protection (RASP) технологии в DevSecOps конвейера може да осигури защита на приложенията в реално време, минимизирайки риска от експлоатация на уязвимости. Прилагането на DevSecOps методологии и автоматизацията на моделирането на заплахи са ключови за подобряване на сигурността на софтуерните продукти. Проактивният подход в CI/CD процесите осигурява не

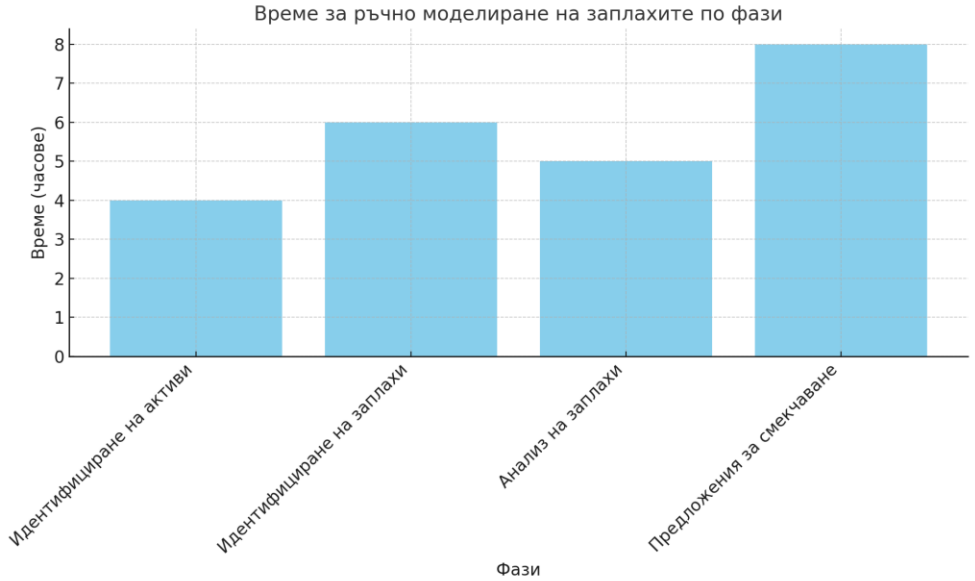
само по-безопасни приложения, но и **значително намалява времето за реакция при инциденти**, като по този начин се повишава надеждността и устойчивостта на софтуерните системи срещу злонамерени атаки. Моделите на машинно обучение могат да анализират исторически данни за заплахи, за да предскажат потенциални уязвимости в нови промени в кода, осигурявайки още по-проактивни мерки за сигурност. Допълнително могат да бъдат внедрени механизми за мониторинг в реално време и известяване, за да се откриват и реагират на заплахи при тяхното възникване, което още повече укрепва сигурността на приложението.

В заключение, интеграцията на моделирането на заплахи в DevOps тръбопровода, както е описано в тази рамка, представлява значителен напредък в осигуряването на съвременните практики за разработка на софтуер. Тя комбинира автоматизация, управление на структурирани данни и насочено сканиране за уязвимости, за да предостави цялостно решение за сигурност, което е едновременно ефективно и ефикасно.

ГЛАВА 4. СРАВНЕНИЕ НА ПОДХОД ЗА РЪЧНИ И АВТОМАТИЧНИ МЕТОДИ ЗА МОДЕЛИРАНЕ НА ЗАПЛАХИ

Ръчно моделиране на заплахи

Фиг. 38 визуализира времето нужно за ръчно моделиране на заплахи, изведено от научните анализи и изследвания, проведени в Глава 1 и Глава 2.

