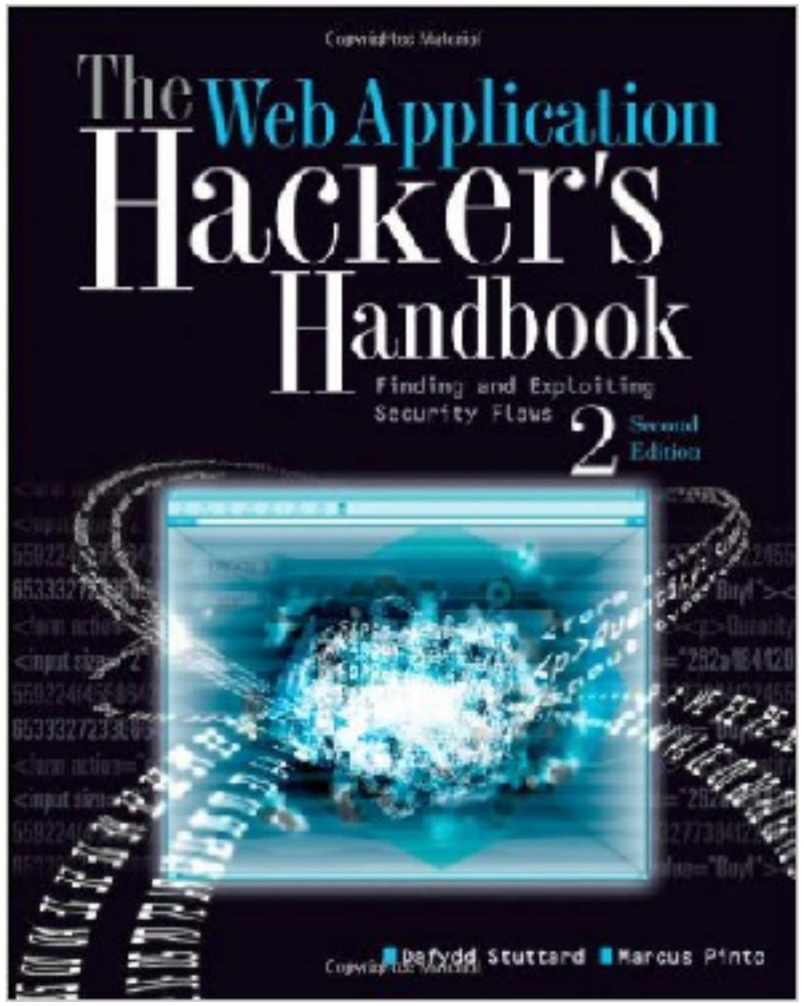
CNIT 129S: Securing Web Applications


Copyrighted Material
The Web Application Hacker's Handbook
Finding and Exploiting Security Flaws
2 Second Edition
Dafydd Stuttard
Marcus Pinto

Ch 7: Attacking Session Management

Updated 3-2-22

Session Management

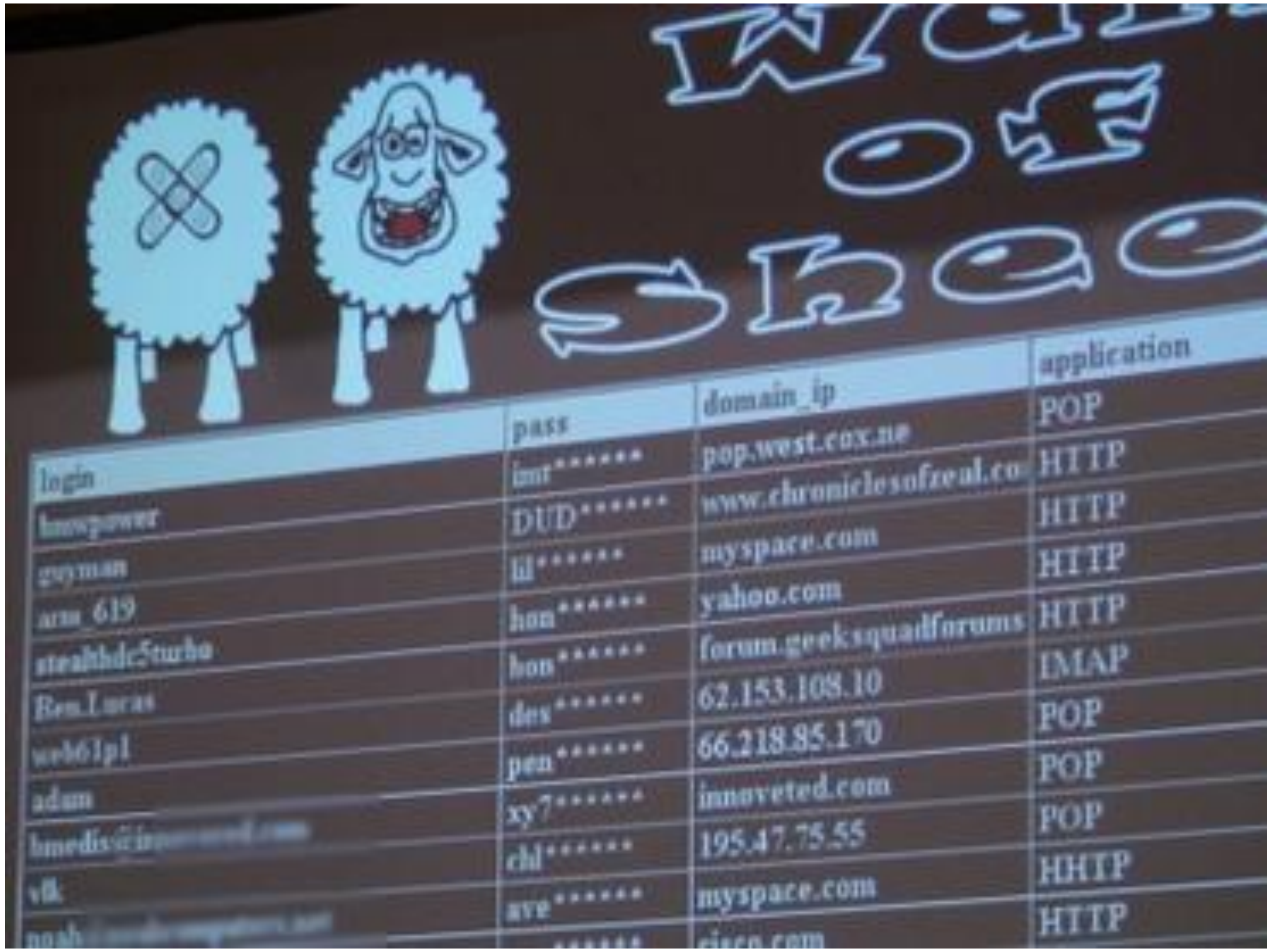
- Enables application to identify a given user over a number of different requests
    - Ex: login, then later requests
- Fundamental security component
- A prime target for attackers

Session Management

- Potential consequences of session management attacks
  - A user can masquerade as another user
  - Privilege escalation to administrator, owning the entire application
- Can be as simple as incrementing the value of a token

