สำนักงานคณะกรรมการการรักษาความมั่นคง
ปลอดภัยไซเบอร์แห่งชาติ

WEBSITE SECURITY

WEBSITE SECURITY

เว็บไซต์ถูกใช้เป็นเครื่องมือที่สำคัญของหน่วยงานต่างๆในการสื่อสาร ประชาสัมพันธ์ หรือให้บริการออนไลน์ต่างๆ กับผู้ใช้งานผ่านเครือข่ายอินเทอร์เน็ตสาธารณะ ด้วยลักษณะของบริการเว็บไซต์ที่เปิดให้ผู้ใช้งานสามารถเข้าถึงได้อยู่ตลอดเวลา ทำให้บริการเว็บไซต์ที่ไม่มีการรักษาความมั่นคงปลอดภัยที่ดีมีความเสี่ยงจากการถูกโจมตีจากผู้ไม่ประสงค์ดีได้อยู่ตลอดเวลาเช่นกัน

โดยภัยคุกคามที่มักจะเกิดขึ้นกับบริการเว็บไซต์ มีดังนี้

- การโจมตีเว็บไซต์เพื่อเปลี่ยนแปลงข้อมูลเผยแพร่หน้าเว็บ (Website Defacement)
- การโจมตีแบบปฏิเสธการให้บริการ (DoS)
- การเข้าถึงข้อมูลส่วนบุคคลของผู้ใช้งาน หรือองค์กร
- ผู้โจมตีเข้าควบคุมเว็บไซต์ที่ได้รับผลกระทบ
- การใช้เว็บไซต์ในการโจมตีช่องโหว่อื่น ๆ

ภัยคุกคามเหล่านี้ส่งผลกระทบต่อความปลอดภัยของข้อมูลในทุกด้าน เช่น ความลับ ความสมบูรณ์ และความพร้อมใช้งาน และอาจสร้างความเสียหายอย่างร้ายแรงต่อชื่อเสียงของเว็บไซต์และเจ้าของเว็บไซต์ ตัวอย่างเช่น เว็บไซต์ขององค์กรและส่วนบุคคลที่ตกเป็นเหยื่อของการทำให้เสียโฉม, DoS, หรือการละเมิดข้อมูลอาจประสบกับความสูญเสียทางการเงินเนื่องจากความไว้วางใจที่ลดลงหรือผู้เข้าชมเว็บไซต์ลดลง

WEBSITE SECURITY guideline

หัวข้อนี้จัดทำขึ้นโดยประยุกต์ใช้หลักการรักษาความมั่นคงปลอดภัยไซเบอร์เว็บไซต์ตาม Website Security โดย Cybersecurity & Infrastructure Security Agency และ Web Security จาก mozilla เพื่อให้หน่วยงานของรัฐและหน่วยงานเอกชนสามารถมีแนวทางในการยกระดับการรักษาความมั่นคงปลอดภัยไซเบอร์ให้กับเว็บไซต์ได้อย่างมีประสิทธิภาพ

https://www.cisa.gov/uscert/ncas/tips/ST18-006
https://infosec.mozilla.org/guidelines/web_security#web-security-cheat-sheet

แนวทางปฏิบัติในการรักษาความมั่นคงปลอดภัยไซเบอร์เว็บไซต์

1 ทำให้ระบบนิเวศของโดเมนมีความปลอดภัย (Secure domain ecosystems)

- ตรวจสอบผู้รับจดทะเบียน (Registrar) และระเบียนระบบชื่อโดเมน (DNS records) สำหรับโดเมนทั้งหมด
- เปลี่ยนรหัสผ่านเริ่มต้นทั้งหมดที่ได้รับจากผู้รับจดทะเบียนโดเมนและ DNS ของคุณ

  ชื่อผู้ใช้และรหัสผ่านเริ่มต้นไม่ปลอดภัย - โดยปกติแล้วจะสามารถค้นหาได้บนอินเทอร์เน็ต การเปลี่ยนชื่อผู้ใช้และรหัสผ่านเริ่มต้นจะป้องกันการโจมตีที่ใช้ประโยชน์จากข้อมูลดังกล่าว
- บังคับใช้การพิสูจน์และยืนยันตัวตนหลายปัจจัย (Multi-factor authentication : MFA)
- ติดตามตรวจสอบบันทึกความโปร่งใสของใบรับรอง (certificate transparency logs) เพื่อตรวจสอบเว็บไซต์ปลอมที่อาจใช้ชื่อโดเมนคล้ายกัน

แนวทางปฏิบัติในการรักษาความมั่นคงปลอดภัยไซเบอร์เว็บไซต์

2 ทำให้บัญชีผู้ใช้มีความปลอดภัย (Secure user accounts)

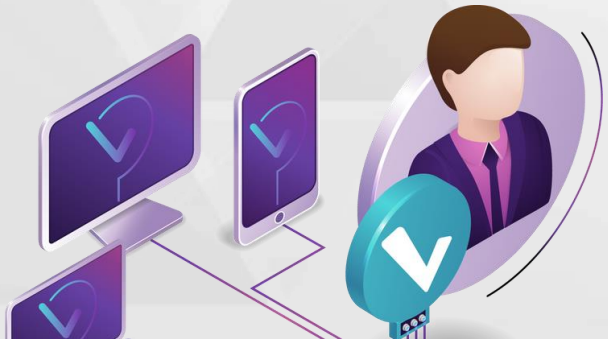
- บังคับใช้การพิสูจน์และยืนยันตัวตนหลายปัจจัยกับบัญชีผู้ใช้ทั้งหมดที่เข้าถึงได้จากอินเทอร์เน็ต โดยให้ความสำคัญกับบัญชีผู้ใช้ที่มีสิทธิ์สูง (Privileged access)
- จำกัดสิทธิ์ของผู้ใช้เท่าที่จำเป็น (Principle of least privilege) รวมทั้งปิดใช้งานบัญชีและสิทธิ์ที่ไม่จำเป็น
- เปลี่ยนชื่อผู้ใช้และรหัสผ่านเริ่มต้นทั้งหมด

