EGA

e-Government Agency

Electronic Government Agency (Public Organization)

สำนักงานรัฐบาลอิเล็กทรอนิกส์ (องค์การมหาชน)

Secure Code

OWASP

Open Web Application Security Project

OWASP
Open Web Application Security Project

EGA
e-Government Agency
THAILAND

Secure Coding Practices Checklist

- Input Validation
- Output Encoding
- Authentication and Password Management
- Session Management
- Access Control
- Cryptographic Practices
- Error Handing and Logging
- Data Protection
- Communication Security
- System Configuration
- Database Security
- File Management
- Memory Management
- General Coding Practices

https://www.owasp.org/index.php/OWASP_Secure_Coding_Practices_-_Quick_Reference_Guide

Input Validation

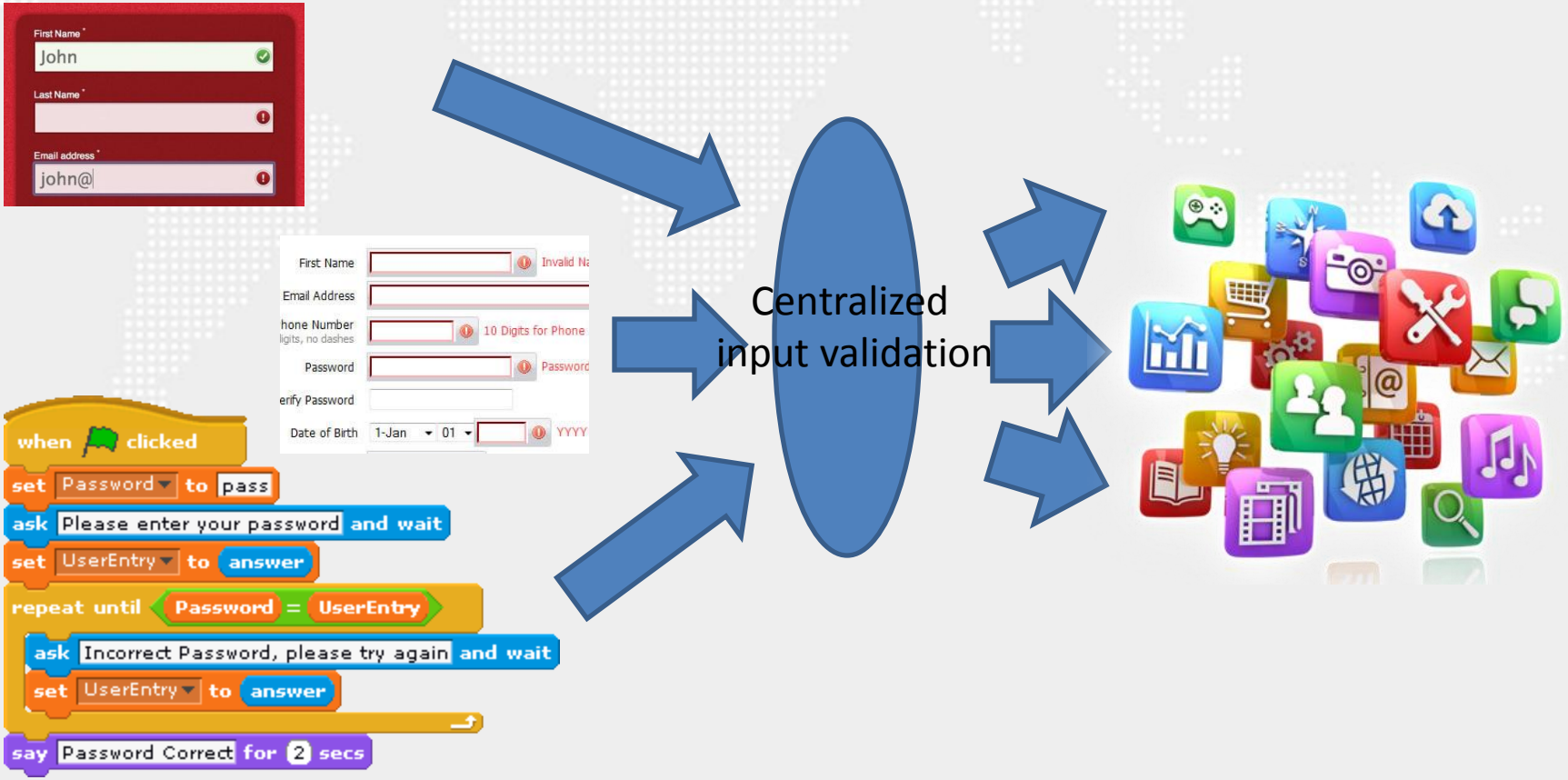
Conduct all data validation on a trusted system

Input Validation

Identify all data sources and classify them into trusted and untrusted. Validate all data from untrusted sources