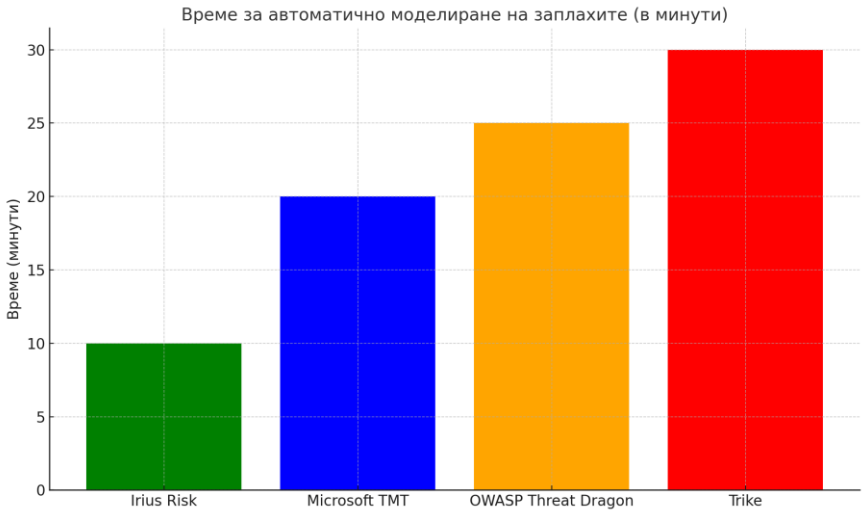


Фигура 38 Графика за времетраене на ръчно моделиране на заплахите

Експерименталните данни, представени на Фиг. 38, са получени в рамките на съвместно изследване с експертния екип по киберсигурност и екип от разработчици в софтуерна организация. Процесът по моделиране на заплахи е реализиран ръчно в четири фази – идентифициране на активи, идентифициране на заплахи, анализ на заплахи и формулиране на предложения за смекчаване – съгласно описаната методология. За структурирането и класифицирането на заплахите са използвани утвърдените рамки OWASP Топ 10 и MITRE ATT&CK, които осигуряват общ език и критерии за оценка на уязвимости и техники на нападение. Оценката и идентификацията са базирани на експертно мнение на участващите екипи. Времето за изпълнение на всяка фаза е проследено и регистрирано в системата за управление на задачи Jira, измерено чрез хронометриране и усреднено след двукратно повторение. Получените стойности са валидирани чрез експертна оценка, което осигурява прозрачност, възпроизводимост и научна достоверност на резултатите.

От графиката могат да се направят следните изводи: (Фиг. 38):

- Фазите на ръчното моделиране на заплахите изискват значително време.
- Идентифициране на заплахи и предложения за смекчаване са най-времеемките стъпки, които се равняват на общо минимум 14 часа.
- Анализът на заплахи и предложенията за смекчаване също изискват значителни ресурси.
- Този процес често води до забавяне на внедряването на решения и до рискове, свързани с човешка грешка.

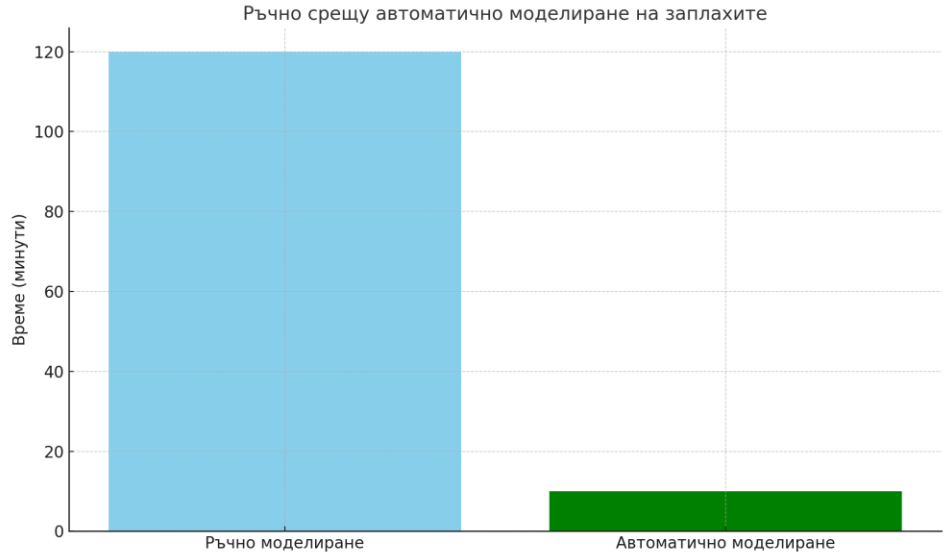


Фигура 39 Сравнение между четири инструмента за автоматизирано моделиране на заплахите

Спрямо направените изследвания и тестове, най-доброто решение, което може да се използва за изпълнение на задачите в дисертационния труд с цел създаване на автоматизирана рамка за моделиране на заплахите е Irius Risk:

- Най-бърз инструмент - завършва автоматичното моделиране между 5 и 10 минути.
- Интуитивен интерфейс - лесен за използване от екипи без опит в моделирането на заплахи.
- Интеграция с DevSecOps - възможност за вграждане в CI/CD процеси.
- Подробен анализ - предоставя конкретни препоръки за намаляване на рисковете.