แนวทางปฏิบัติในการรักษาความมั่นคงปลอดภัยไซเบอร์เว็บไซต์

3 ค้นหาและแก้ไขช่องโหว่อย่างต่อเนื่อง (Continuously scan for – and remediate – critical and high vulnerabilities)

- แก้ไขช่องโหว่ระดับวิกฤต (critical) และระดับสูง (high) ทั้งหมด ภายใน 15 และ 30 วัน ตามลำดับ ในระบบที่สามารถเข้าถึงได้ทางอินเทอร์เน็ต โดยสแกนหาช่องโหว่จากการตั้งค่า (Configuration vulnerabilities) และช่องโหว่ของซอฟต์แวร์ รวมถึงเปิดใช้งานการอัปเดตอัตโนมัติทุกครั้ง

- เปลี่ยนระบบปฏิบัติการ แอปพลิเคชันและฮาร์ดแวร์ ซึ่งหมดระยะเวลาการสนับสนุนโดยผู้ผลิต



แนวทางปฏิบัติในการรักษาความมั่นคงปลอดภัยไซเบอร์เว็บไซต์

4 ทำให้ข้อมูลระหว่างการขนส่งมีความปลอดภัย (Secure data in transit)

- ปิดการใช้งาน Hypertext Transfer Protocol (HTTP); บังคับใช้ Hypertext Transfer Protocol Secure (HTTPS) และ HTTP Strict Transport Security (HSTS) ผู้ใช้งานเว็บไซต์คาดหวังว่าความเป็นส่วนตัวของพวกเขาจะได้รับการปกป้อง ดังนั้นเพื่อให้แน่ใจว่าการสื่อสารระหว่างเว็บไซต์และผู้ใช้ได้รับการเข้ารหัส จึงต้องบังคับใช้ HTTPS เสมอ และควรบังคับใช้ HSTS หากเป็นไปได้

- ปิดการใช้งานอัลกอริทึมที่ไม่มีความมั่นคงปลอดภัย เช่น SSLv2, SSlv3, 3DES, RC4

แนวทางปฏิบัติในการรักษาความมั่นคงปลอดภัยไซเบอร์เว็บไซต์

5 สำรองข้อมูล (Backup data)

- หมั่นสำรองข้อมูลอย่างสม่ำเสมอ โดยเฉพาะข้อมูลและระบบที่สำคัญเว็บไซต์
- เก็บข้อมูลสำรองไว้ในที่ปลอดภัยและแยกทางกายภาพออกจากระบบ (Physically remote environment)
- ทดสอบการกู้คืนข้อมูล

แนวทางปฏิบัติในการรักษาความมั่นคงปลอดภัยไซเบอร์เว็บไซต์

6 ทำให้เว็บแอปพลิเคชันมีความปลอดภัย (Secure web applications)

- ระบุและลดความเสี่ยงของการถูกโจมตีบนเว็บแอปพลิเคชันที่สำคัญ 10 อันดับแรก (OWASP) จากนั้นจึงพิจารณาระบุและลดความเสี่ยงอื่น ๆ เป็นลำดับถัดไป อาทิ การปฏิบัติที่ดีที่สุดในการรักษาความปลอดภัยของระบบและเครือข่ายที่มีการใช้งานเทคโนโลยีอินเทอร์เน็ต จาก Center for Internet Security (CISX) เปิดการบันทึกการใช้งาน (Logging) และตรวจสอบบันทึกการใช้งานกิจกรรมบนเว็บไซต์อย่างสม่ำเสมอ เพื่อตรวจหากิจกรรมที่ผิดปกติ ทั้งนี้ ควรส่งบันทึกการใช้งานดังกล่าวไปยังเซิร์ฟเวอร์ส่วนกลาง (Centralized log server)

- ใช้การพิสูจน์และยืนยันตัวตนหลายปัจจัยสำหรับผู้ใช้งาน ซึ่งล็อกอินเข้าสู่เว็บแอปพลิเคชัน รวมถึงการใช้งานโครงสร้างพื้นฐานของเว็บไซต์ด้วย

https://owasp.org/www-project-top-ten/)

https://www.cisecurity.org/controls

แนวทางปฏิบัติในการรักษาความมั่นคงปลอดภัยไซเบอร์เว็บไซต์

7 ทำให้เว็บเซิร์ฟเวอร์มีความปลอดภัย (Secure web servers)

- ตรวจสอบความปลอดภัยโดยอ้างอิงตาม security checklist ของแอปพลิเคชันนั้น ๆ เช่น Apache, MySQL และดำเนินการตั้งค่าให้มีความมั่นคงปลอดภัย (harden configurations)
- ใช้แอปพลิเคชันที่เชื่อถือได้และมีความจำเป็นสอดคล้องกับความต้องการขององค์กร รวมทั้งปิดการใช้งานฟีเจอร์ที่ไม่จำเป็น
- ใช้การแบ่งส่วนและแยกเครือข่าย (network segmentation and segregation)
  - การแบ่งส่วนและแยกเครือข่ายจะทำให้ผู้โจมตีเคลื่อนไหว (move laterally) ในเครือข่ายได้ยากขึ้น ตัวอย่างเช่น การวางเว็บเซิร์ฟเวอร์ไว้บนเครือข่ายแบบ demilitarized zone (DMZ) ที่ถูกกำหนดค่าอย่างเหมาะสมจะจำกัดการรับส่งข้อมูลที่ได้รับอนุญาตระหว่างระบบภายในเครือข่ายแบบ DMZ และระบบในเครือข่ายภายในขององค์กร
- ให้คำนึงไว้เสมอว่าทรัพย์สินอยู่ที่ไหน
  - ถ้าเรารู้ว่าข้อมูลสำคัญของเราอยู่ตรงไหน เราก็จะสามารถบริหารจัดการและจำกัดการเข้าถึงได้อย่างเหมาะสม