Wall of Sheep

- **My Gmail was hacked at Defcon**
- **By stealing and replaying my session cookie**
- **Using Hamster and Ferret**



login	pass	domain_ip	application
hnwpower	imr******	pop.west.cox.ne	POP
guyman	DUD******	www.chroniclesofzeal.co	HTTP
arm_619	lil******	myspace.com	HTTP
stealthdc5turbo	hon******	yahoo.com	HTTP
Ben.Lucas	hon******	forum.geeksquadforums	HTTP
web61p1	des******	62.153.108.10	IMAP
adam	pen******	66.218.85.170	POP
bmedis@[redacted]	xy7******	innoveted.com	POP
vlk	chl******	195.47.75.55	POP
noah[redacted]	ave******	myspace.com	HHTP
	******	cisco.com	HTTP

The Need for State

Web 1.0

- **Stateless**
- **Static pages**
- **No custom content**
- **No logging in**

Logging In

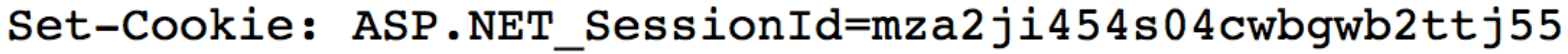
- **Allow user to register and log in**
- **Require a session to maintain the "state" of being authenticated**
- **Otherwise user would have to log in to each page**

No Login

- **Application with no login function still use sessions**
- **Shopping basket**

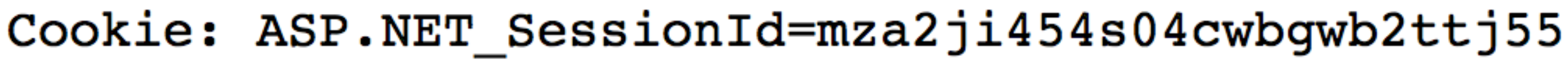
Session Token

- **Server's first response contains a Set-Cookie: header**



```
Set-Cookie: ASP.NET_SessionId=mza2ji454s04cwbgwb2ttj55
```

- **Each subsequent request from the client contains the Cookie: header**



```
Cookie: ASP.NET_SessionId=mza2ji454s04cwbgwb2ttj55
```

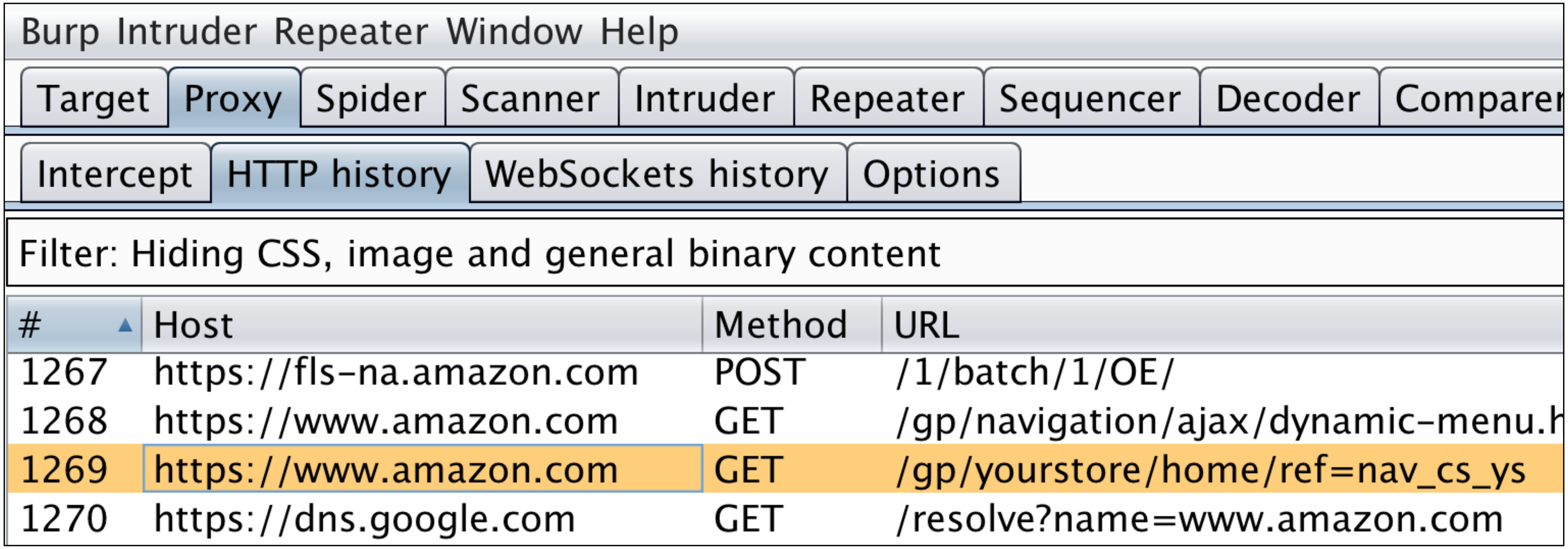
Vulnerabilities

- **Two categories**
  - **Weakness in generation of session tokens**
  - **Weakness in handling session tokens throughout their life cycle**

Finding the Real Session Token

- Sometimes a platform like ASP.NET generates a token but the app doesn't really use it
- To find a token, establish a session and then replay a request
  - Systematically remove each item you suspect of being the real token
  - Wait till response is no longer customized for your session
  - Burp Repeater is good for this

