НАУЧНИ ТРУДОВЕ НА УНСС (5) 2023, ИК – УНСС
ISSN (online): 2534-8957; ISSN (print): 0861-9344; http://unwe-research-papers.org/
DOI: https://doi.org/10.37075/RP.2023.5.05

МЕТОДИ ЗА ТЕСТВАНЕ, ОЦЕНКА И АНАЛИЗ НА СИГУРНОСТТА НА УНИВЕРСИТЕТСКИ ИНФОРМАЦИОННИ СИСТЕМИ

Венко Андонов[1]
e-mail: vandonov@unwe.bg

Резюме

Методите за тестване, оценка и анализ на сигурността на университетски информационни системи, имат за цел изграждането на унифицирана рамка, която да води към повишаване на информационната сигурност в този тип институции, при които е характерно администрирането на огромен обем чувствителна информация от разнообразно естество. Тестването на сигурността е многостранен процес, включващ стъпки за гарантиране на безопасността на системите в тяхната цялост. Предложените методи са апробирани с три системи на Университета за национално и световно стопанство, като тук е представена една от тях; и са изведени изводи за повишаване на сигурността и включване на описаните процедури като постоянна практика.

Ключови думи: информационна сигурност, киберсигурност, тестване, софтуер, уязвимости

JEL: I23, M15

Увод

Тестването на сигурността на софтуера е общ термин за набор от методологии и практики, насочени към идентифициране на уязвимости в софтуера и системите, като се гарантира, в определена степен, че те са защитени от потенциални заплахи. Съществуват няколко основни направления в това отношение.

Едно от най-важните и с най-висока степен на полезност е тестът за проникване (penetration testing / pen testing) – при него се симулират кибератаки върху системи, за да установят уязвимости, които могат да бъдат използвани (Bertoglio, 2017). Процесът обикновено следва следните стъпки: планиране и разузнаване, сканиране за стандартни уязвимости, изпълнение на тестови атаки през евентуално установени уязвимости, създаване на доклад с препоръки за отстраняване на проблемите.

[1] Асистент, доктор, катедра „Информатика“, факултет „Приложна информатика и статистика“, УНСС

Статично тестване за сигурност на приложението (SAST) като метод включва анализ на програмния код, компилиран байт код или двоични файлове на даденото приложение (Li, 2023). Той открива уязвимости в началото на цикъла на разработка, което е полезно, но може да обхване само стандартни ситуации, подлежащи на анализиране на кода. Динамичното тестване на сигурността на приложенията (DAST) оценява приложенията в тяхното работещо състояние (Severns, 2021). Интерактивното тестване на сигурността на приложенията (IAST) обединява силните страни както на SAST, така и на DAST, като изследва приложенията по време на тяхното изпълнение, което често води до по-точно откриване на уязвимости (Comparitech, 2023).

Анализ на състава на софтуера е метод, при който се идентифицират уязвимости в софтуерни компоненти с отворен код и библиотеки, разработени от трети страни. Тъй като тези компоненти могат да въведат външни уязвимости, може да се окажат вектор за атака на системата, без да има уязвимости в самата нея като реализация (Myers, 2021).

Оценката на конфигурацията на сигурността включва щателен преглед на конфигурациите на системата и приложенията, за да се провери спазване на оптималните практики за сигурност (CIS, 2021).

Самозащита на приложения по време на изпълнение (RASP) позволява разпознаване и противодействия на заплахи в реално време (Alvarenga, 2023).

Red-teaming упражненията имат за цел да се организират цялостни симулирани кибератаки, които тестват цялата дигитална инфраструктура на организацията, дори физическия достъп, за да оценят цялостните защитни способности. В допълнения съществува и метод на Blue-teaming – в противоположност на другия екип, ИТ специалистите се защитават от кибератаките, което симулира начина на работа в такъв тип ситуация (Niezen, 2018; Certitude Security, 2020).

Наблягайки на проактивната защита, непрекъснатият мониторинг включва наблюдение в реално време за бързо откриване и справяне със заплахите, когато се материализират (Ge, 2015).