แนวทางปฏิบัติในการรักษาความมั่นคงปลอดภัยไซเบอร์เว็บไซต์

8 จัดลำดับความสำคัญโดยใช้ Web Security Cheat Sheet

การดำเนินการเพื่อให้เว็บไซต์มีความมั่นคงปลอดภัยนั้นสามารถดำเนินการได้หลายประการ ทั้งนี้ ควรพิจารณาจากประโยชน์ที่จะได้รับและความยากง่ายในการดำเนินการ ซึ่งจะทำให้องค์กรสามารถจัดลำดับความสำคัญในการดำเนินการได้

Guideline	Security Benefit	Implementation Difficulty	Order†	Requirements	Notes
HTTPS	MAXIMUM	MEDIUM		Mandatory	Sites should use HTTPS (or other secure protocols) for all communications
Public Key Pinning	LOW	MAXIMUM	--	Mandatory for maximum risk sites only	Not recommended for most sites
Redirections from HTTP	MAXIMUM	LOW	3	Mandatory	Websites must redirect to HTTPS, API endpoints should disable HTTP entirely
Resource Loading	MAXIMUM	LOW	2	Mandatory for all websites	Both passive and active resources should be loaded through protocols using TLS, such as HTTPS
Strict Transport Security	HIGH	LOW	4	Mandatory for all websites	Minimum allowed time period of six months
TLS Configuration	MEDIUM	MEDIUM	1	Mandatory	Use the most secure Mozilla TLS configuration for your user base, typically Intermediate
Content Security Policy	HIGH	HIGH	10	Mandatory for new websites Recommended for existing websites	Disabling inline script is the greatest concern for CSP implementation

Web Security Cheat Sheet

Guideline	Security Benefit	Implementation Difficulty	Order†	Requirements	Notes
HTTPS	MAXIMUM	MEDIUM		Mandatory	Sites should use HTTPS (or other secure protocols) for all communications
Public Key Pinning	LOW	MAXIMUM	--	Mandatory for maximum risk sites only	Not recommended for most sites
Redirections from HTTP	MAXIMUM	LOW	3	Mandatory	Websites must redirect to HTTPS, API endpoints should disable HTTP entirely
Resource Loading	MAXIMUM	LOW	2	Mandatory for all websites	Both passive and active resources should be loaded through protocols using TLS, such as HTTPS
Strict Transport Security	HIGH	LOW	4	Mandatory for all websites	Minimum allowed time period of six months
TLS Configuration	MEDIUM	MEDIUM	1	Mandatory	Use the most secure Mozilla TLS configuration for your user base, typically Intermediate
Content Security Policy	HIGH	HIGH	10	Mandatory for new websites Recommended for existing websites	Disabling inline script is the greatest concern for CSP implementation
Cookies	HIGH	MEDIUM	7	Mandatory for all new websites Recommended for existing websites	All cookies must be set with the Secure flag, and set as restrictively as possible
contribute.json	LOW	LOW	9	Mandatory for all new Mozilla websites Recommended for existing Mozilla sites	Mozilla sites should serve contribute.json and keep contact information up-to-date
Cross-origin Resource Sharing	HIGH	LOW	11	Mandatory	Origin sharing headers and files should not be present, except for specific use cases
Cross-site Request Forgery Tokenization	HIGH	UNKNOWN	6	Varies	Mandatory for websites that allow destructive changes Unnecessary for all other websites Most application frameworks have built-in CSRF tokenization to ease implementation
Referrer Policy	LOW	LOW	12	Recommended for all websites	Improves privacy for users, prevents the leaking of internal URLs via Referer header
robots.txt	LOW	LOW	14	Optional	Websites that implement robots.txt must use it only for noted purposes
Subresource Integrity	MEDIUM	MEDIUM	15	Recommended‡	‡ Only for websites that load JavaScript or stylesheets from foreign origins

หมายเหตุ : ลำดับถูกจัดเรียงตามผลกระทบที่อาจเกิดขึ้นและความง่ายในการจัดการ

Web Security Cheat Sheet

1 - Transport Layer Security (TLS/SSL)

Guideline	Security Benefit	Implementation Difficulty	Order†	Requirements	Notes
TLS Configuration	MEDIUM	MEDIUM	1	Mandatory	Use the most secure Mozilla TLS configuration for your user base, typically Intermediate

Transport Layer Security (TLS) เป็นโปรโตคอลความปลอดภัยที่เข้ารหัสอีเมลเพื่อความเป็นส่วนตัว โดยป้องกันการเข้าถึงอีเมลของคุณโดยไม่ได้รับอนุญาตเมื่อส่งอีเมลผ่านการเชื่อมต่ออินเทอร์เน็ต ช่วยทำให้ผู้ใช้มั่นใจได้ว่าคนที่ใช้งานฝั่งของ Server หรือ Website เป็นคนๆ นั้นจริง ๆ และก็ยังช่วยป้องกันการปลอมแปลงข้อมูล

Compatibility

Configuration	Oldest compatible clients
Modern	Firefox 63, Android 10.0, Chrome 70, Edge 75, Java 11, OpenSSL 1.1.1, Opera 57, and Safari 12.1
Intermediate	Firefox 27, Android 4.4.2, Chrome 31, Edge, IE 11 on Windows 7, Java 8u31, OpenSSL 1.0.1, Opera 20, Safari 9
Backwards Compatible (Old)	Firefox 1, Android 2.3, Chrome 1, Edge 12, IE8 on Windows XP, Java 6, OpenSSL 0.9.8, Opera 5, and Safari 1