На Фиг. 40 е представено сравнението между ръчно и автоматизирано моделиране (автоматизирано моделиране на заплахи чрез инструменти във Фиг. 39) на заплахи, извършено върху една и съща архитектурна диаграма от високо ниво, която включва уеб клиент, уеб сървър и свързана към него база данни, внедрени в публична инфраструктура. Целта на сравнението е да се оцени времевата ефективност на двата подхода при запазване на функционалната същност на процеса по моделиране.



Фигура 40 Ръчно срещу полу-автоматично (Автоматизирано моделиране на заплахи)

При ръчния метод процесът, извършен чрез експертна оценка на индивидуално експериментално проучване е реализиран в четири последователни стъпки:

- Идентифициране на активи

- Идентифициране на заплахи,
- Анализ на заплахи
- Формулиране на предложения за смекчаване.

Във фазата на идентифициране на компонентите са разгледани трите основни елемента от архитектурната диаграма – уеб клиент, уеб сървър и база данни. За идентифициране и класифициране на заплахите са използвани рамките OWASP Top 10 и MITRE ATT&CK Framework, които осигуряват стандартизирана основа за анализ на уязвимости и техники на атака. Анализът на възможните механизми за противодействие е извършен чрез допълнително проучване на онлайн източници, включително официалните платформи на OWASP и MITRE, като целият процес е реализиран без използване на автоматизирани инструменти.

Времето, отразено в графиката, обхваща пълния цикъл на ръчното моделиране – от идентифицирането на активите до формулирането на мерки за смекчаване – и е измерено с помощта на уеб базирания хронометър на Google. Стойностите са определени въз основа на експертна оценка и реално проследяване на времето за изпълнение.

Резултатите показват значително съкращаване на времето при автоматизираното моделиране в сравнение с ръчния процес. Автоматизираните инструменти реализират пълния цикъл по идентифициране и оценка на заплахите над десет пъти по-бързо, което подчертава предимствата на внедряването им в съвременните DevSecOps практики.

Предложение на метод за подобрение и автоматизиране на процеса за моделиране на заплахите и динамично сканиране за уязвимости

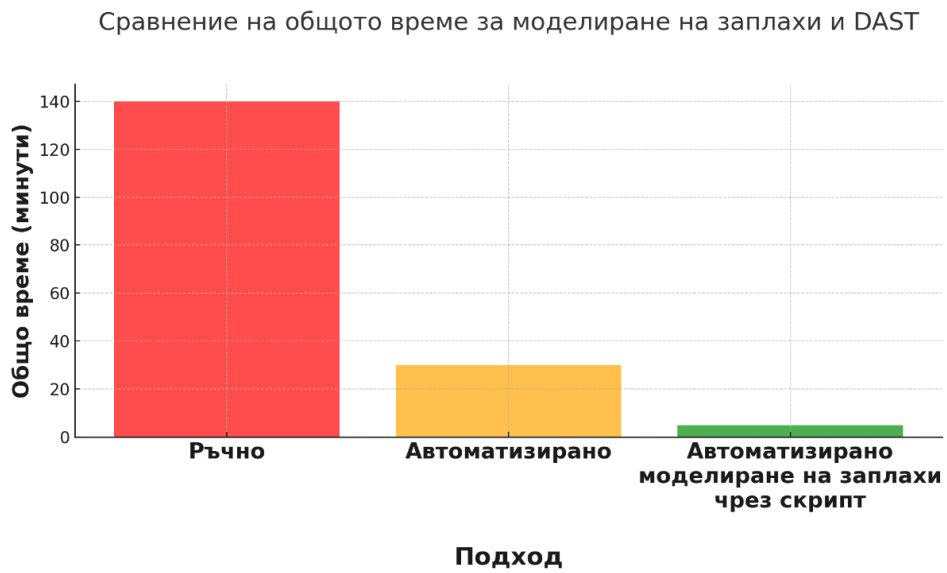
От направените проучвания, тестове и получени резултати от Глава 2 и Глава 3 става ясно, че моделирането на заплахи заедно с динамичното тестване за уязвимости в дадено уеб приложение могат да бъдат тежки процеси, които изискват прекалено дълго време и ресурси, превръщайки ги в методики, които не всеки екип или организация би предприела да имплементира в своите среди. Поради тази причина е създаден метод за улесняване на процесите за моделиране на заплахи и динамичното сканиране на уязвимости чрез скриптове, които автоматизират целия процес от начало до край, съкращавайки времето, което е нужно за да завърши целия цикъл от генериране на модел за заплахи до сканирането на уязвимости в едно уеб приложение.