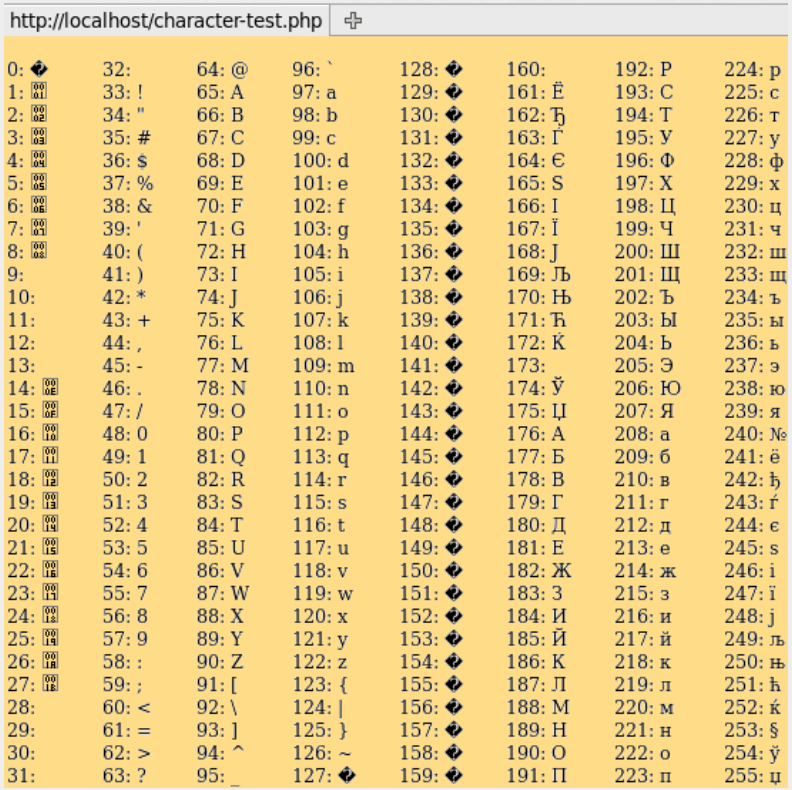
Input Validation

There should be a centralized input validation routine for the application



Input Validation

Specify proper character sets, such as UTF-8,for all sources of input

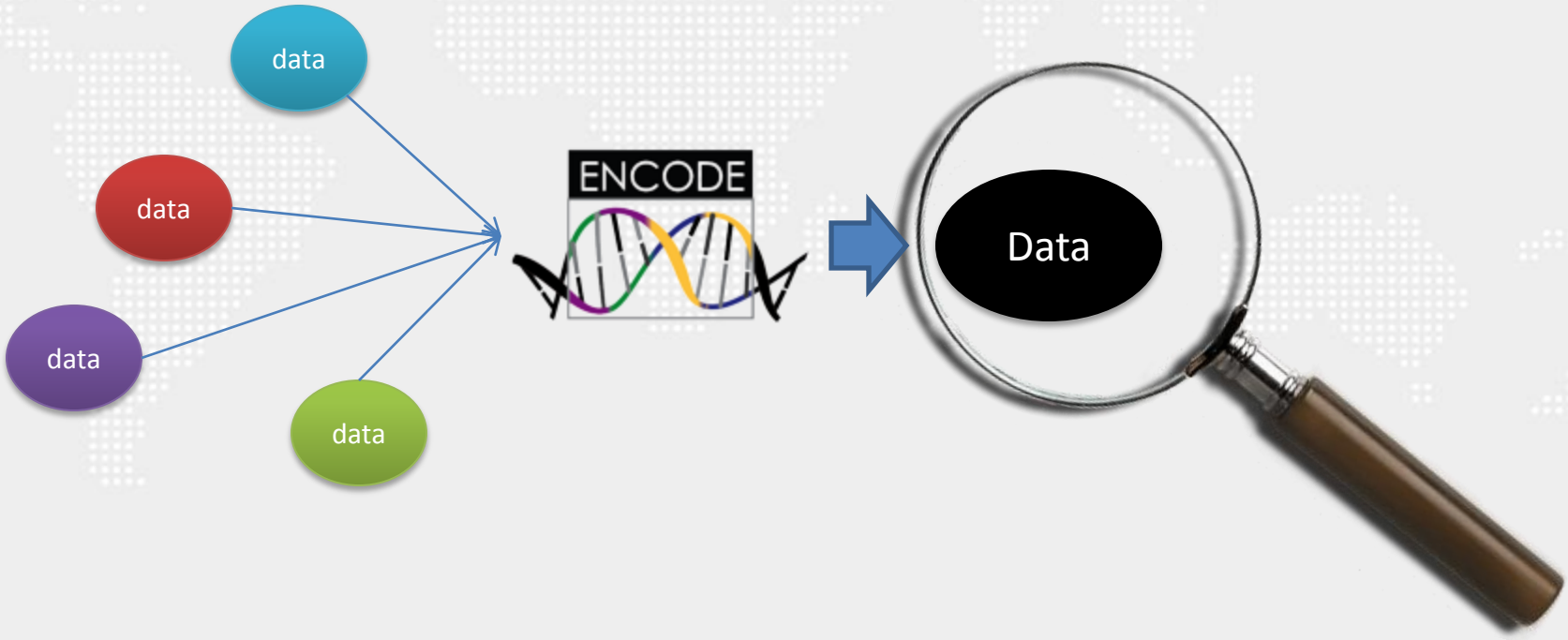


http://localhost/character-test.php

0:	32:	64: @	96: `	128:	160:	192: Р	224: р
1:	33: !	65: A	97: a	129:	161: Ё	193: С	225: с
2:	34: "	66: B	98: b	130:	162: Ђ	194: Т	226: т
3:	35: #	67: C	99: c	131:	163: Ѓ	195: У	227: у
4:	36: $	68: D	100: d	132:	164: Є	196: Ф	228: ф
5:	37: %	69: E	101: e	133:	165: S	197: Х	229: х
6:	38: &	70: F	102: f	134:	166: I	198: Ц	230: ц
7:	39: '	71: G	103: g	135:	167: Ї	199: Ч	231: ч
8:	40: (	72: H	104: h	136:	168: J	200: Ш	232: ш
9:	41:)	73: I	105: i	137:	169: Љ	201: Щ	233: щ
10:	42: *	74: J	106: j	138:	170: Њ	202: Ъ	234: ъ
11:	43: +	75: K	107: k	139:	171: Ћ	203: Ы	235: ы
12:	44: ,	76: L	108: l	140:	172: Ќ	204: Ь	236: ь