Тестване на сигурността на софтуерни системи

Тестването на сигурността на софтуерна система е процес, който цели да идентифицира и отстранява уязвимости, които могат да бъдат използвани от злонамерени атакуващи. Този процес се състои от няколко последователни фази, обхванати като цикъл, всеки от които има своя специфичен обхват и значение (фигура 1).

Определяне на обхват на тестването: тази фаза определя къде, как и до каква степен ще се проведе тестването. Определянето на обхвата уточнява кои системи, приложения или компоненти ще бъдат включени, както и кой

тип уязвимости се търси. Определяне на обхвата е важно, за да се гарантира, че всички критични компоненти на системата са подложени на анализ. Това включва определяне на активите, домейните и приложенията, които ще бъдат тествани, събиране на информация: разузнаване на активи като имена на домейни, поддомейни, IP адреси и достъпни услуги. Също така има значение използваните технологии и версиите на софтуера, както и извличане на данни от публично достъпни източници като WHOIS и Shodan.

Подготовка на средата за тестване: преди да започне самото тестване, е важно да се настрои средата така, че тя да бъде максимално приближена до реалната продукционна среда, но без риска от реален проблем или загуба на данни.

Приоритизиране на потенциални области, слабости и уязвимости: не всички уязвимости са равни. Някои могат да имат по-голямо въздействие върху системата от други. Затова е важно да се оцени кои области са най-критични и да се фокусира върху тях с приоритет.

Използване на инструменти за тестване: съществуват много инструменти, които могат да автоматизират процеса на търсене на уязвимости. Изборът на правилния инструмент е важен за ефективността на тестването. Част от по-често използваните такива са разгледани по-нататък.

Оценка на резултатите от тестването: след завършване на тестването, резултатите трябва да бъдат анализирани за точност и важност. Това помага да се определи какви действия трябва да бъдат предприети.

Изготвяне на план за промени/корекции/подобрения: въз основа на оценката на резултатите се разработва план за промени в системата, с цел отстраняване на идентифицираните уязвимости.

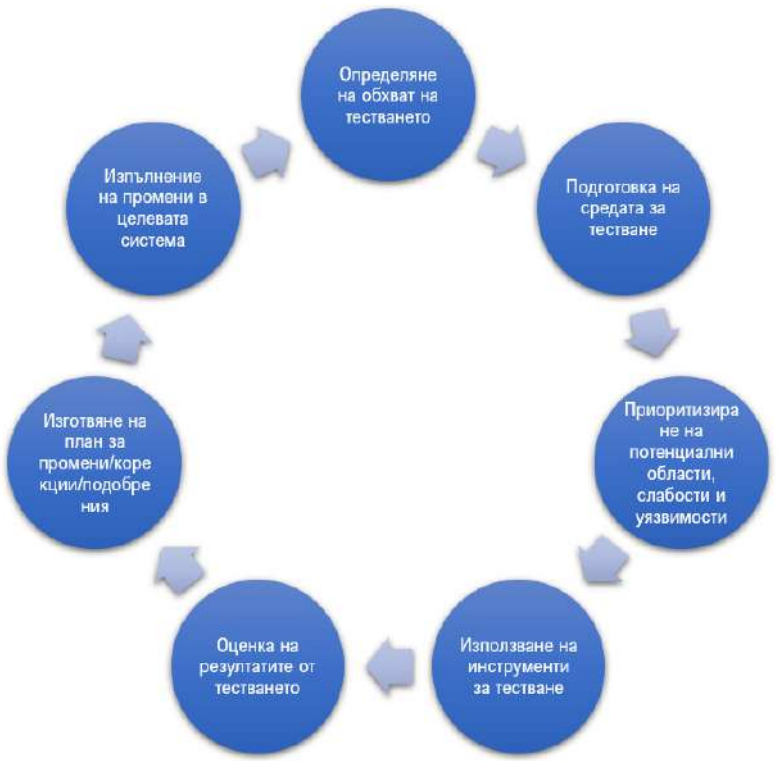
Изпълнение на промени в целевата система: след като е разработен планът, промените трябва да бъдат приложени в системата. Това може да включва отстраняване на пропуски, конфигурационни промени или преработка на програмния код.