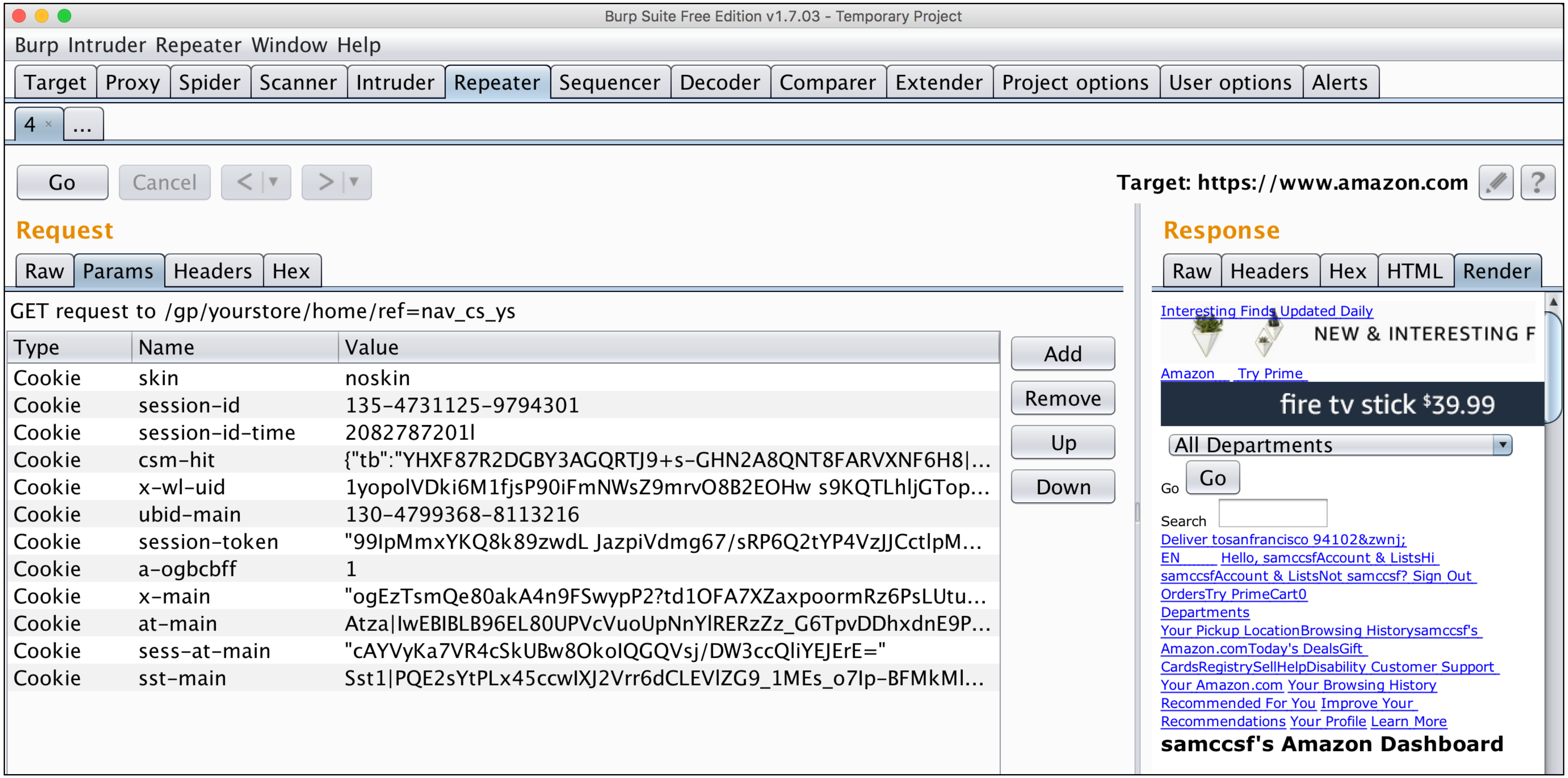
Amazon.com



Filter: Hiding CSS, image and general binary content

# ▲	Host	Method	URL
1267	https://fls-na.amazon.com	POST	/1/batch/1/OE/
1268	https://www.amazon.com	GET	/gp/navigation/ajax/dynamic-menu.h
1269	https://www.amazon.com	GET	/gp/yourstore/home/ref=nav_cs_ys
1270	https://dns.google.com	GET	/resolve?name=www.amazon.com

Original Request


Burp Suite Free Edition v1.7.03 - Temporary Project
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
4 ×
...
Go
Cancel
<
>
Add
Remove
Up
Down
Raw Params Headers Hex
Raw Headers Hex HTML Render
All Departments
Go

Target: https://www.amazon.com

Request

GET request to /gp/yourstore/home/ref=nav_cs_ys

Type	Name	Value
Cookie	skin	noskin
Cookie	session-id	135-4731125-9794301
Cookie	session-id-time	2082787201l
Cookie	csm-hit	{"tb":"YHXF87R2DGBY3AGQRTJ9+s-GHN2A8QNT8FARVXNF6H8|...
Cookie	x-wl-uid	1yopolVDki6M1fjsP90iFmNWsZ9mrvO8B2EOHw s9KQTLhljGTop...
Cookie	ubid-main	130-4799368-8113216
Cookie	session-token	"99IpMmxYKQ8k89zwdL JazpiVdmg67/sRP6Q2tYP4VzJJCctlpM...
Cookie	a-ogbcbff	1
Cookie	x-main	"ogEzTsmQe80akA4n9FSwypP2?td1OFA7XZaxpoormRz6PsLUtu...
Cookie	at-main	Atza|IwEBIBLB96EL80UPVcVuoUpNnYlRERzZz_G6TpvDDhxdnE9P...
Cookie	sess-at-main	"cAYVyKa7VR4cSkUBw8OkoIQGQVsj/DW3ccQliYEJErE="
Cookie	sst-main	Sst1|PQE2sYtPLx45ccwIXJ2Vrr6dCLEVlZG9_1MEs_o7lp-BFMkMl...

Response

Interesting Finds Updated Daily
NEW & INTERESTING F
Amazon Try Prime
fire tv stick $39.99