13:	45: -	77: M	109: m	141:	173:	205: Э	237: э
14:	46: .	78: N	110: n	142:	174: Ў	206: Ю	238: ю
15:	47: /	79: O	111: o	143:	175: Џ	207: Я	239: я
16:	48: 0	80: P	112: p	144:	176: А	208: а	240: №
17:	49: 1	81: Q	113: q	145:	177: Б	209: б	241: ё
18:	50: 2	82: R	114: r	146:	178: В	210: в	242: ђ
19:	51: 3	83: S	115: s	147:	179: Г	211: г	243: ѓ
20:	52: 4	84: T	116: t	148:	180: Д	212: д	244: є
21:	53: 5	85: U	117: u	149:	181: Е	213: е	245: s
22:	54: 6	86: V	118: v	150:	182: Ж	214: ж	246: і
23:	55: 7	87: W	119: w	151:	183: З	215: з	247: ї
24:	56: 8	88: X	120: x	152:	184: И	216: и	248: j
25:	57: 9	89: Y	121: y	153:	185: Й	217: й	249: љ
26:	58: :	90: Z	122: z	154:	186: К	218: к	250: њ
27:	59: ;	91: [	123: {	155:	187: Л	219: л	251: ћ
28:	60: <	92: \	124: |	156:	188: М	220: м	252: ќ
29:	61: =	93:]	125: }	157:	189: Н	221: н	253: §
30:	62: >	94: ^	126: ~	158:	190: О	222: о	254: ў
31:	63: ?	95: _	127:	159:	191: П	223: п	255: џ