Скриптът преминава през няколко стъпки:

1. Стъпка едно – чрез API заявка се изгражда връзка към IriusRisk, за да свали вече генерирана диаграма от високо ниво за моделиране на заплахи. Скриптът извлича диаграмата с всички потенциални заплахи и извежда резултата в XML формат (Приложение 14.4)
2. Стъпка две – чрез инструмента **xmlstarlet** се извършва проверка получения XML файл от Irius Risk с потенциални заплахи и се извежда информацията съдържаща CWE данни в JSON формат (***Common Weakness Enumeration*** *е общоприет стандарт и речник на често срещани слабости в софтуерната сигурност. Той подпомага разработчиците, тестерите и експертите по сигурност при идентифициране, класифициране и адресиране на проблеми, за да подобрят общата устойчивост на приложенията*)
3. Стъпка три – стартиране на динамично сканиране срещу уеб приложението Spring Pet clinic, което е инсталирано на локален Docker container и е с IP адрес http://192.168.88.135:8080 порт 8080. Извеждат се резултатите с уязвимостите от динамичното сканиране в JSON формат.
4. Стъпка четири – Извършва се сравнение на CWE данните от файла за моделирането на заплахите от Irius Risk срещу получените резултати от динамичното сканиране за уязвимости. Идеята е да се проверят дали резултатите от моделирането на потенциални заплахи са поправени и разрешени след динамичното тестване за уязвимости. Съпоставяме CWE данните от първия метод, когато заплахите се моделират и системата все още не е изградена и накрая след като уеб приложението е в готов продукт и CWE резултатите са взети от вече функциониращото приложение.

Сравнение на резултатите получени от методите за ръчно, полу-автоматично моделиране на заплахите и автоматизирано моделиране на заплахите.

Фигура 44 показва сравнението на общото време за моделиране на заплахи и динамично тестване (DAST). Предложеното в дисертационния труд решение със скрипт значително намалява времето до 5.36 минути, сравнено с 140 минути при ръчното моделиране и 30 минути при полу-автоматизирания подход.



Фигура 44 Сравнение на времето за моделиране и тестване

На Фиг. 44 е представено сравнението на общото време за моделиране на заплахи и извършване на динамично тестване за уязвимости (DAST) при три различни подхода:

- Ръчно моделиране
- Автоматизирано моделиране чрез инструменти
- Автоматизирано моделиране чрез скрипт

Експериментът комбинира процеса по моделиране на заплахи и времето, нужно за динамичното сканиране на уеб приложение. И в трите случая е използвана една и съща архитектурна диаграма от високо ниво, включваща уеб клиент, уеб сървър и база данни, свързани в публична инфраструктура.

Изводи и заключения към четвърта глава

Извършени са детайлни измервания и анализи на времената за моделиране на заплахи и динамично тестване за уязвимости с помощта на три различни подхода: ръчно моделиране на заплахи, полу-автоматизирано моделиране чрез стандартни инструменти, и автоматизирано моделиране на заплахи чрез разработен скрипт. Целта на изследването е да се оцени ефективността, точността и времевата оптимизация на всеки от тези методи.

Времеви измервания и анализ:

1. Ръчно моделиране на заплахи
    - Процесът отнема значително време – около 120 минути за моделиране и допълнителни 10-20 минути за динамично тестване чрез OWASP ZAP и Burp Suite Professional.
    - Изисква висока експертиза и човешко участие, което увеличава риска от грешки.
2. Полу автоматизирано моделиране на заплахи
    - Използвани е инструмента Irius Risk, който автоматизира частично процеса.
    - Времето за моделиране се намалява до 20-30 минути, но динамичното тестване отнема същото време 10-20 минути, както при ръчния подход.
3. Автоматизирано моделиране чрез скрипт
    - Скриптът интегрира Irius Risk за моделиране на заплахи и OWASP ZAP за динамично тестване. Целият процес – от извличане на заплахите до динамичното тестване – се изпълнява за 5.36 минути (включително 36 секунди за динамично сканиране).
4. Основни резултати:
    - Ръчно моделиране на заплахи - Въпреки високата точност, процесът е бавен, скъп и податлив на грешки. Трудно е приложим за големи и сложни системи.
    - Полу-автоматизирано моделиране на заплахи чрез инструменти - Намаляват времето за анализ, но изискват човешка намеса, но имат ограничения в автоматизацията и интеграцията с CI/CD тръбопроводи.
    - Автоматизирано моделиране чрез скрипт - Най-ефективният подход, който комбинира висока точност, минимална човешка намеса и пълна автоматизация. Скриптът предоставя стандартизирани и валидирани резултати, приложими за модерните DevSecOps практики.
5. Характеристики на динамичното тестване:
    - Сканирано е уеб приложението Spring PetClinic, което предоставя среда със средна сложност, подходяща за измерване на ефективността на инструментите.