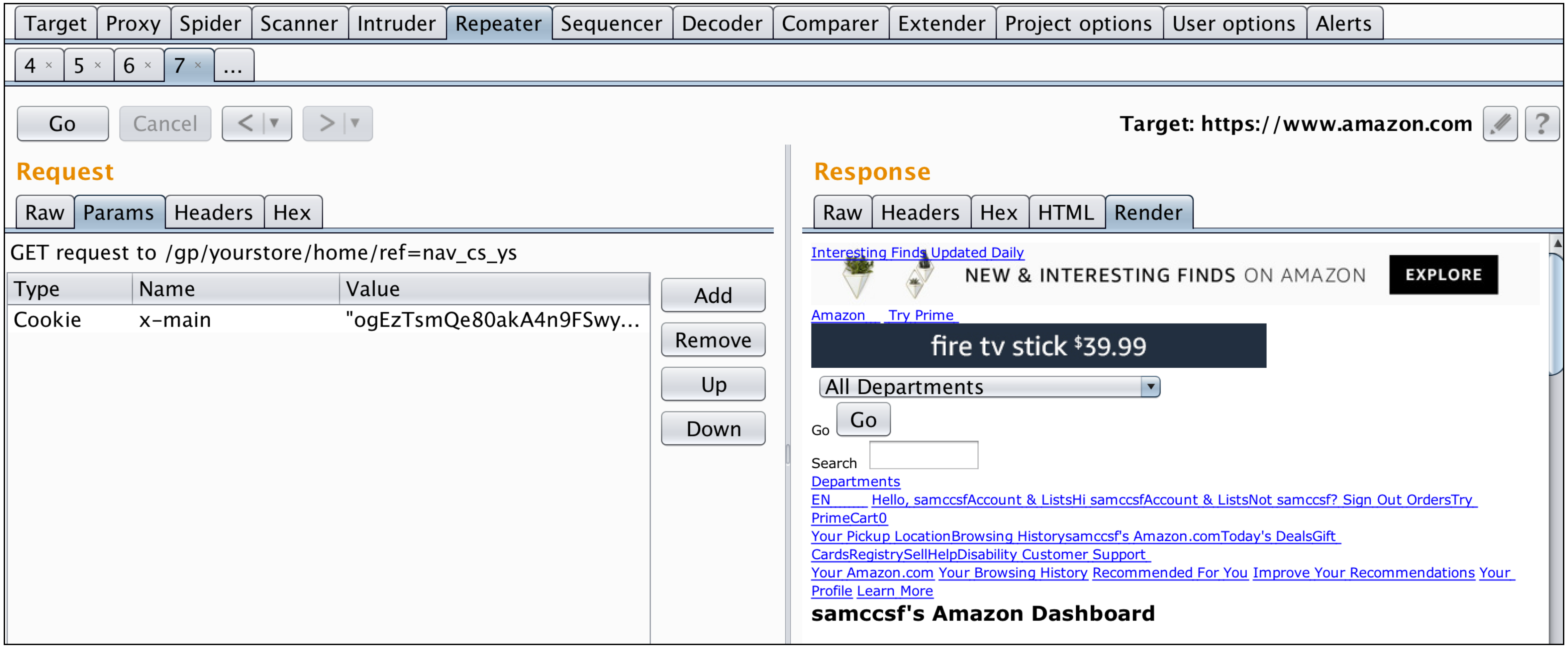
Go

Search

Deliver tosanfrancisco 94102&zwnj;
EN Hello, samccsfAccount & ListsHi
samccsfAccount & ListsNot samccsf? Sign Out
OrdersTry PrimeCart0
Departments
Your Pickup LocationBrowsing Historysamccsf's
Amazon.comToday's DealsGift
CardsRegistrySellHelpDisability Customer Support
Your Amazon.com Your Browsing History
Recommended For You Improve Your
Recommendations Your Profile Learn More

samccsf's Amazon Dashboard

Minimal Request



Target: https://www.amazon.com

Request

GET request to /gp/yourstore/home/ref=nav_cs_ys

Type	Name	Value
Cookie	x-main	"ogEzTsmQe80akA4n9FSwy...

Response

Interesting Finds Updated Daily

NEW & INTERESTING FINDS ON AMAZON EXPLORE

Amazon Try Prime

fire tv stick $39.99

All Departments

Go Go

Search

Departments

EN Hello, samccsfAccount & ListsHi samccsfAccount & ListsNot samccsf? Sign Out OrdersTry PrimeCart0

Your Pickup LocationBrowsing Historysamccsf's Amazon.comToday's DealsGift CardsRegistrySellHelpDisability Customer Support

Your Amazon.com Your Browsing History Recommended For You Improve Your Recommendations Your Profile Learn More