Input Validation

Encode data to a common character set before validating

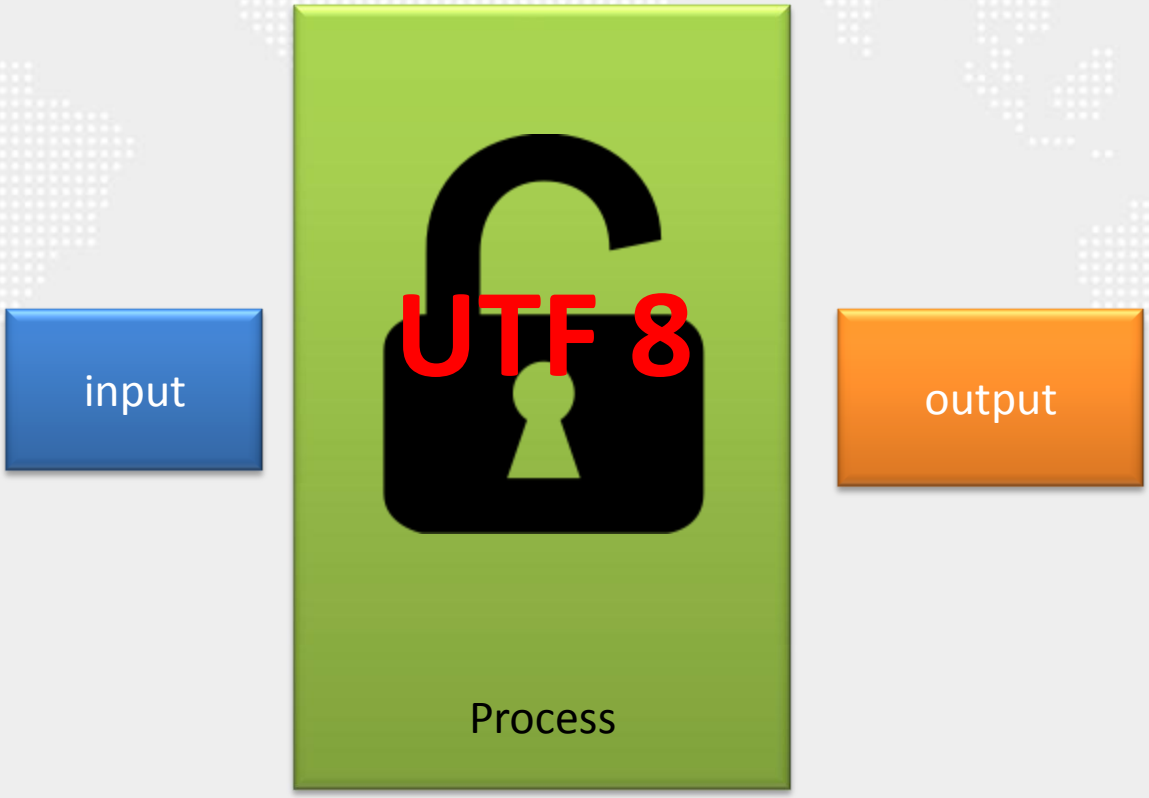


Input Validation

All validation failures should result in input rejection

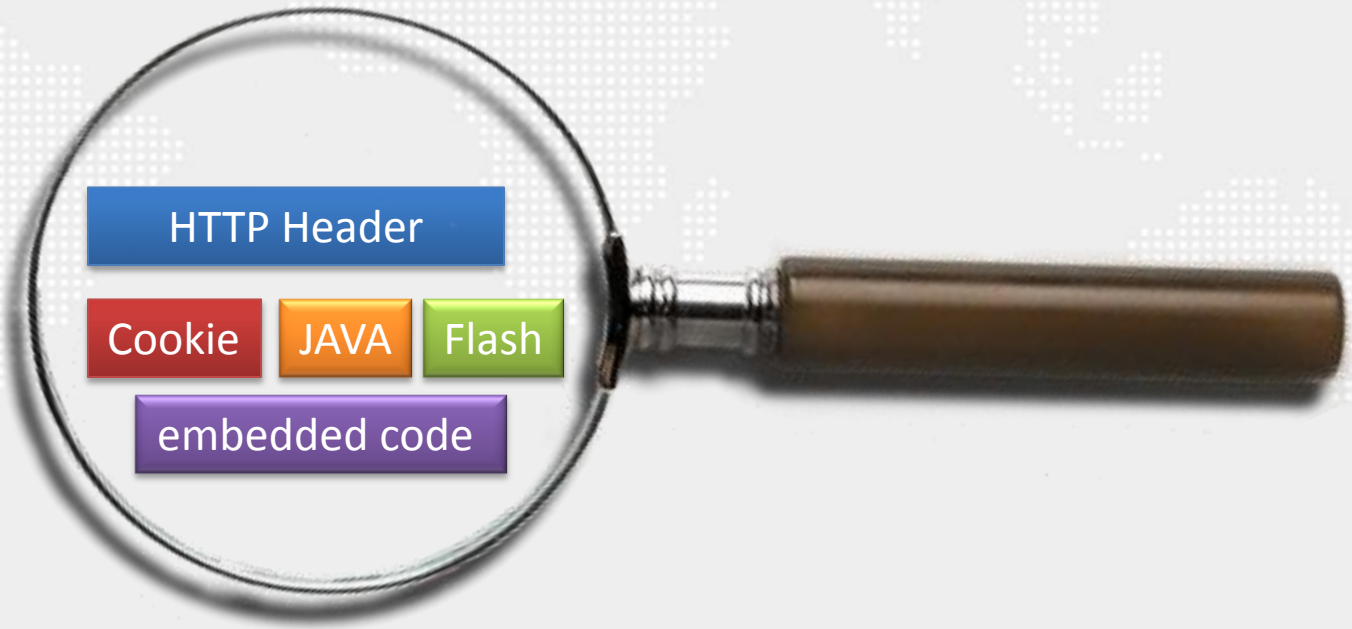
Input Validation

Determine if the system supports UTF-8 extended character sets and if so, validate after UTF-8 decoding is completed



Input Validation

Validate all client provided data before processing, including all parameters, URLs and HTTP header content



Input Validation

Verify that header values in both requests and responses contain only ASCII characters