Технологиите, заплахите и средата постоянно се променят, затова е важно системите редовно да се преоценяват за нови уязвимости. Всеки от горепосочените етапи допринася за общата цел на подобряване на сигурността на системата и гарантира, че тя е защитена от потенциални атаки.

Номенклатури на слабостите и уязвимостите на софтуерните системи

Съвременните процеси за разработка на софтуер са сложни, което прави системите податливи на множество проблеми в сигурността. Справянето с тези потенциални уязвимости налага единно разбиране на техните харак-

теристики, модели и последици. Common Weakness Enumeration (CWE) е развиваща се платформа и таксономия, която предоставя стандартна номенклатура за софтуерни слабости и уязвимости.



Източник: Разработка на автора

Фигура 1: Процес по тестване на сигурността на конкретна информационна система

Чрез ползването на единна таксономия и номенклатура за софтуерни уязвимости, CWE намалява двусмислието в това как те се дефинират и разбират от различни хора и организации. Служи и като хранилище на натрупаните знания, което позволява на заинтересованите страни да се учат от минали грешки и повтарящи се проблеми със сигурността. С общ набор от термини специалистите по сигурността, разработчиците и изследователите могат да комуникират софтуерните уязвимости по-ефективно. CWE подпомага интегрирането и сравняването на инструменти за сигурност, като гарантира, че те идентифицират и се борят с уязвимостите цялостно. Включва различни списъци, всеки от които е съобразен с различни изисквания (Christey, 2008).

Инструментите за сигурност могат да използват CWE идентификатори, за да комуникират идентифицирани слабости, улеснявайки последователно докладване и смекчаване. Чрез изучаването му разработчиците и другите заинтересовани страни могат да подобрят уменията си в разпознаването и коригирането на уязвимостите.

Потенциални уязвимости на уеб и мобилни приложения

Съвременните системи, които се изграждат в университетите, обикновено са уеб и/или мобилно-базирани, което предполага фокус върху този тип уязвимости. Както уеб приложенията, така и мобилните приложения имат свой уникален набор от потенциални уязвимости. За да предпазят софтуерните системи, от съществено значение е разработчиците и екипите по сигурността да са наясно с тези уязвимости, редовно да тестват своя софтуер и да използват техники за смекчаване, за да намалят потенциалните заплахи.

При **Cross-Site Scripting (XSS) уязвимостите**, нападателите „инжектират“ злонамерени скриптове в уеб страниците. Рисковете са кражба на сесия, на самоличност, злонамерено пренасочване, обезобразяване на сайта и други (Kirda, 2009). При **SQL Injection**, злонамерени SQL изрази (за действия в базата данни) се въвеждат в полета и се изпълняват от уеб приложението. Това може да доведе до рискове като нарушения на данните, неоторизиран достъп до данни и потенциален контрол върху базата данни (Halfond, 2006). При **Cross-Site Request Forgery (CSRF)** жертвите, които са потребители на приложението, са подмамени да извършат нежелани действия на място, където са автентикирани. Това създава рискове като неоторизирани действия от името на удостоверен потребител, кражба на данни и компрометиране на акаунт (Barth, 2008). При **Server-Side Request Forgery (SSRF)** уеб приложението е принудено да прави нежелани HTTP заявки към вътрешен ресурс, което може да доведе до рискове като разкриване на информация и потенциално дистанционно изпълнение на команди върху сървъра. При **грешка в конфигурацията на сървърите** е възможно лошо конфигурираните настройки за сигурност или съобщенията за грешки да разкрият уязвимости или информация. Това създава рискове като нарушения на данните, неоторизиран достъп и допълнителни възможности за използване. При **несигурна десериализация**, изпращането на опасни или злонамерени обекти може да доведе до рискове като дистанционно изпълнение на код и потенциално компрометиране на системата.