Web Security Cheat Sheet

NCSA สกมช

2 - Resource Loading

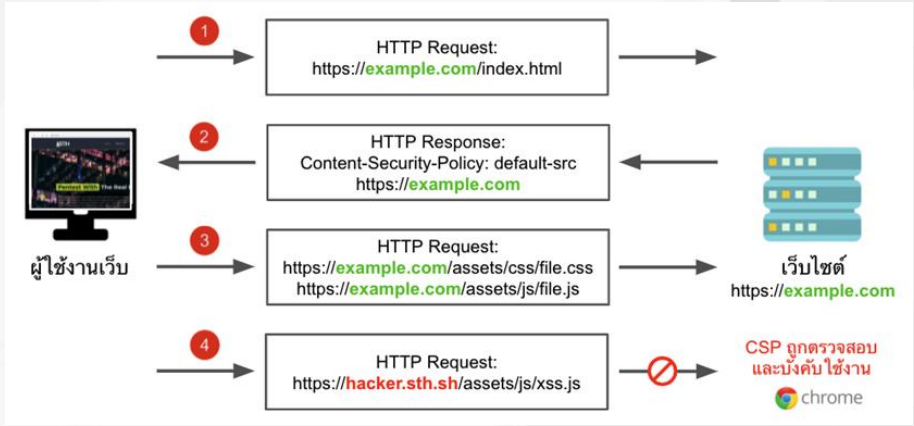
Guideline	Security Benefit	Implementation Difficulty	Order†	Requirements	Notes
Resource Loading	MAXIMUM	LOW	2	Mandatory for all websites	Both passive and active resources should be loaded through protocols using TLS, such as HTTPS

Know what you load, know when you load

Web Security Cheat Sheet

3 - Content Security Policy

เป็นส่วนที่ช่วยให้การกำหนดต้นทางของเนื้อหาที่อนุญาตสำหรับเว็บไซต์ โดยการจำกัดเนื้อหาที่เบราเซอร์สามารถโหลดได้ ได้แก่ js และ css.ซึ่งจะช่วยในการลดความเสี่ยงการถูกโจมตีด้วยช่องโหว่แนว Client Side อย่าง XSS



Guideline	Security Benefit	Implementation Difficulty	Order†	Requirements	Notes
Content Security Policy	HIGH	HIGH	10	Mandatory for new websites Recommended for existing websites	Disabling inline script is the greatest concern for CSP implementation

https://www.cyfence.com/article/how-to-set-up-a-content-security-policy-for-website/

Web Security Cheat Sheet

ตัวอย่างการตั้งค่า CSP ของ

https://www.twitter.com

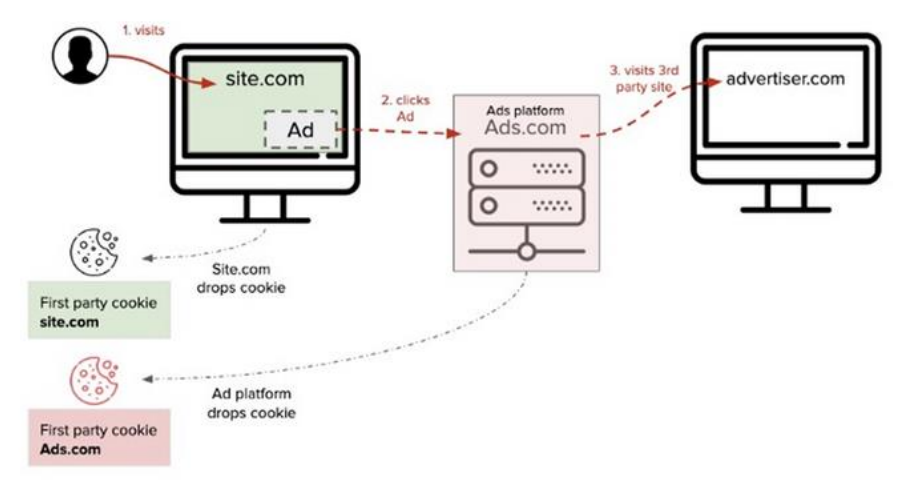
Evaluated CSP as seen by a browser supporting CSP Version 3 — expand/collapse all

- ✓ connect-src
- ✓ default-src
- ✓ form-action
  - ✓ 'self'
  - ✓ https://twitter.com
  - ✓ https://*.twitter.com
- ✓ font-src
- ✓ frame-src
- ✓ img-src
- ✓ manifest-src
- ✓ media-src
- ✓ object-src
- ✓ script-src — Host whitelists can frequently be bypassed. Consider using 'strict-dynamic' in combination with CSP nonces or hashes.
- ✓ style-src
- ✓ worker-src
- ✓ base-uri [missing] — Missing base-uri allows the injection of base tags. They can be used to set the base URL for all relative (script) URLs to an attacker controlled domain. Can you set it to 'none' or 'self'?
- ⓘ require-trusted-types-for [missing] — Consider requiring Trusted Types for scripts to lock down DOM XSS injection sinks. You can do this by adding "require-trusted-types-for 'script'" to your policy.