ASCII TABLE

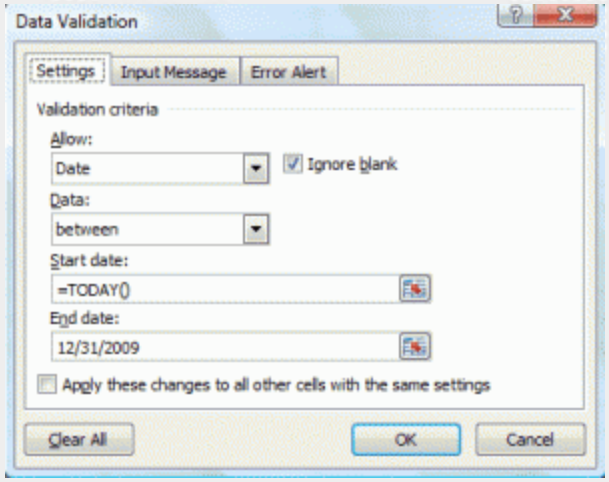
Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]	48	30	110000	60	0	96	60	1100000	140	`
1	1	1	1	[START OF HEADING]	49	31	110001	61	1	97	61	1100001	141	a
2	2	10	2	[START OF TEXT]	50	32	110010	62	2	98	62	1100010	142	b
3	3	11	3	[END OF TEXT]	51	33	110011	63	3	99	63	1100011	143	c
4	4	100	4	[END OF TRANSMISSION]	52	34	110100	64	4	100	64	1100100	144	d
5	5	101	5	[ENQUIRY]	53	35	110101	65	5	101	65	1100101	145	e
6	6	110	6	[ACKNOWLEDGE]	54	36	110110	66	6	102	66	1100110	146	f
7	7	111	7	[BELL]	55	37	110111	67	7	103	67	1100111	147	g
8	8	1000	10	[BACKSPACE]	56	38	111000	70	8	104	68	1101000	150	h
9	9	1001	11	[HORIZONTAL TAB]	57	39	111001	71	9	105	69	1101001	151	i
10	A	1010	12	[LINE FEED]	58	3A	111010	72	:	106	6A	1101010	152	j
11	B	1011	13	[VERTICAL TAB]	59	3B	111011	73	;	107	6B	1101011	153	k
12	C	1100	14	[FORM FEED]	60	3C	111100	74	<	108	6C	1101100	154	l
13	D	1101	15	[CARRIAGE RETURN]	61	3D	111101	75	=	109	6D	1101101	155	m
14	E	1110	16	[SHIFT OUT]	62	3E	111110	76	>	110	6E	1101110	156	n
15	F	1111	17	[SHIFT IN]	63	3F	111111	77	?	111	6F	1101111	157	o
16	10	10000	20	[DATA LINK ESCAPE]	64	40	1000000	100	@	112	70	1110000	160	p
17	11	10001	21	[DEVICE CONTROL 1]	65	41	1000001	101	A	113	71	1110001	161	q
18	12	10010	22	[DEVICE CONTROL 2]	66	42	1000010	102	B	114	72	1110010	162	r
19	13	10011	23	[DEVICE CONTROL 3]	67	43	1000011	103	C	115	73	1110011	163	s
20	14	10100	24	[DEVICE CONTROL 4]	68	44	1000100	104	D	116	74	1110100	164	t
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]	69	45	1000101	105	E	117	75	1110101	165	u
22	16	10110	26	[SYNCHRONOUS IDLE]	70	46	1000110	106	F	118	76	1110110	166	v
23	17	10111	27	[ENG OF TRANS. BLOCK]	71	47	1000111	107	G	119	77	1110111	167	w
24	18	11000	30	[CANCEL]	72	48	1001000	110	H	120	78	1111000	170	x
25	19	11001	31	[END OF MEDIUM]	73	49	1001001	111	I	121	79	1111001	171	y
26	1A	11010	32	[SUBSTITUTE]	74	4A	1001010	112	J	122	7A	1111010	172	z
27	1B	11011	33	[ESCAPE]	75	4B	1001011	113	K	123	7B	1111011	173	{
28	1C	11100	34	[FILE SEPARATOR]	76	4C	1001100	114	L	124	7C	1111100	174	|
29	1D	11101	35	[GROUP SEPARATOR]	77	4D	1001101	115	M	125	7D	1111101	175	}
30	1E	11110	36	[RECORD SEPARATOR]	78	4E	1001110	116	N	126	7E	1111110	176	~
31	1F	11111	37	[UNIT SEPARATOR]	79	4F	1001111	117	O	127	7F	1111111	177	[DEL]
32	20	100000	40	[SPACE]	80	50	1010000	120	P					
33	21	100001	41	!	81	51	1010001	121	Q					
34	22	100010	42	"	82	52	1010010	122	R					
35	23	100011	43	#	83	53	1010011	123	S					
36	24	100100	44	$	84	54	1010100	124	T					
37	25	100101	45	%	85	55	1010101	125	U					
38	26	100110	46	&	86	56	1010110	126	V					
39	27	100111	47	'	87	57	1010111	127	W					
40	28	101000	50	(	88	58	1011000	130	X					
41	29	101001	51	)	89	59	1011001	131	Y					
42	2A	101010	52	*	90	5A	1011010	132	Z					
43	2B	101011	53	+	91	5B	1011011	133	[					
44	2C	101100	54	,	92	5C	1011100	134	\					
45	2D	101101	55	-	93	5D	1011101	135	]					
46	2E	101110	56	.	94	5E	1011110	136	^					
47	2F	101111	57	/	95	5F	1011111	137	_					