При мобилните приложения е възможно да има уязмимости като **несигурно съхранение на данни на устройството** – ако те съхраняват данни локално без подходяща защита, това може да доведе до рискове като излагане на чувствителна информация, кражба на данни и изтичане на иденти-

фикационни данни. При **несигурна комуникация между мобилното приложение и backend сървърите**, това може да доведе до рискове като Man-in-the-middle атаки, кражба на данни и неразрешени действия. При **неподходящо използване на криптография**, слабите или неправилно внедрени криптографски механизми могат да доведат до рискове като нарушения на данните, манипулиране и разкриване на потребителска информация. При **изтичане на данни по невнимание**, поради лоши практики данните могат да изтичат към други приложения или лог файлове, което може да доведе до излагане на чувствителна информация, идентификационни данни. При **недостатъчно добра автентикация и/или оторизация**, слабите или несъществуващи механизми за удостоверяване могат да доведат до рискове като неоторизиран достъп до функционалността на приложението, потенциални нарушения на данните и представяне под чужда самоличност. При **инжектиране на данни в клиентската част на приложението**, в мобилното приложение могат да бъдат инжектирани злонамерени данни, което може да доведе до рискове като нарушения на данните, злонамерени пренасочвания и неоторизирани действия в рамките на приложението.

Инструменти за тестване на информационната сигурност и възможностите за пробиви

Разгледани са някои от най-популярните инструменти в основните категории, които са релевантни за разглежданата област на тестване на сигурността.

Скенери за уязвимости на уеб приложения

Qualys Web Application Scanning (WAS) улеснява идентифицирането и намаляването на уязвимостите в уеб приложенията. Извършва автоматизирани сканирания за откриване на уязвимости, проверява в известни бази данни за уязвимости и използва евристични техники за откриване на потенциални пропуски в сигурността. Той може да идентифицира уязвимости като разгледаните XSS, SQL инжектиране и CSRF и т.н. Има възможност и за ръчно тестване за проникване, като предоставя подробен списък на идентифицираните уязвимости.

Netsparker е инструмент за идентифициране на уязвимости и провеждане на автоматизирани тестове за проникване на уеб приложения.

Burp Suite е интегрирана платформа за извършване на тестове за сигурност на уеб приложения. Известен е със своята цялостна среда за ръчно тестване, заедно с възможностите за автоматизирано сканиране. Той е сил-

но предпочитан от тестерите за проникване заради своята интерактивна и практическа среда за тестване.

OWASP ZAP (Zed Attack Proxy) е водещ инструмент с отворен код в областта на сигурността на уеб приложенията. Той предоставя автоматизирани скенери и различни инструменти за ръчно тестване.

Автоматизирани скенери за уязвимости

Nessus е инструмент за сканиране на уязвимости. Този софтуер използва разнообразни техники и сигнатури, за да открие потенциални проблеми в мрежовите системи и уеб приложения. Nessus предоставя детайлни доклади и оценки за сериозност на уязвимостите.

Acunetix е мощен скенер за уеб уязвимости, който е особено полезен за разработчиците и експертите по сигурността, които искат да гарантират, че техните уеб приложения са защитени от атаки като SQL инжекции, XSS и други.

Netsparker е автоматизиран скенер за сигурност на уеб приложения, който е предназначен да открива и анализира уязвимости в уеб базирани системи.

Инструменти за експлоатиране на уязвимости

Metasploit е мощен инструмент за разработване, тестване и изпълнение на експлоатационен код срещу отдалечени целеви системи. Този инструмент позволява да се създават и изпълняват различни видове атаки и така да се оценява сигурността на системите.

SQL Injection инструменти

SQLMap е инструмент с отворен код, който автоматизира процеса на откриване и експлоатация на уязвимости, свързани със SQL инжекции.

XSS (Cross-Site Scripting) инструменти

XSSer е инструмент за тестване на проникване с отворен код, който автоматизира процеса на откриване и използване на XSS инжекции.