Directive	Value	Source	คำอธิบาย
default-src	'self'	–	นโยบายของ CSP จะมีแยกย่อยหลาย Directive ถ้าหากอันไหนไม่ได้ถูกกำหนดไว้ จะมาอ้างอิงจากนโยบายของ default-src เป็นค่าเริ่มต้น (ถ้าถูกกำหนดไว้ก็จะใช้ของแต่ละอันนั้น ๆ ไปแทน) โดยค่าเริ่มต้นของนโยบายในตัวอย่างนี้คือ self แปลว่ายอมให้นำเนื้อหาเฉพาะจากเว็บเดียวกันเข้ามาใช้งานได้
connect-src	'self' blob:	https://api.twitter.com [...]	จำกัดให้สามารถเรียกใช้งาน API ภายนอกได้เฉพาะจาก URL รูปแบบที่กำหนดเท่านั้น เช่น https://api.twitter.com และ เนื้อหารองรับรูปแบบ Blob
form-action	'self'	https://twitter.com https://*.twitter.com;	สามารถใช้ HTML tag อย่าง <form> ในการส่งข้อมูลไปที่เว็บเดียวกันและเว็บที่เป็น Subdomain เท่านั้นตามรูปแบบ *.twitter.com
frame-src	'self'	https://twitter.com https://mobile.twitter.com [...] https://www.gstatic.com/recaptcha/;	สามารถนำเว็บอื่นมาแสดงผลผ่าน HTML tag <iframe> จาก Domain ที่กำหนดเท่านั้น เช่น https://mobile.twitter.com ถ้าหากเว็บไซต์พยายามจะนำเว็บอื่นนอกเหนือจากที่กำหนดมาแสดงผลผ่าน <iframe> จะไม่สามารถทำได้
img-src	'self' blob: data:	https://*.cdn.twitter.com [...] https://imgix.revue.co;	สามารถนำ URL ของรูปจาก Domain ที่กำหนดไว้มาแสดงได้ในเว็บไซต์เท่านั้น เช่น https://demo.cdn.twitter.com/demo.jpg จะสามารถนำมาแสดงได้แต่ https://*.cdn.hacker.com/hack.jpg จะไม่สามารถนำมาแสดงบนเว็บได้
script-src	'self' 'unsafe-inline' 'nonce-MmI4YzMzNmYtMzY2YS00NjM3LTg5NGYtNGI3NWQwMTcwNDcw';	https://*.twimg.com [...]	กำหนดให้เว็บไซต์สามารถรันคำสั่ง JavaScript ในรูปแบบไหน และจากที่ไหนได้บ้าง จากตัวอย่างของ Twitter.com คือ ยอมให้รัน JavaScript แบบ Inline (อยู่ในเอกสาร HTML) ได้ แต่ต้องมีการสุ่มค่า (Nonce) แบบใช้ครั้งเดียว ใส่ในแต่ละ HTTP Request กับใน <script> เช่น <script nonce="${RANDOM}">[...]</script> และตรวจสอบว่าเป็นค่าเดียวกันเสมอ สามารถรัน JavaScript จากลิงก์ภายนอกได้แต่ต้องมาจาก URL ที่กำหนดเท่านั้นเช่น https://cdn.twimg.com/file.js

Web Security Cheat Sheet

4 - contribute.json



"first-party bounce trackers"

Guideline	Security Benefit	Implementation Difficulty	Order†	Requirements	Notes
contribute.json	LOW	LOW	9	Mandatory for all new Mozilla websites Recommended for existing Mozilla sites	Mozilla sites should serve contribute.json and keep contact information up-to-date

แม้ว่าจะบล็อก third-party cookie จาก เบราว์เซอร์ แต่คุณคลิกที่แอดใน Domain ที่คุณกำลังใช้งาน third-party web จะสร้าง first-party cookie ไปเก็บไว้ในคอมพิวเตอร์ของคุณแทน third-party web ก็จะยังสามารถติดตามคุณจากตรงนั้น

Web Security Cheat Sheet

5 - Cookies

Cookie เป็นไฟล์ข้อความ (Text File) ขนาดเล็กที่ถูกเก็บบันทึกเอาไว้ในคอมพิวเตอร์ของเรา ไฟล์ Cookie จะถูกสร้างขึ้นโดยอัตโนมัติ เมื่อเราใช้เว็บเบราว์เซอร์เข้าชมเว็บไซต์ต่าง ๆ

โดย Cookie จะช่วยให้ประสบการณ์ในการใช้อินเทอร์เน็ตของผู้ใช้ได้รับความสะดวกสบายมากยิ่งขึ้น โดยทำหน้าที่จดจำข้อมูลที่เป็นประโยชน์ต่อตัวเว็บไซต์ ตัวอย่างเช่น หากเราเข้าเว็บขายสินค้าออนไลน์ แล้วเรามีสินค้าใส่เอาไว้ในตะกร้า หากไม่มี Cookie เมื่อเราปิดหน้าเว็บไป สินค้าในตะกร้าก็จะหายไปด้วย

Guideline	Security Benefit	Implementation Difficulty	Order†	Requirements	Notes
Cookies	HIGH	MEDIUM	7	Mandatory for all new websites Recommended for existing websites	All cookies must be set with the Secure flag, and set as restrictively as possible

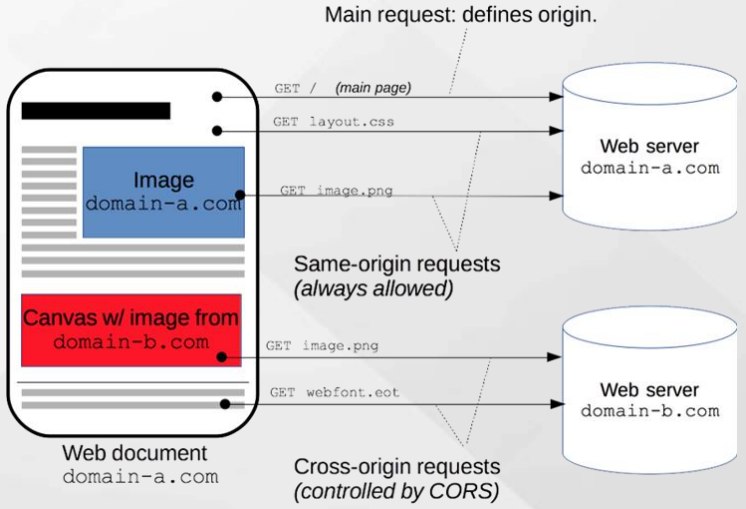
Web Security Cheat Sheet

6 Cross-origin Resource Sharing

คือ เป็นกลไกที่ใช้ HTTP headers เพิ่มเติมเพื่อให้บราวเซอร์ได้รับสิทธิ์ในการเข้าถึงทรัพยากรที่เลือกจากเซิร์ฟเวอร์บนโดเมนอื่นมาแสดงบนหน้าเว็บบราวเซอร์ได้