Input Validation

Validate data from redirects

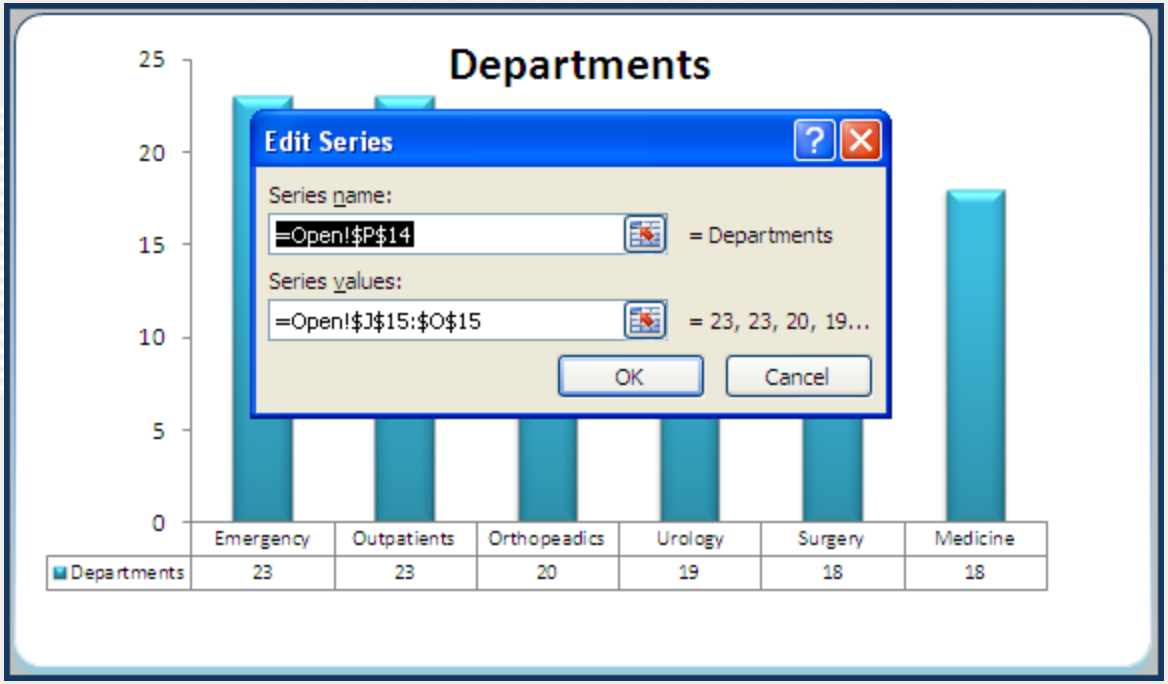
Input Validation

Validate for expected data types



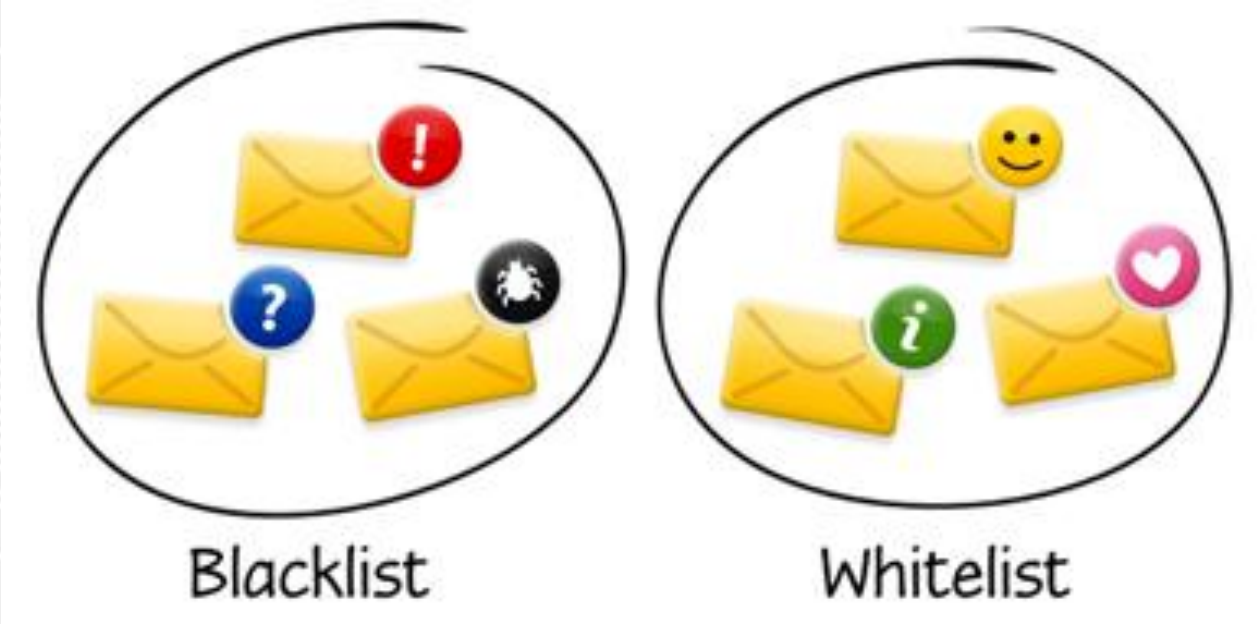
Input Validation

Validate data range ,lenght



Input Validation

Validate all input against a "white" list of allowed characters



Input Validation

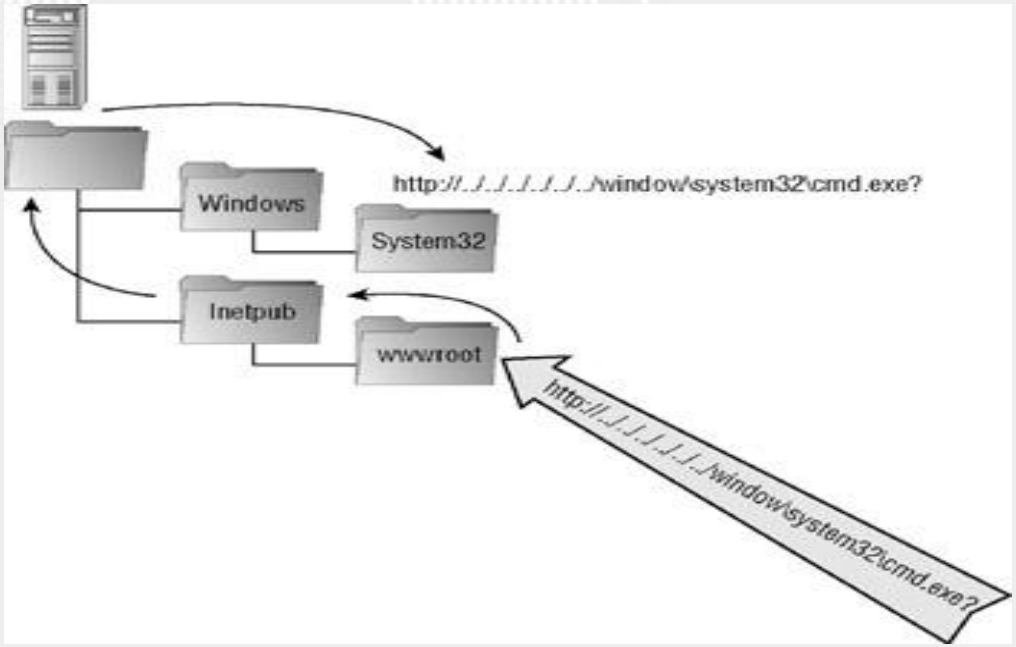
If any potentially *hazardous characters must be allowed as input, be sure that you implement additional controls like output encoding, secure task specific APIs and accounting for the utilization of that data throughout the application .*
Examples of common hazardous characters include:
< > " ' % ( ) & + \ \' \"

' &

Input Validation

If your standard validation routine cannot address the following inputs, then they should be checked discretely

- o Check for null bytes (%00)
- o Check for new line characters (%0d, %0a, \r, \n)
- o Check for “dot-dot-slash" (../ or ..\) path alterations characters.



Secure Coding Practices Checklist

- Input Validation
- Output Encoding
- Authentication and Password Management
- Session Management
- Access Control
- Cryptographic Practices
- Error Handing and Logging
- Data Protection
- Communication Security
- System Configuration
- Database Security
- File Management
- Memory Management
- General Coding Practices

Output Encoding

- Conduct all encoding on a trusted system
- Utilize a standard
- *Contextually output encode all data returned to the client that originated outside the application's trust boundary.*
- Encode all characters unless they are known to be safe for the intended interpreter
- Contextually *sanitize all output of un-trusted data to queries for SQL, XML, and LDAP*
- *Sanitize all output of un-trusted data to operating system commands*

Secure Coding Practices Checklist

- Input Validation
- Output Encoding
- Authentication and Password Management
- Session Management
- Access Control
- Cryptographic Practices
- Error Handing and Logging
- Data Protection
- Communication Security
- System Configuration
- Database Security
- File Management
- Memory Management
- General Coding Practices

Authentication and Password Management

- Require authentication for all pages and resources, except those specifically intended to be public
- All authentication controls must be enforced on a trusted system
- Establish and utilize standard, tested, authentication services whenever possible
- Use a centralized implementation for all authentication controls, including libraries that call external authentication services
- Segregate authentication logic from the resource being requested and use redirection to and from the centralized authentication control
- All authentication controls should fail securely
- All administrative and account management functions must be at least as secure as the primary authentication mechanism
- If your application manages a credential store, it should ensure that only cryptographically strong one-way salted hashes of passwords are stored and that the table/file that stores the passwords and keys is write-able only by the application. (Do not use the MD5 algorithm if it can be avoided)
- Password hashing must be implemented on a trusted system (e.g., The server).