BeEF (Browser Exploitation Framework) е насочен към уеб браузърите и се фокусира върху уязвимостите от страна на клиента. Този инструмент позволява на атакуващите страни да използват уеб браузърите на жертвите си за изпълнение на различни видове атаки.

Инструменти за разбиване на пароли

John the Ripper е инструмент за разбиване на пароли, който използва методи като брутфорс и речникови атаки, за да създава комбинации от символи и да опитва да разбие паролите.

Hashcat е инструмент за възстановяване на пароли от налични хешове. Този софтуер е ефективен при разбиването им, а такива често могат да се получат от източени бази данни.

Инструменти за тестване на мрежи

Wireshark е широко използван анализатор на мрежови протоколи, който позволява на администраторите да записват и анализират мрежовия трафик.

Nmap: (Network Mapper) е инструмент с отворен код, който се използва за откриване на мрежи и одит на сигурността на устройствата в тях. Този софтуер предоставя информация за откритите портове и услуги, работещи на целевите системи и помага на администраторите да оценят тяхната сигурност.

Съществуват и множество други инструменти от най-разнообразно естество, но изброените дотук са едни от най-често използваните в контекста, който се разглежда тук. Важно е да се отбележи, че тези инструменти могат да помогнат за идентифициране на уязвимостите, но за анализ на констатациите и резултатите от инструментите и определяне на въздействието на потенциалните проблеми в реалния свят е необходим анализ от експерти в областта на киберсигурността, т.е. откриването на потенциални проблеми.

Подготовка на среда за извършване на тестване за уязвимости

Тестването на дадена система в контролирана среда често включва изграждането на среда, която възпроизвежда продукционната система, без да я засяга.

Първа стъпка е **създаване на имитационна среда**: клонира се продукционната система, създавайки точни копия на сървърите, базите данни и свързаната инфраструктура. Тази симулирана среда се изолира напълно от основната, като притежава свой собствен URL, база данни и, ако е възможно, собствена мрежова зона. Това може да се окаже сложен процес, когато става въпрос за системи, интегрирани с вътрешни и външни други системи. Вместо да се използват реални потребителски данни, приложението трябва да работи с обезличени или модифицирани версии. Различни инструменти и скриптове могат да автоматизират процеса на т. нар. **анонимизация**. Често не е необходимо използването на пълния обем данни от продукционната

среда; достатъчно е подмножество, което отразява разнообразието и сложността на истинските данни (освен ако тестването на производителността и реагиране на определени видове заявки не е важна част от тестовия процес). Следва **симулация на интеграции** – за връзки с услуги от трети страни (обикновено, но не само API) се използват имитации или „sandbox“ среди. За вътрешните интеграции могат да се използват т. нар. „stubs“ – опростени версии на услугите, които възпроизвеждат поведението на оригиналните, но без реални действия. Не винаги това е оптимално решение, тъй като е възможно е да бъде пропуснат потенциален проблем със сигурността, произтичащ именно от тези интеграции. **Конфигурацията на уеб приложението в тестовата среда** да е настроена така, че да се свързва с имитациите на услугите, а не с реалните. Механизмите за удостоверяване, като API ключове и пароли, трябва да са променени, за да се различават от тези в продукционната среда.

Защитната стена трябва да е настроена така, че тестовата среда да не може да се свързва с производствената или други важни вътрешни мрежи. Достъпът до тестовата среда е ограничен чрез механизми като VPN и IP бели списъци. Тестовата среда трябва да разчита на **свои инструменти за наблюдение**, независими от всякакви други, гарантирайки чистота на аналитичните данни. Нужна е по-голяма детайлност, което осигурява пълна видимост по време на тестовите процеси. Преди интензивни тестове се създават **автоматизирани моментни снимки на базата данни и системата**, позволявайки бързо възстановяване на стабилно състояние и евентуално – повтаряне на част от тестовете. Конфигурациите и скриптовете следва да са поддържани под **система за контрол на версиите**, осигурявайки последователност и история на промените. Следва да се опишат подробни **протоколи за тестовите процедури**, уточнявайки задачите и границите на всеки тест. Тестовата среда се актуализира редовно, за да отразява последните промени в продукционната – включително системни актуализации, изменения в приложението и конфигурационни промени.