อินเตอร์เน็ตเป็นการสื่อสารระหว่างคอมพิวเตอร์ ดังนั้นคอมพิวเตอร์แต่ละเครื่องต้องมี Protocol ที่เหมือนกัน ถึงจะสื่อสารกันรู้เรื่อง เว็บบราวเซอร์จะส่ง HTTP request เมื่อต้องการขอข้อมูลข้ามโดเมนหรือ port ที่ต่างกัน และต้องทำตามข้อตกลงการสื่อสาร(Protocol)

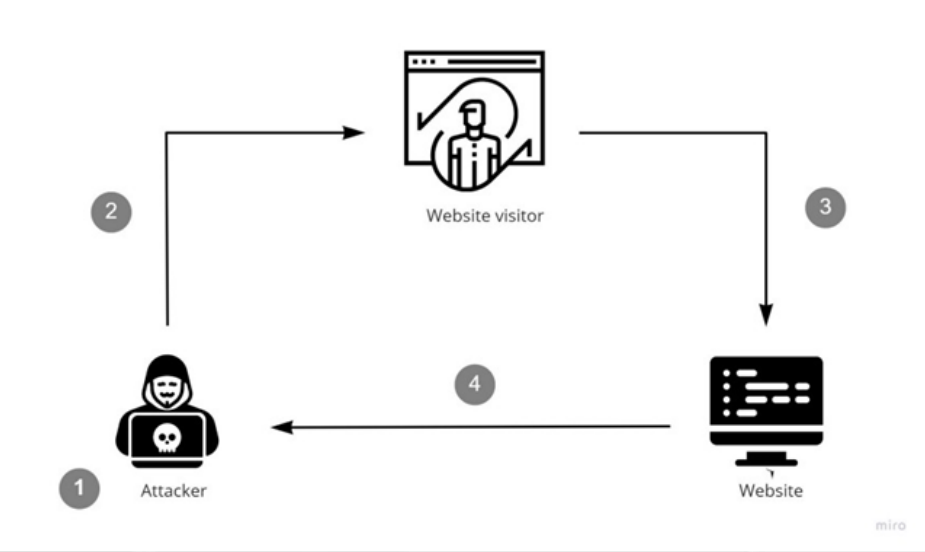
Guideline	Security Benefit	Implementation Difficulty	Order†	Requirements	Notes
Cross-origin Resource Sharing	HIGH	LOW	11	Mandatory	Origin sharing headers and files should not be present, except for specific use cases



Web Security Cheat Sheet

7 - CSRF Prevention

CSRF เป็นช่องโหว่ที่ Attakcer ส่ง HTML หรือ JavaScript ให้ Web browser ของเหยื่อส่ง HTTP request เพื่อไปกระทำการบางอย่างที่เป็นอันตรายต่อผู้ใช้งาน



Guideline	Security Benefit	Implementation Difficulty	Order†	Requirements	Notes
Cross-site Request Forgery Tokenization	HIGH	UNKNOWN	6	Varies	Mandatory for websites that allow destructive changes Unnecessary for all other websites Most application frameworks have built-in CSRF tokenization to ease implementation

ตัวอย่างเช่น Request GET สำหรับการโอนเงินผ่านธนาคาร $100 อาจมีลักษณะดังนี้:

```
GET http://netbank.com/transfer.do?acct=PersonB&amount=$100 HTTP/1.1
```

Attacker สามารถแก้ไข script นี้ จะทำให้มีการโอนเงิน $100 ไปยังบัญชีของตนเอง ตอนนี้ request ที่เป็นอันตรายอาจมีลักษณะดังนี้:

```
GET http://netbank.com/transfer.do?acct=AttackerA&amount=$100
HTTP/1.1
```

Attacker สามารถฝัง request ลงใน hyperlink ที่ดูน่าเชื่อถือแบบนี้:

```
<a href="http://netbank.com/transfer.do?
acct=AttackerA&amount=$100">Read more!</a>
```

Web Security Cheat Sheet

8 - Referrer Policy

Guideline	Security Benefit	Implementation Difficulty	Order†	Requirements	Notes
Referrer Policy	LOW	LOW	12	Recommended for all websites	Improves privacy for users, prevents the leaking of internal URLs via Referer header

เมื่อเราต้องการขับ Traffic ออกจาก Website ของเราด้วย Links
ตัว Protocol จะเปิดเผยข้อมูลบางส่วนจาก Referring Site
โดย Referrer-Policy Feature จะช่วยให้เราจัดการว่า เราควรจะรวมเอา Information อยู่ใน Outbound Requests เหล่านั้นมากน้อยแค่ไหน

Directives

- `no-referrer` : never send the `Referer` header
- `same-origin` : send referrer, but only on requests to the same origin
- `strict-origin` : send referrer to all origins, but only the URL without the path (e.g. https://example.com/)
- `strict-origin-when-cross-origin` : send full referrer on same origin, URL without the path on foreign origin

Referrer-Policy Feature มีอยู่ 3 ส่วน:

•เป็น meta tag:

```
<meta name="referrer" content="origin">
```

•เป็น HTTP header:

```
Referrer-Policy: no-referrer
```

•เป็นใน *a* tag:

```
<a href="http://example.com" referrerpolicy="origin">
```

เราสามารถ Config. ค่าของ Information ได้อย่างละเอียดด้วย no-referrer, no-referrer-when-downgrade, origin, origin-when-cross-origin, same-origin, strict-origin, strict-origin-when-cross-origin และ unsafe-url