- Времето за сканиране чрез OWASP ZAP е само 36 секунди, което демонстрира бързината на автоматизирания подход.

6. Заключения:
   - Автоматизираното моделиране чрез скрипт съкращава времето за анализ с над 95% в сравнение с ръчния подход.
   - Интеграцията на Irius Risk и OWASP ZAP гарантира висока степен на точност и проверка на резултатите.
   - Разработеният метод за автоматизиране на моделиране не заплахите и динамичното тестване за уязвимости чрез скрипт премахва нуждата от ръчни действия и гарантира стандартизиран процес, който може да се интегрира в CI/CD тръбопроводи.
   - Приложимост: Подходът е подходящ за съвременните изисквания в DevSecOps, като осигурява постоянен мониторинг на сигурността и бърза реакция на нови заплахи.

Тези резултати и анализи показват ясно, че автоматизацията на моделирането на заплахи и динамичното тестване за уязвимости чрез скриптове предоставя значителни предимства в бързото действие, ефективност и качество, което го прави идеалният избор за съвременните организации и разработчици. Автоматизираното моделиране на заплахи и динамичното тестване за уязвимости, реализирани чрез предложената скриптова интеграция, демонстрират значително подобрение в ефективността и точността на целия процес. Скриптът автоматизира ключови стъпки като извличане на заплахите от IriusRisk, динамично сканиране чрез OWASP ZAP и сравнение на резултатите, което намалява общото време за изпълнение до 5.36 минути. В ръчния подход този процес отнема над 300 минути, докато полу-автоматизираните инструменти изискват около 30 минути. Използването на стандартизирани инструменти и интеграция в CI/CD процеси не само съкращава времето, но и елиминира човешките грешки, които често съпътстват ръчните методи.

Резултатите показват, че всички потенциални заплахи, идентифицирани чрез IriusRisk, са успешно валидирани и отстранени след динамичното тестване с OWASP ZAP. Освен това, скриптовият подход доказва своята приложимост при реални сценарии с уеб приложения като Spring PetClinic, което служи като пример за средно сложна архитектура. Важно е да се отбележи, че автоматизацията не само ускорява процеса, но и осигурява по-

точни и последователни резултати, което прави този метод подходящ за динамичната и високо рискова среда на съвременните DevSecOps практики.

При проведените тестове върху уеб приложението Spring PetClinic OWASP ZAP откри повече уязвимости, но с по-голяма вероятност за фалшиви положителни резултати (false positives). Burp Suite Professional, макар и с по-малък обхват на открити уязвимости, демонстрира висока надеждност и точност на резултатите. Анализът на тежестта на уязвимостите показва, че OWASP ZAP предоставя по-широка информация, включително информационни и уязвимости с ниска тежест, които могат да бъдат полезни в ранните етапи от разработката на софтуер. Burp Suite Professional, от своя страна, се отличава с фокусирано откриване на по-значими уязвимости, което го прави подходящ за финални проверки на сигурността.

Комбинацията от двата инструмента предлага оптимален подход за осигуряване на сигурността на уеб приложения. Такава комбинация гарантира както бързо идентифициране на потенциални рискове, така и задълбочена проверка за елиминиране на критични уязвимости.

ПРИНОСИ

Научни приноси:

НП-1. Разработен е автоматизиран метод за моделиране на заплахи и динамично тестване, който съкращава времето с 82% за идентификация и валидиране на уязвимости в уеб приложения спрямо полуавтоматичния метод за моделиране на заплахи.

Научно-приложни приноси:

НПП-1. Извършен е сравнителен анализ на методите за моделиране на заплахи, включително ръчни, полуавтоматични и автоматизирани подходи, като са идентифицирани техните силни и слаби страни.