Тестване на система Уеб-Студент на УНСС

Следвайки описаните методи за тестване, оценка и анализ на сигурността на университетски информационни системи, бяха проведени три моментни теста на сигурността в УНСС – на системата за прием на кандидат-студенти, на приложение за ръководителите на катедри, на системата „Уеб-Студент“. Тук е представена информация за проведеното тестване на последната от тях и резултатите от него, както и направените анализи и изводи.

За целите на тестване на сигурността използваме бета версия на нова версия на приложението „Уеб-Студент“ на Университета за национално и

световно стопанство – за уеб и като мобилно приложение за смартфони. Преди въвеждане на системата в експлоатация е проведен тест, чийто резултати са представени по-долу като пример за имплементация на описания подход. Система „Уеб-Студент“ се използва от студентите на УНСС и предоставя цялостна информация за тяхното следване, както и множество интерактивни услуги за тях – записване за семестър, плащане на семестриални такси, кандидатстване за общежития, за стипендии, за спорт, за избираеми дисциплини, издаване на уверения, дигитална студентска книжка, плащане на задължения по общежитията и други. Характерът на системата предполага достъп до чувствителни данни и висок приоритет при осигуряване на защитата ѝ.

Подход и инструменти

Съгласно описания по-горе подход, следва да: се **определи обхват на тестването** – в случая това е уеб приложение и мобилно приложение „Уеб-Студент“, което ще се използва от студентите на УНСС за преглед и управление на цялата им студентска информация и активности; следва да е сигурно, че няма възможност за неоторизиран достъп до данни и извършване на действия; сложността се допълва от тясната свързаност на системата с множество други информационни системи (студентски, за общежития, плащания с два различни виртуални POS терминала и др.); се **подготви среда за тестване** – поради споменатата обвързаност на „Уеб-Студент“ с множество други системи процесът по подготовка на среда за тестване е сложен процес, изискващ конфигуриране на сървъри и копия на бази данни, независими от използваните в продукционна среда; се **приоритизират потенциални слабости и уязвимости** – високата степен на интерактивни функции на системата води до потенциален риск от всички разгледани по-горе потенциални уязвимости на уеб и мобилните приложения; се **използват инструменти за тестване на сигурността** – в случая ще се използва един от най-популярните инструменти – BurpSuite.

BurpSuite вече беше споменат по-горе като един от най-популярните инструменти за тестване на сигурността в уеб приложенията. При този инструмент има следните особености при тълкуването на резултатите: оценка на сериозност и увереност: класифицира констатациите въз основа на тежестта (low, medium, high) и нивата на увереност (certain, firm, tentative). Това помага да се приоритизира решаването на проблемите; за всеки открит проблем предоставя подробно описание, местоположението на проблема и заявка/отговор, за да подчертае къде е идентифицирана уязвимостта; дават се предложения как да се коригират идентифицирани уязвимости, разбира се, това подлежи на интерпретация и адаптация към конкретната система; както всич-

ки автоматизирани скенери, BurpSuite може да има фалшиви положителни резултати. От съществено значение е анализаторите по сигурността да проверяват констатациите и ръчно, за да се уверят, че са автентични.

Резултати

Извършеното тестване на сигурността на система „Уеб-Студент“ с помощта на BurpSuite показа следните резултати (както беше споменато по-горе, тези тествания представляват моментна снимка на състоянието на системата и зависят от текущата версия на разработваната информационна система, както и множество други фактори, т.е. следва да се провеждат възможно най-често при промяна на контекста, за да са актуални), представени на фигура 2.