Web Security Cheat Sheet

9 - robots.txt

Guideline	Security Benefit	Implementation Difficulty	Order†	Requirements	Notes
robots.txt	LOW	LOW	14	Optional	Websites that implement robots.txt must use it only for noted purposes

ตัวอย่างการสร้างไฟล์ robots.txt

User-agent: Googlebot

Disallow: /admin-area/ //ไม่อนุญาตให้เข้าถึงโฟลเดอร์ wp-admin

Allow: /shop/cloth/ //อนุญาตให้เข้า Folder หรือ Directory นี้ได้

Sitemap: // จะเรียกตำแหน่ง XML ของเว็บที่เชื่อมโยงกับ URL ที่ระบุไว้

ไฟล์ Robots.txt จะทำหน้าที่อนุญาตและยกเว้นการเข้าถึงไฟล์และโฟลเด้อต่าง ๆ ที่อยู่บน web server ให้กับ web robots ซึ่งเป็นโปรแกรมรวมรวมข้อมูลเว็บไซต์ (Crawlers หรือ Spider) ซึ่งถูกรันแบบอัตโนมัติจากหลากหลายแหล่งที่มา และเพื่อความเป็นส่วนตัวของข้อมูลบนโลกอินเตอร์เน็ต ซึ่งอาจจะมีข้อมูลบางอย่างบนเว็บไซต์ที่เราไม่ต้องการให้ robots เหล่านี้นำไปทำ index หรือทำอย่างอื่น จึงเกิดไฟล์ robots.txt ขึ้นมาเพื่อบอกให้ robots เหล่านั้นรู้ว่า directory ส่วนไหน หรือไฟล์ไหนบนเว็บไซต์ของเรา ที่สามารถนำไปทำ index ได้และไฟล์ไหนไม่อนุญาติให้นำไปสร้าง index

Web Security Cheat Sheet

10 - Subresource Integrity

เป็นฟีเจอร์ความปลอดภัยที่จะเป็นตัวช่วยให้ Browser สามารถตรวจสอบไฟล์ Scripts หรือ Stylesheets นั้นๆ ได้ว่าไม่ถูกแก้ไขไปจากไฟล์ต้นฉบับเหมาะสำหรับผู้ที่เป็นเจ้าของเว็บไซต์, นักพัฒนา, ผู้ให้บริการ CDN หรือ Third Party Hosts โดย SRI จะเป็นตัวช่วยในการยืนยัน Source ไฟล์ที่ถูกดึงมาใช้จาก Third Party ทั้งหลายว่าเป็นไฟล์ต้นฉบับ ไม่ได้ถูกแก้ไขโดยผู้ไม่หวังดี ดังนั้น เพื่อเป็นการยืนยันข้อมูลกับ Browser เราจึงต้องมี flag บางอย่างไว้ให้ Browser ไว้คอยตรวจสอบโดย flag ที่ว่านี้ ก็คือ แอตทริบิวต์ที่ชื่อ integrity นั่นเอง

Guideline	Security Benefit	Implementation Difficulty	Order†	Requirements	Notes
Subresource Integrity	MEDIUM	MEDIUM	15	Recommended‡	‡ Only for websites that load JavaScript or stylesheets from foreign origins

วิธีการใช้งานตัว SRI คือ โค้ดที่เราใช้เรียก CDN จะมีค่า integrity="..." มาให้ด้วย ค่านี้เป็นตัวหนังสือที่เข้ารหัสของไฟล์ตัวจริง ถ้าเกิด CDN ส่งไฟล์มาแล้วไม่ตรงกับตัวหนังสือเข้ารหัส แปลว่า CDN นั้นแถมโค้ดแปลก ๆ มาด้วย บราวเซอร์ก็จะไม่โหลดไฟล์เพื่อความปลอดภัยนั่นเอง

```
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstra
p.min.css" integrity="sha384-
MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO"
crossorigin="anonymous">
```

Web Security Cheat Sheet

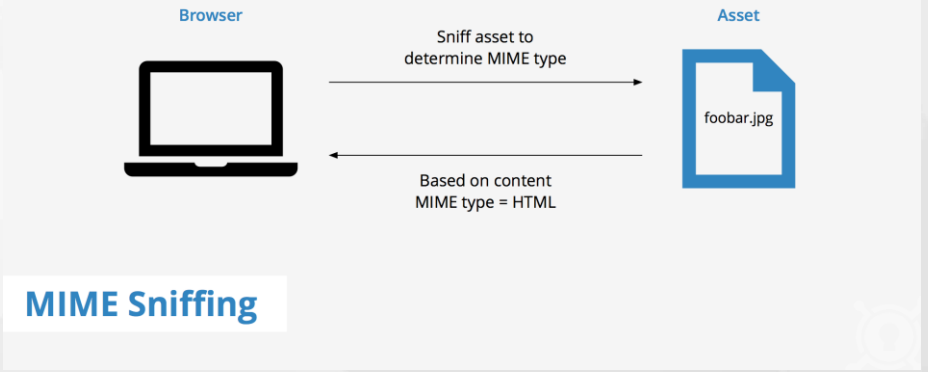
11 X-Content-Type-Options

Guideline	Security Benefit	Implementation Difficulty	Order†	Requirements	Notes
X-Content-Type-Options	LOW	LOW	8	Recommended for all websites	Websites should verify that they are setting the proper MIME types for all resources

เป็น Header ที่จะป้องกันการโจมตีด้วยการส่งข้อมูลที่ไม่ถูกต้องตามประเภทของไฟล์ (MIME types) เช่น การส่งข้อมูลที่ไม่ใช่ไฟล์ที่ต้องการก็จะไม่ถูกนำมาประมวลผลตามประเภทของไฟล์