samccsf's Amazon Dashboard

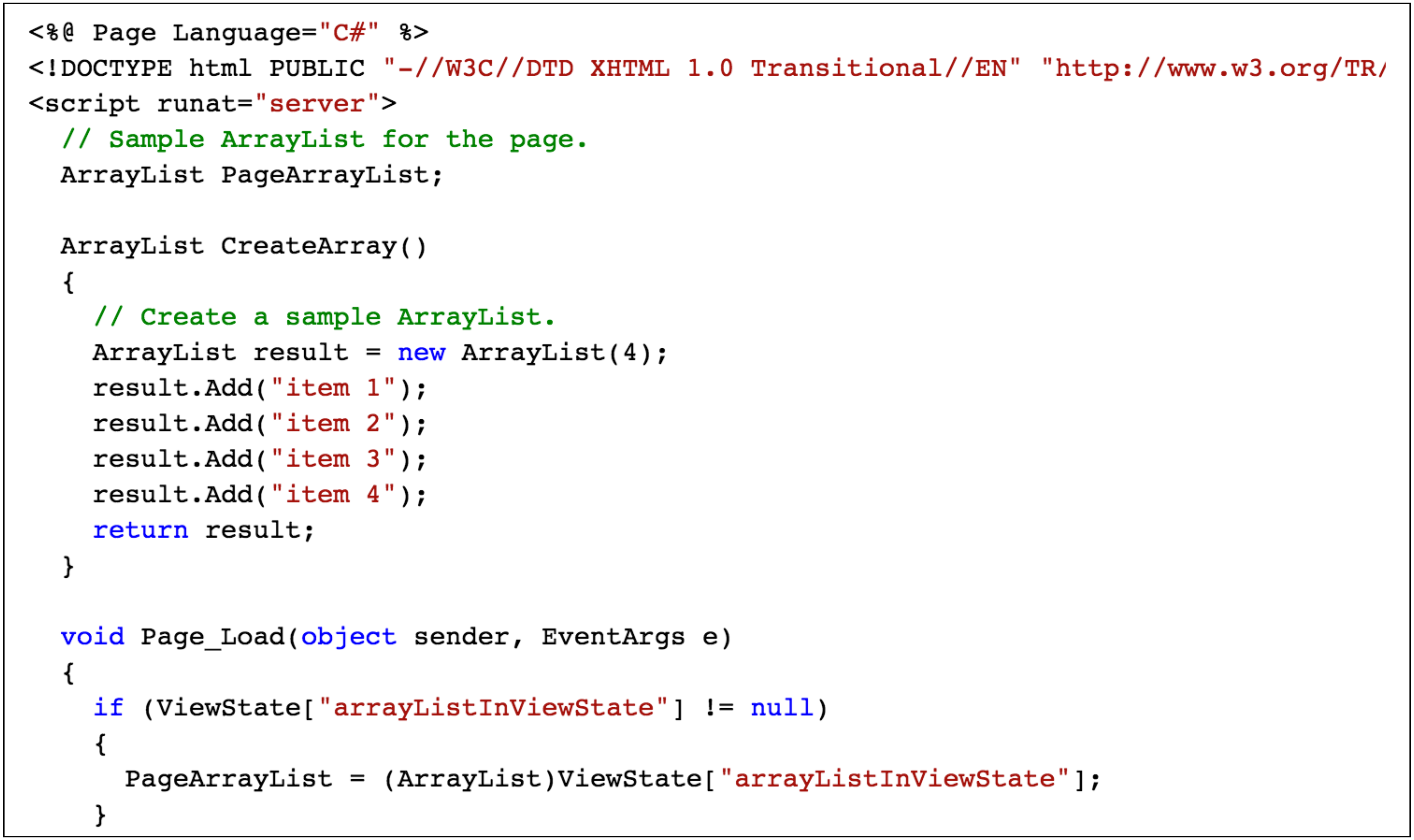
Alternatives to Sessions

HTTP Authentication

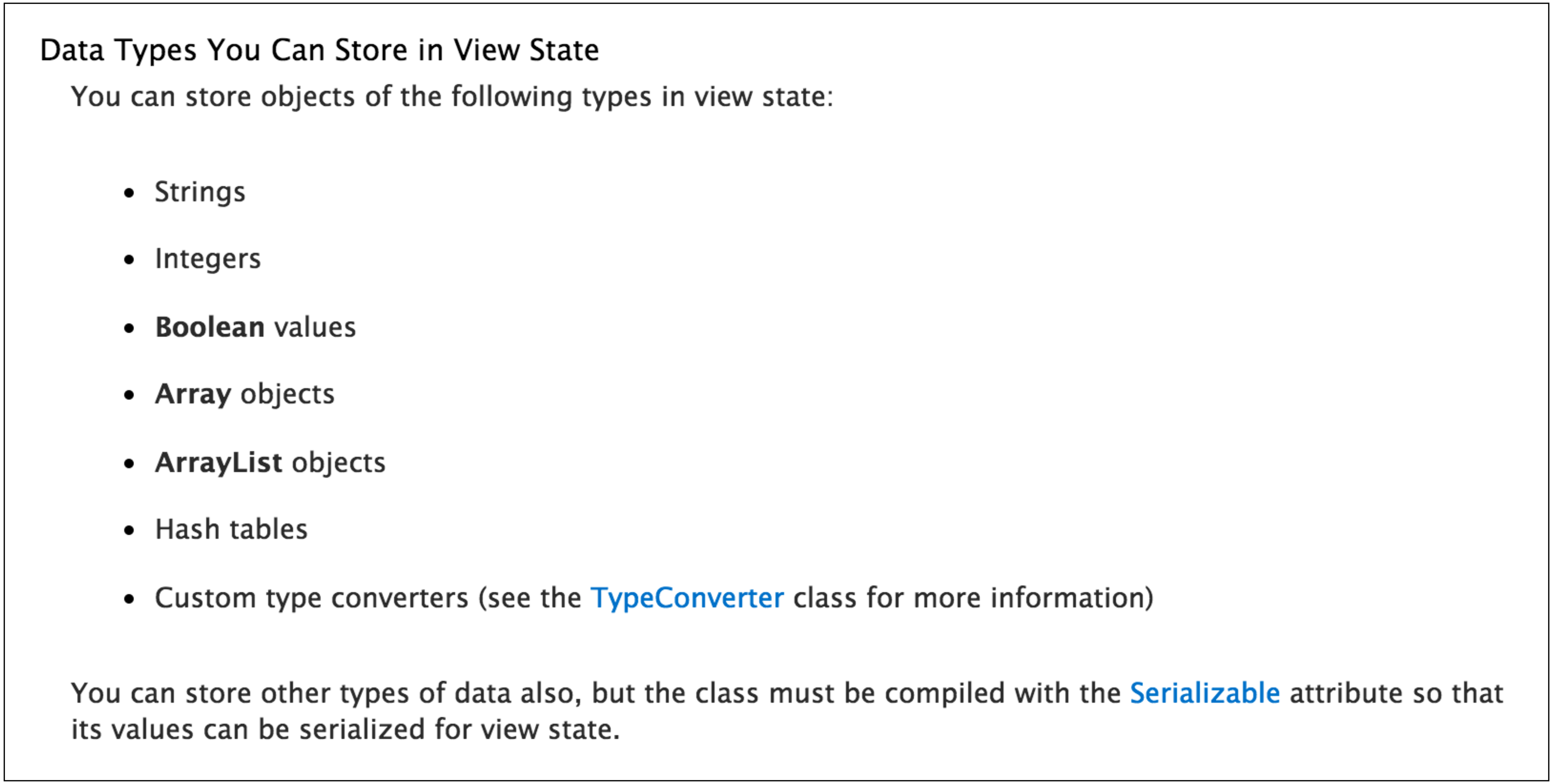
- **Basic, Digest, NTLM**
- **Pass credentials with every request in HTTP headers**
- **Not via application-specific code**
- **Rarely used on Internet-based applications**

Sessionless State Mechanisms

- **Application doesn't issue session tokens or manage the state**
- **Transmit all data required to manage the state via the client in a cookie or hidden form field**
- **Ex: ASP.NET ViewState**
  - **Link Ch 7a**



```
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
<script runat="server">
  // Sample ArrayList for the page.
  ArrayList PageArrayList;

  ArrayList CreateArray()
  {
    // Create a sample ArrayList.
    ArrayList result = new ArrayList(4);
    result.Add("item 1");
    result.Add("item 2");
    result.Add("item 3");
    result.Add("item 4");
    return result;
  }

  void Page_Load(object sender, EventArgs e)
  {
    if (ViewState["arrayListInViewState"] != null)
    {
      PageArrayList = (ArrayList)ViewState["arrayListInViewState"];
    }
```



Data Types You Can Store in View State

You can store objects of the following types in view state:

- Strings
- Integers
- **Boolean** values
- **Array** objects
- **ArrayList** objects
- Hash tables
- Custom type converters (see the TypeConverter class for more information)

You can store other types of data also, but the class must be compiled with the Serializable attribute so that its values can be serialized for view state.

Securing View State

- **Data must be protected, usually as a binary blob that is encrypted or signed**
  - **To prevent re-use on another machine**
- **ASP.NET View State uses Base64 and a hash made from a Machine Authentication Code**
- **Includes the machine's MAC address**
- **Expiration time enforces session timeouts**

Indicators of Sessionless State Mechanisms

- **Token-like data items >=100 bytes long**
- **New token-like item in response to every request**
- **Data is encrypted (structureless) or signed (structured accompanied by a few random bytes)**
- **Application rejects attempts to submit the same item with more than one request**

Weaknesses in Token Generation

I figured out a way to hack any of Facebook's 2 billion accounts, and they paid me a $15,000 bounty for it

- **Password reset uses a 6-digit code**
- **Guesses are rate-limited on www.facebook.com**
- **But not on beta.facebook.com**
- **Got in with Burp Intruder**

Link Ch 7k

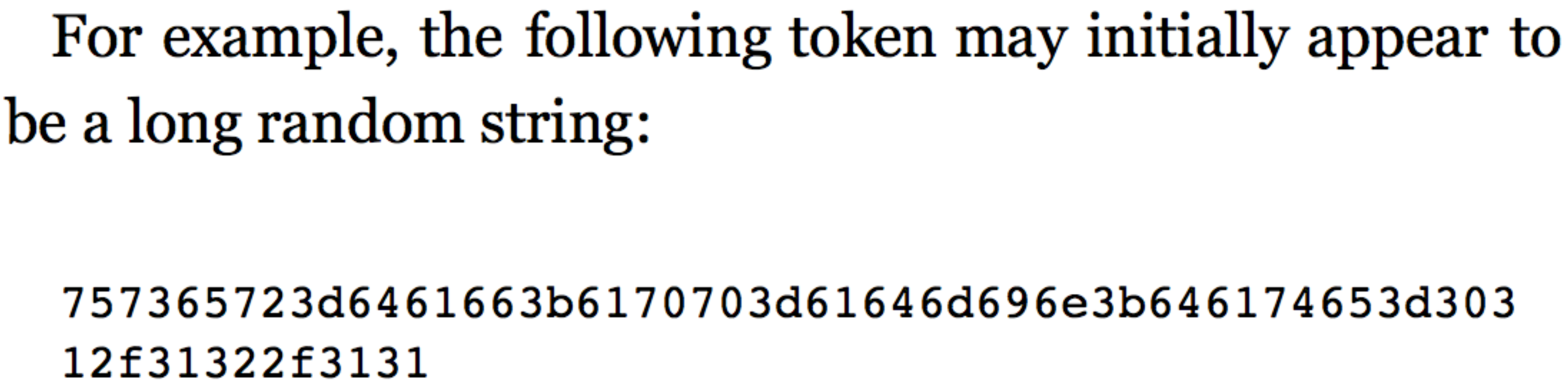
Token Use Cases

- **Password recovery tokens sent to user's email address**
- **Tokens in hidden form fields to prevent cross-site forgery attacks**
- **Tokens used to grant one-time access to protected resources**
- **Persistent tokens used to "remember me"**
- **Tokens used to allow customers of shopping application to see the status of an order**