НПП-2. Предложен е цялостен подход за управление на сигурността в CI/CD тръбопроводи чрез интегриране на инструменти за статично (SAST) и динамично (DAST) тестване в DevSecOps рамка.

НПП -3. Направена е съпоставка на резултатите от моделирането на заплахи с резултатите от статично и динамично тестване за уязвимости, което гарантира последователност и валидност на анализа.

Приложни приноси:

ПП-1. Създаден е софтуерен прототип на автоматизиран конвейер за моделиране на заплахи и динамично тестване, който използва Irius Risk и OWASP ZAP, позволявайки ефективно управление на сигурността в ранните етапи от разработката.

ПП-2. Създадена е DevSecOps рамка, която комбинира инструменти за SAST (Snyk), DAST (OWASP ZAP) и моделиране на заплахи и осигурява непрекъсната интеграция и мониторинг на сигурността в CI/CD тръбопроводи.

ПП-3. Проведени са експерименти с уеб приложението Spring PetClinic, демонстриращи ефективността на предложените методи и инструменти в реални условия.

ПП-4. Разработен е скрипт, който автоматизира процеса на моделиране на заплахи и DAST, като същевременно предоставя възможност за бъдещо надграждане с PYTM за напълно кодова автоматизация.

Публикации

1. Lyuben Nikolov and Adelina Aleksieva-Petrova, Action Research on the DevSecOps Pipeline, 2023 International Scientific Conference on Computer Science (COMSCI), DOI: 10.1109/COMSCI59259.2023.10315920, https://ieeexplore.ieee.org/document/10315920.

2. Lyuben Nikolov and Adelina Aleksieva-Petrova, Threat Modeling Based on STRIDE Framework for Web Application, The International Conference on Applied Physics, Simulation and Computing, Rome, Italy,. June 20-22, 2024

3. Nikolov, Lyuben & Aleksieva-Petrova, Adelina. (2024). Framework for Integrating Threat Modeling into a DevOps Pipeline for Enhanced Software Development. 1-5. 10.23919/SoftCOM62040.2024.10721871.

4. Nikolov, Lyuben and Aleksieva-Petrova, Adelina. (2025). Enhancing Application Security: The Role of Automation in Threat Modeling and Dynamic Vulnerability Assessment. In *WSEAS Transactions on Systems and Control (под печат)*

Abstract of the Ph.D. Thesis

Lyuben Asenov Nikolov, M.Sc.

METHODS AND RESOURCES FOR ENSURING THE SECURITY AND PROTECTION OF WEB APPLICATIONS

The primary objective of this dissertation is to conduct a comprehensive investigation into the practices and methodologies employed by malicious actors in cyberspace for executing web-based attacks. The research aims to propose and evaluate an integrated approach to automated threat modeling, combining multiple tools and incorporating both static and dynamic vulnerability assessment techniques to strengthen security throughout the software development lifecycle of web applications. By embedding security mechanisms early and continuously within the development process, the proposed framework aspires to enhance the resilience, reliability, and overall security posture of web systems operating in today's highly dynamic and distributed computing environments. The research focuses on the speed and efficiency of threat identification approaches, as well as the possibilities for improving web application security by implementing static and dynamic vulnerability analysis methods. The emphasis is on improving these techniques to increase accuracy, automation, and reduce response time when detecting security issues.

The object of study in this dissertation encompasses the approaches, methods, and algorithms for threat modeling, as well as the processes of static (SAST) and dynamic (DAST) testing of web applications, aimed at identifying, assessing, and mitigating vulnerability-related risks. The research further examines the integration of these methods into the Software Development Life Cycle (SDLC) and analyzes their impact on enhancing the security and efficiency of software systems.

This dissertation emphasizes the significance of automation and integration of threat modeling and vulnerability testing within modern DevSecOps practices. The dissertation demonstrates that combining tools such as Irius Risk, OWASP ZAP, and Burp Suite Professional, supported by automation scripts, considerably enhances the efficiency, accuracy, and speed of vulnerability detection in web applications. The proposed automated approach minimizes human errors, shortens testing time, and increases result reliability, offering a substantial improvement over traditional manual or semi-automated methods. The outcomes confirm the effectiveness of the proposed framework for optimizing the software development lifecycle by strengthening security, reducing costs, and contributing to the advancement of cybersecurity practices in the software industry.