MIME Type นั้นเป็นการบ่งบอกให้ทราบถึงว่าไฟล์ต่าง ๆ เหล่านั้นเป็น Format อะไร ซึ่งโครงสร้างของ MIME Type จะเป็นลักษณะ type/subtype ยกตัวอย่าง เช่น ไฟล์ add9.jpg จะมี MIME Type เป็น image/jpeg

เป็นการกำหนดไปเลยว่า content ที่จะให้แสดงผลเป็นข้อมูลประเภทไหน เพื่อไม่ให้ปัญหาของการคิดไปเองของ Browser เช่นใน HTTP Response มีการกำหนด Content-Type: text/html แม้จะเป็นไฟล์แบบ gif , Browser ก็จะแสดงผลเป็น gif โดยไม่สนใจ Content-Type ที่มีการกำหนดไว้เลย โดยปัญหานี้เรียกว่า “MIME Sniffing”



Web Security Cheat Sheet

12 X-Frame-Options

Guideline	Security Benefit	Implementation Difficulty	Order†	Requirements	Notes
X-Frame-Options	HIGH	LOW	5	Mandatory for all websites	Websites that don't use DENY or SAMEORIGIN must employ clickjacking defenses

** X-Frame-Options สามารถกำหนดได้เป็น

deny คือไม่ยินยอมให้เว็บไซต์ใด ๆนำเพจไปใส่ใน iframe

SAMEORIGIN กำหนดให้สามารถนำเพจไป render ใน iframe ได้ หากเป็น origin เดียวกัน

ALLOW-FROM กำหนดไปเป็น URL ใด ๆที่อนุญาตให้เราอนุญาตให้นำเพจไปเป็น iframe ได้

Clickjacking Attack คือ?

การโจมตีเว็บไซต์รูปแบบหนึ่งที่ผู้โจมตี (Attacker) นำเอา code เข้าไปฝังซ่อนไว้ส่วนใดส่วนหนึ่งที่หน้าเว็บไซต์อื่น อย่างเช่น ปุ่ม Submit, หน้าถัดไป (Next page), Picture Link เป็นต้น เพื่อรับ Response ทั้งหมดให้ Redirect ไปยังเว็บไซต์ปลอมที่ถูกสร้างขึ้น

เอาไว้กันการนำ page ไป render อยู่ภายใต้ iframe ซึ่งทำให้การโจมตีอย่าง Clickjacking ไม่เวิร์คด้วย ยกตัวอย่างการ Clickjacking คือการนำ page ไปใส่ไว้ใน iframe แล้วทำเป็นโปร่งใสแปะไว้ในเพจของการใช้งานปกติ เมื่อมีการ click หรือใช้งานใด ๆในเพจที่ถูกซ้อน ก็จะกลายเป็นการกระทำของการเพจที่ซ้อนนั้นแทน

Web Security Cheat Sheet

13 X-XSS-Protection Cross-Site Scripting

ใช้ในการ filter XSS ต่างๆที่ถูกใส่ไว้ในเว็บเพจที่ได้รับ โดย X-XSS-Protection
มี 2 mode คือ

- 0 คือ disable
- 1 คือ enable โดย Browser จะเป็นตัวกรองการแสดงผลของเพจที่ได้รับ
- 1 คือการ filter XSS ออก แต่ยัง render page ปกติ
- 1; mode=block คือการ filter XSS เมื่อพบแล้วจะทำการหยุดการ render page
- 1; report=http://target.com/url คือการ filter XSS แล้วจะทำการแจ้งเตือนไปยัง page ที่เรากำหนด

เป็นคุณลักษณะของ Internet Explorer, Chrome และ Safari ที่หยุดการโหลดหน้าเว็บเมื่อตรวจพบการโจมตีแบบ cross-site scripting (XSS) การป้องกันเหล่านี้ไม่จำเป็นในเบราว์เซอร์สมัยใหม่

```
# Block pages from loading when they detect reflected XSS attacks
X-XSS-Protection: 1; mode=block
```

Guideline	Security Benefit	Implementation Difficulty	Order†	Requirements	Notes
Cross-site Request Forgery Tokenization	HIGH	UNKNOWN	6	Varies	Mandatory for websites that allow destructive changes Unnecessary for all other websites Most application frameworks have built-in CSRF tokenization to ease implementation

ข้อสงสัย/เสนอแนะ

หากหน่วยงานพบปัญหา ข้อขัดข้อง หรือมีความประสงค์จะเสนอแนะเพิ่มเติม เกี่ยวกับการดำเนินการตามคำแนะนำฉบับนี้ สามารถติดต่อโดยตรงได้ที่ศูนย์ประสานการรักษาความมั่นคงปลอดภัยระบบคอมพิวเตอร์แห่งชาติ (ศปช.) สำนักงานคณะกรรมการการรักษาความมั่นคงปลอดภัยไซเบอร์แห่งชาติ (สกมช.)

เลขที่ ๑๒๐ หมู่ ๓ อาคารรัฐประศาสนภักดี (อาคารบี) ชั้น ๗ ศูนย์ราชการเฉลิมพระเกียรติ ถนนแจ้งวัฒนะ แขวงทุ่งสองห้อง เขตหลักสี่ กรุงเทพฯ ๑๐๒๑๐ โทรศัพท์ ๐ ๒๑๔๒ ๖๘๘๘ โทรสาร ๐ ๒๑๔๓ ๗๕๙๓
อีเมล ncert@ncsa.or.th

หน่วยงานสามารถติดตามข่าวสารล่าสุดจากสำนักงานผ่าน Line Openchat ตาม QR Code นี้

Contact us

National Cyber Security Agency – NCSA

NCSA Thailand

Line ID : NCSA Thailand

E-Mail: saraban@ncsa.or.th

02-142-6885

shorturl.at/hkAC2