Unpredictability

- **The security of the application depends on all those tokens being unpredictable**

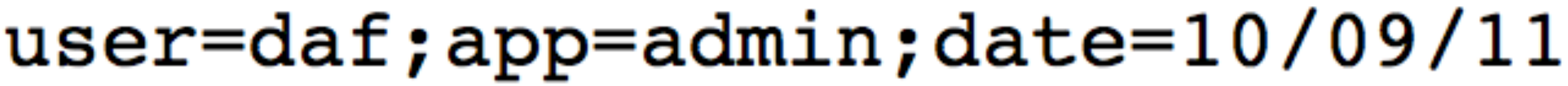
Meaningful Tokens



For example, the following token may initially appear to be a long random string:

```
757365723d6461663b6170703d61646d696e3b646174653d303
12f31322f3131
```

- **It's just hexadecimal ASCII for**

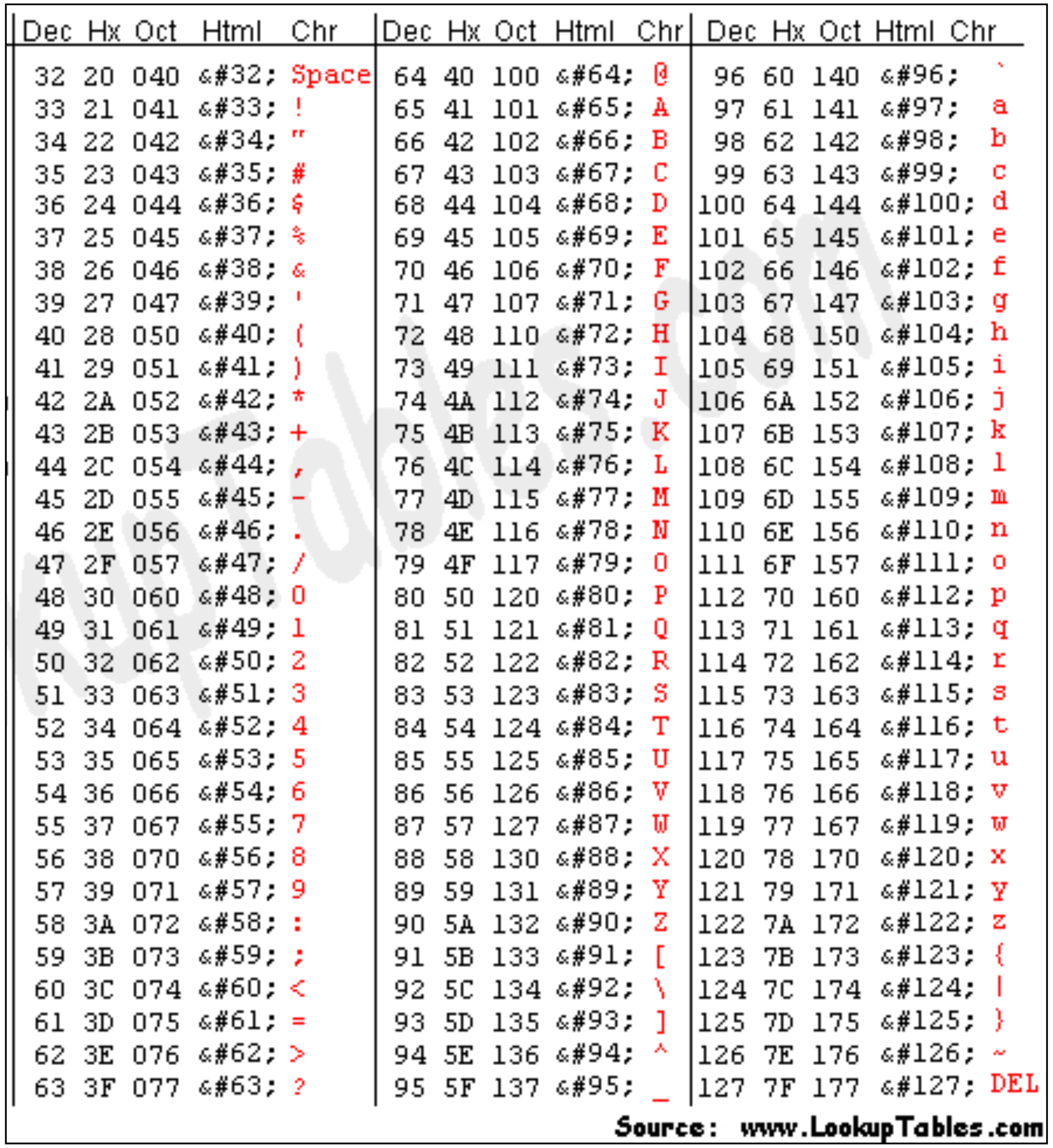


```
user=daf;app=admin;date=10/09/11
```