Към момента на тестване са открити 30 потенциални проблемни точки с ниско ниво на опасност (low severity), като 24 от тях са сигурни (certain) и 6 са почти сигурни (firm). Няма нито една потенциална уязвимост със средна (medium) или висока (high) опасност. Допълнително, системата е генерирала 38 информативни съобщения (“Information”), които не са проблеми, а помощна информация за хората, анализиращи резултатите.

		Confidence			
		Certain	Firm	Tentative	Total
Severity	High	0	0	0	0
	Medium	0	0	0	0
	Low	24	6	0	30
	Information	33	5	0	38
	False Positive	0	0	0	0

Източник: Отчет, генериран от BurpSuite, при тестване на софтуера

Фигура 2: Обобщени резултати от тестване с BurpSuite

Всяка една потенциална уязвимост е описана и класифицирана съгласно разгледания по-горе стандарт CWE (Common Weakness Enumeration).

Откритите потенциални проблеми и уязвимости с ниско ниво на опасност са следните: **Content type incorrectly stated (некоректно дефиниран тип на файлове)** – класифициран в категория CAPEC-63 – Cross-Site Scripting (XSS). Оказва се, че има няколко шрифта, за които сървърът не е предоставил информация за content-type, което би могло да доведе до про-

блем, ако тези шрифтове се качват от потребителите на системата. В случая това не е така, шрифтовете са фиксирани, поради което няма как да се реализира атака през този вектор. **Strict transport security not enforced (не е наложена стриктна транспортна сигурност)** – класифициран в категория CAPEC-157 – Sniffing Attacks. Уеб сървърът не е върнал към браузъра заглавка, изискваща задължително използване на HTTPS протокол за комуникация, поради което е възможно хакер, който преди това си е осигурил достъп до мрежата на потребителя, да подслушва и подмени неговия трафик и да го пренасочи към HTTP връзка, след което да може да подслушва некриптирания трафик между приложението и крайния потребител. Опасността от такова събитие е ниска, тъй като изисква атакуващият преди това да е получил достъп до мрежата. Въпреки това е препоръчително да се отрази в конфигурацията на приложението, като се добави Strict-Transport-Security header в отговорите.

Общият брой на потенциалните проблеми е 30, тъй като има 6 инстанции на първия проблем и 24 инстанции на втория (това зависи от броя тествани адреси).

Анализ и изводи

Проведеното тестване на сигурността за потенциални уязвимости на бета версия на система „Уеб-Студент“ разкрива главно проблеми с ниско ниво на опасност, като те са два типа, като в отчета са отбелязани като тридесет, заради тестването на различни модули и части на системата. Важно е да подчертаем, че няма установени уязвимости със средна или висока степен на опасност. Изследванията сочат някои конкретни проблеми, като некоректно определения тип на файлове за шрифтове и липсата на форсиране на задължително използване на HTTPS протокол. Въпреки че общата установена опасност е ниска, е препоръчително да се вземат мерки за нейното отстраняване, за да се подобри общата сигурност на системата. В допълнение към това, е важно периодично да се провеждат подобни тестове, за да се уверим, че системата остава безопасна с течение на времето и развитието на нови версии.

Заключение

Защитата на университетските информационни системи е от първостепенно значение предвид чувствителните данни, с които те работят. Процесът на тестване и оценка, макар и сложен, е от съществено значение за идентифициране на потенциални уязвимости и осигуряване на стабилна защита срещу заплахи. Като се придържат към систематичен подход, включващ

подготовка, събиране на информация и анализ на уязвимостите, могат да укрепят своите системи срещу кибер заплахи. Използването на добре структурирани методи за тестване, може значително да подпомогне подобряването на сигурността на тези критични системи.

Университетите често включват различни системи за редица административни, академични и изследователски нужди. Те могат да включват системи за управление на обучението, информационни системи за студенти, счетоводни системи, бази данни за научни изследвания и много други. Всяка система може да донесе свои собствени уязвимости и когато се интегрира, могат да се появят нови уязвимости поради слаби практики за интеграция, липса на стандартизирани протоколи за сигурност в системите или пренебрегнати съображения за сигурност по време на процеса. От съществено значение е да се извърши задълбочена оценка на сигурността, специално фокусирана върху тези интеграционни точки.