Authentication and Password Management

- Validate the authentication data only on completion of all data input, especially for *sequential authentication implementations*
- Authentication failure responses should not indicate which part of the authentication data was incorrect. For example, instead of "Invalid username" or "Invalid password", just use "Invalid username and/or password" for both.
- Utilize authentication for connections to external systems that involve sensitive information or functions
- Authentication credentials for accessing services external to the application should be encrypted and stored in a protected location on a trusted system

Authentication and Password Management

- Use only HTTP POST requests to transmit authentication credentials
- Only send non-temporary passwords over an encrypted connection or as encrypted data, such as in an encrypted email. Temporary passwords associated with email resets may be an exception
- Enforce password complexity requirements established by policy or regulation.
- Enforce password length requirements established by policy or regulation.
- Password entry should be obscured on the user's screen.
- Enforce account disabling after an established number of invalid login attempts
- Password reset and changing operations require the same level of controls as account creation and authentication.
- Password reset questions should support sufficiently random answers.
- If using email based resets, only send email to a pre-registered address with a temporary link/password
- Temporary passwords and links should have a short expiration time
- Enforce the changing of temporary passwords on the next use

Secure Coding Practices Checklist

- Input Validation
- Output Encoding
- Authentication and Password Management
- Session Management
- Access Control
- Cryptographic Practices
- Error Handing and Logging
- Data Protection
- Communication Security
- System Configuration
- Database Security
- File Management
- Memory Management
- General Coding Practices

Error Handing and Logging

- Do not disclose sensitive information in error responses, including system details, session identifiers or account information
- Use error handlers that do not display debugging or stack trace information
- Implement generic error messages and use custom error pages
- The application should handle application errors and not rely on the server configuration
- Properly free allocated memory when error conditions occur
- Error handling logic associated with security controls should deny access by default
- All logging controls should be implemented on a trusted system
- Logging controls should support both **success and failure** of specified security events
- Ensure logs contain important *log event data*
- Ensure log entries that include un-trusted data will not execute as code in the intended log viewing interface or software
- Restrict access to logs to only authorized individuals

Error Handing and Logging

- Utilize a master routine for all logging operations
- Do not store sensitive information in logs, including unnecessary system details, session identifiers or passwords
- Ensure that a mechanism exists to conduct log analysis
- Log all input validation failures
- Log all authentication attempts, especially failures
- Log all access control failures
- Log all apparent tampering events, including unexpected changes to state data
- Log attempts to connect with invalid or expired session tokens
- Log all system exceptions
- Log all administrative functions, including changes to the security configuration settings
- Log all backend TLS connection failures
- Log cryptographic module failures
- Use a cryptographic hash function to validate log entry integrity

Secure Coding Practices Checklist

- Input Validation
- Output Encoding
- Authentication and Password Management
- Session Management
- Access Control
- Cryptographic Practices
- Error Handing and Logging
- Data Protection
- Communication Security
- System Configuration
- Database Security
- File Management
- Memory Management
- General Coding Practices

General Coding Practices

- Use tested and approved managed code rather than creating new unmanaged code for common tasks
- Utilize task specific built-in APIs to conduct operating system tasks. Do not allow the application to issue commands directly to the Operating System, especially through the use of application initiated command shells
- Use checksums or hashes to verify the integrity of interpreted code, libraries, executables, and configuration files
- Utilize locking to prevent multiple simultaneous requests or use a synchronization mechanism to prevent race conditions
- Protect shared variables and resources from inappropriate concurrent access
- Explicitly initialize all your variables and other data stores, either during declaration or just before the first usage
- In cases where the application must run with elevated privileges, raise privileges as late as possible, and drop them as soon as possible
- Avoid calculation errors by understanding your programming language's underlying representation and how it interacts with numeric calculation.

General Coding Practices

- Do not pass user supplied data to any dynamic execution function
- Restrict users from generating new code or altering existing code
- Review all secondary applications, third party code and libraries to determine business necessity and validate safe functionality, as these can introduce new vulnerabilities
- Implement safe updating. If the application will utilize automatic updates, then use cryptographic signatures for your code and ensure your download clients verify those signatures. Use encrypted channels to transfer the code from the host server

Secure Coding Practices Checklist

- Input Validation
- Output Encoding
- Authentication and Password Management
- Session Management
- Access Control
- Cryptographic Practices
- Error Handing and Logging
- Data Protection
- Communication Security
- System Configuration
- Database Security
- File Management
- Memory Management
- General Coding Practices

QUESTION & ANSWER SESSION

Name พงศ์ระพี นาคมณี [Information Security Engineer]
e-mail : pongrapee@ega.or.th tel. : 02-612-6000(4303)

Thank You

Electronic Government Agency (Public Organization)

website : www.ega.or.th
e-mail : helpdesk@ega.or.th
Tel. : (+66) 0 2612 6000
Hotline : (+66) 0 2612 6060