- **Easy to guess other token values, like user=admin**

ASCII

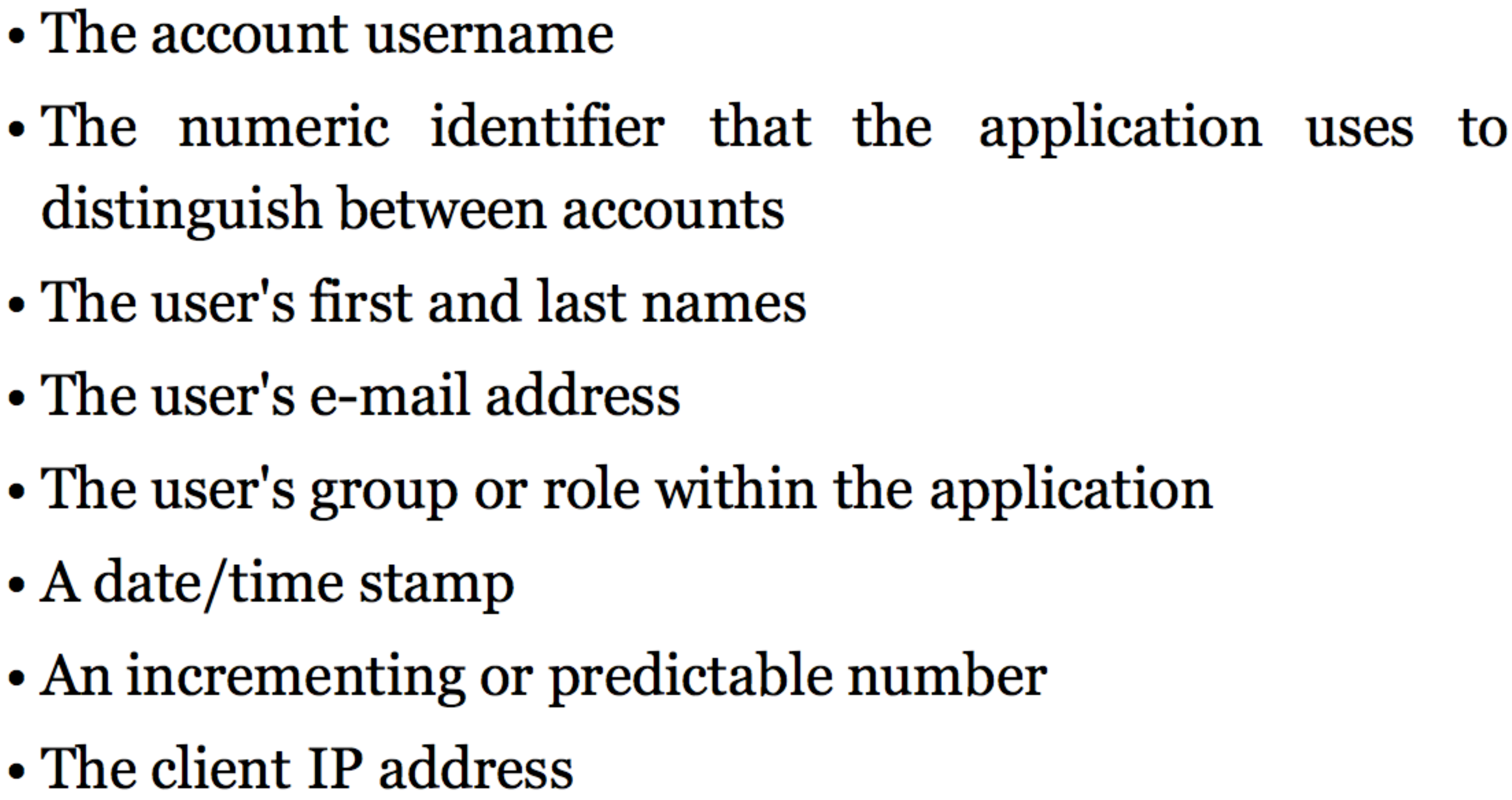
- 75 73 65 72
- u s e r



Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
36	24	044	&#36;	$	68	44	104	&#68;	D	100	64	144	&#100;	d
37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
59	3B	073	&#59;	;	91	5B	133	&#91;	[	123	7B	173	&#123;	{
60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	|
61	3D	075	&#61;	=	93	5D	135	&#93;	]	125	7D	175	&#125;	}
62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~
63	3F	077	&#63;	?	95	5F	137	&#95;	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Components in Structured Tokens



- The account username
- The numeric identifier that the application uses to distinguish between accounts
- The user's first and last names
- The user's e-mail address
- The user's group or role within the application
- A date/time stamp
- An incrementing or predictable number
- The client IP address

Common Encoding Schemes

- **XOR**
- **Base64**
- **Hexadecimal ASCII codes**

Hack Steps

- **Obtain a single token**
- **Modify it in systematic ways**
- **Change it one byte at a time, or one bit at a time**
- **Resubmit it to see if it's still accepted**
- **Some fields may be ignored**
- **Burp Intruder's "char frobber" function**

Hack Steps

- **Log in as several users at different times**
- **Record the tokens**
- **If you can, register similar usernames like A, AA, AAA, AAAB, etc.**
- **Try similar series for other data, such as email addresses**

Hack Steps

- **Analyze the tokens for correlations**
- **Look for encoding or obfuscation**
- **A series of repeating letters like AAA will produce repeating encoded characters like zzz if XOR is used**
- **Base64 often ends in = or ==**

Hack Steps

- **Use the patterns you've found to try guessing tokens of other users**
- **Find a page that is session-dependent**
- **Use Burp Intruder to send many requests using guessed tokens**
- **Monitor results to see if any pages load correctly**

Kahoot!

Ch 7a

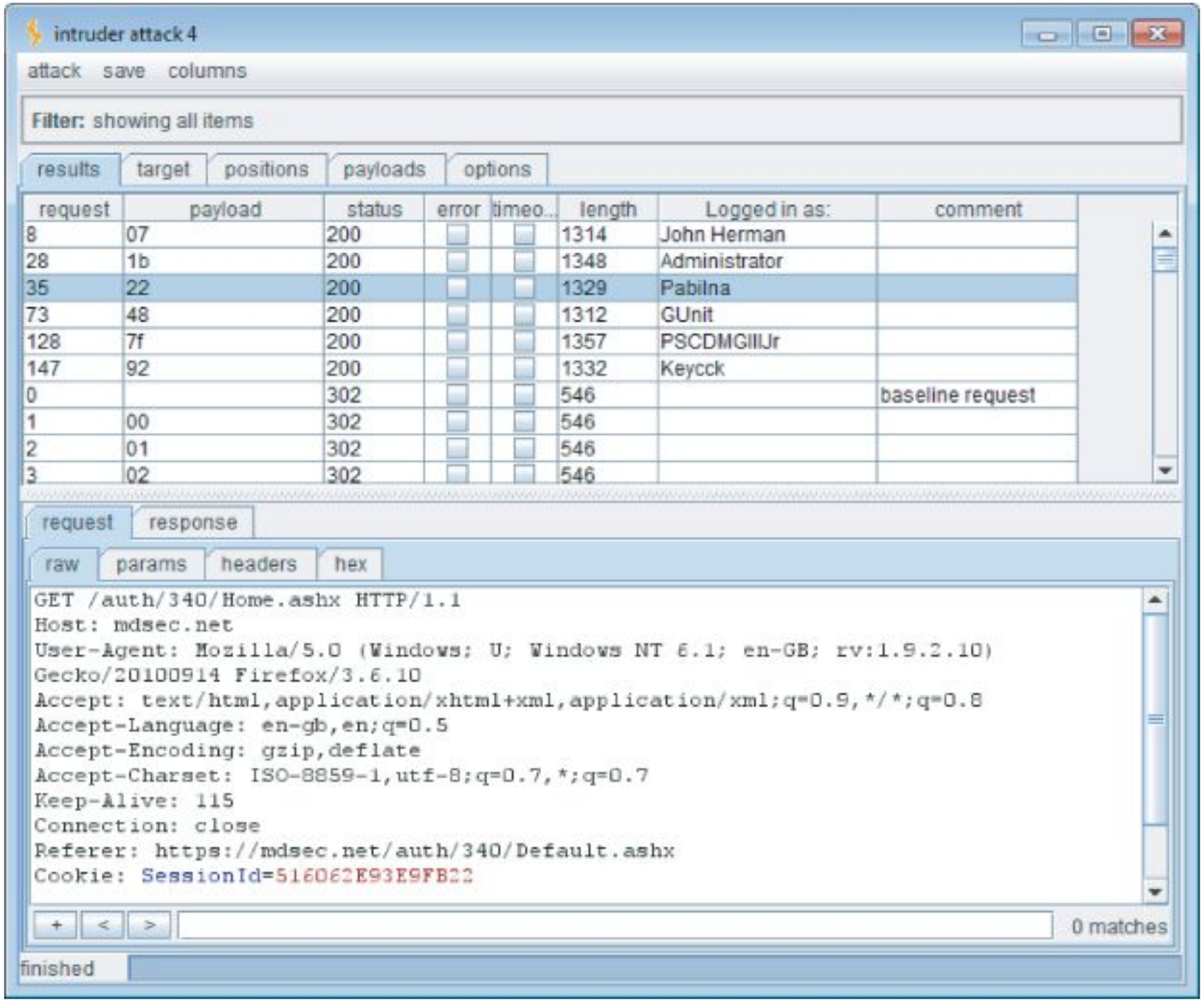
Predictable Tokens

Patterns

- From a sample of tokens, it may be possible to predict valid tokens
- Commercial implementations such as web servers or application platforms may be more vulnerable
    - Because it's easier to gather a large sample of tokens, from your own test system