Спонсориране на научното изследване

Публикацията съдържа резултати от изследване, финансирано със средства от целева субсидия за НИД на УНСС по договор No. НИД НИ – 19/2023/A

Използвана литература

Bertoglio, D.D. and Zorzo, A.F. (2017). Overview and open issues on penetration test, Journal of the Brazilian Computer Society, 23(1), pp. 1-16.

Li, K., Chen, S., Fan, L., Feng, R., Liu, H., Liu, C., Liu, Y., and Chen, Y. (2023). Comparison and Evaluation on Static Application Security Testing (SAST) Tools for Java, ESEC/FSE 2023 Research Papers.

Severns, R. (2021). Dynamic Application Security Testing: Overview and Tooling, Stackhawk, available at: https://www.stackhawk.com/blog/dynamic-application-security-testing-overview/

Comparitech. (2023). What is IAST? (Interactive Application Security Testing) [online], available at: https://www.comparitech.com/net-admin/interactive-application-security-testing-iast/

Myers, B. (2021). Software Composition Analysis: Overview and Tooling Guide. Stackhawk. Available at: https://www.stackhawk.com/blog/software-composition-analysis-sca-overview-and-tooling-guide/

CIS. (2021). How Configuration Assessments Help Improve Cyber Defenses, CSO Online, available at: https://www.csoonline.com/article/3612398/how-configuration-assessments-help-improve-cyber-defenses.html

Alvarenga, G. (2023). Runtime Application Self-Protection (RASP), CrowdStrike, available at: https://www.crowdstrike.com/cybersecurity-101/cloud-security/runtime-application-self-protection-rasp/

Certitude Security. (2020). What Are Blue Teams and Blue Team Exercises? [online], available at: https://www.certitudesecurity.com/blog/analysis-and-assessments/what-are-blue-teams-and-blue-team-exercises/

Niezen, F., Eloff, J.H.P. and Solms, R. von. (2018). An approach to minimizing legal and reputational risk in Red Team exercises, Computers & Security, 78, pp. 35-47.

Ge, M., Bang, Y. and Kim, D. (2015). Continuous Monitoring and Assessment of Cybersecurity Risks in Large Organizations, in 48th Hawaii International Conference on System Sciences, pp. 5007-5016.

Christey, S. (2008). A Comparison of the CWE Development and Research Views [online], CWE – MITRE Corporation, available at: https://cwe.mitre.org/documents/views/view-comparison.html

Halfond, W. G., Viegas, J., and Orso, A. (2006). A Classification of SQL Injection Attacks and Countermeasures, in Proceedings of the IEEE International Symposium on Secure Software Engineering.

Kirda, E., Kruegel, C., Vigna, G., Jovanovic, N., and Balduzzi, M. (2009). Noxes: a client-side solution for mitigating cross-site scripting attacks, in Proceedings of the ACM Symposium on Applied Computing.

Barth, A., Jackson, C., and Mitchell, J. C. (2008). Robust defenses for cross-site request forgery, in Proceedings of the 15th ACM conference on Computer and communications security.

***

METHODS FOR TESTING, ASSESSMENT AND ANALYSIS OF THE SECURITY OF UNIVERSITY INFORMATION SYSTEMS

Assist. Prof. Venko Andonov, PhD
Computer Science Department
Applied Informatics and Statistics Faculty
University of National and World Economy
e-mail: vandonov@unwe.bg

Abstract

The methods for testing, evaluating and analyzing the security of university information systems aim to build a unified framework that leads to an increase in information security in this type of institution, which is characterized by the administration of a huge volume of sensitive information of a diverse nature. Security testing is a multifaceted process involving steps to ensure the safety of systems in their entirety. The proposed methods have been tested with three systems of the University of National and World Economy, and one of them is presented here; and conclusions are drawn to increase security and include the described procedures as a permanent practice.

Keywords: information security, cybersecurity, security testing, software, vulnerabilities

JEL: I23, M15