Sequential Tokens

- **Burp trying sequential payloads**
- **Winners shown at to the top**



intruder attack 4
attack save columns
Filter: showing all items
results target positions payloads options

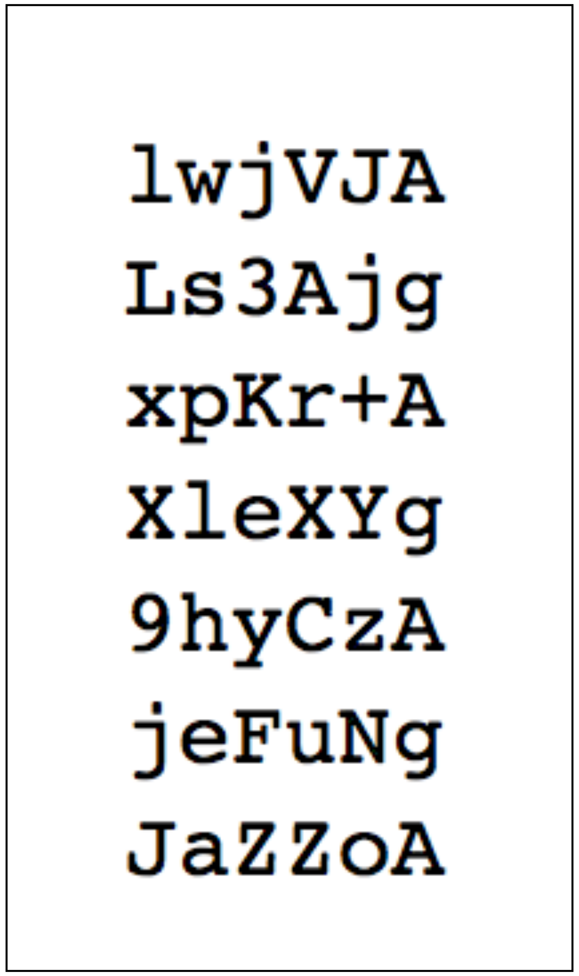
request	payload	status	error	timeo...	length	Logged in as:	comment
8	07	200	□	□	1314	John Herman	
28	1b	200	□	□	1348	Administrator	
35	22	200	□	□	1329	Pabilna	
73	48	200	□	□	1312	GUnit	
128	7f	200	□	□	1357	PSCDMGIlUr	
147	92	200	□	□	1332	Keycck	
0		302	□	□	546		baseline request
1	00	302	□	□	546		
2	01	302	□	□	546		
3	02	302	□	□	546		

```
GET /auth/340/Home.ashx HTTP/1.1
Host: mdsec.net
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-GB; rv:1.9.2.10)
Gecko/20100914 Firefox/3.6.10
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-gb,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: close
Referer: https://mdsec.net/auth/340/Default.ashx
Cookie: SessionId=516062E93E9FB22
```

Three Sources of Predictable Session Tokens

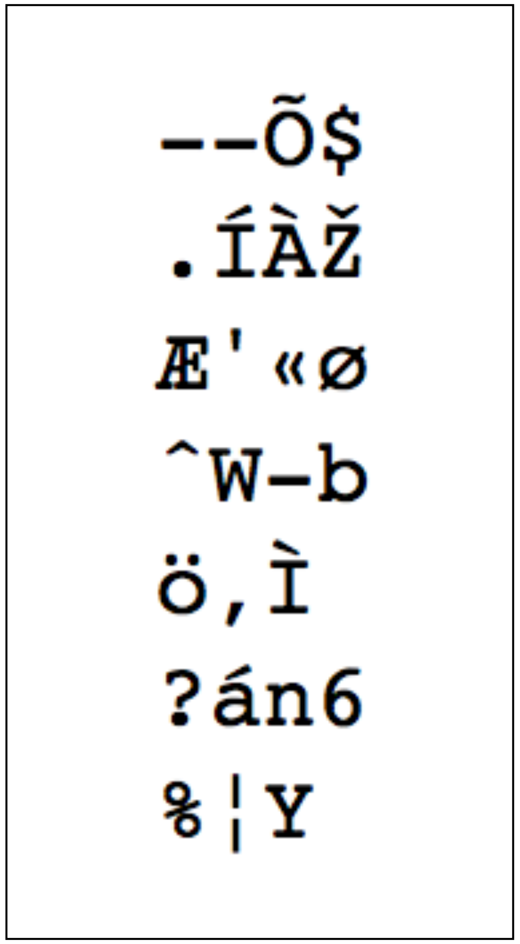
- **Concealed sequences**
- **Time dependency**
- **Weak random number generation**

Concealed Sequences



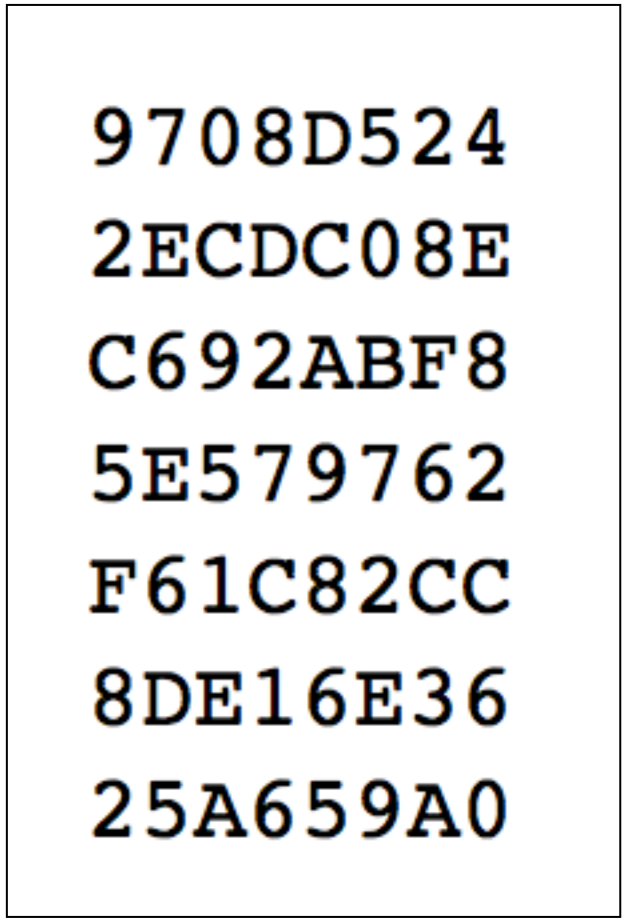
```
lwjVJA
Ls3Ajg
xpKr+A
XleXYg
9hyCzA
jeFuNg
JaZZoA
```

- **This sequence appears to be Base64-encoded**
- **Decodes to the gibberish on the right**



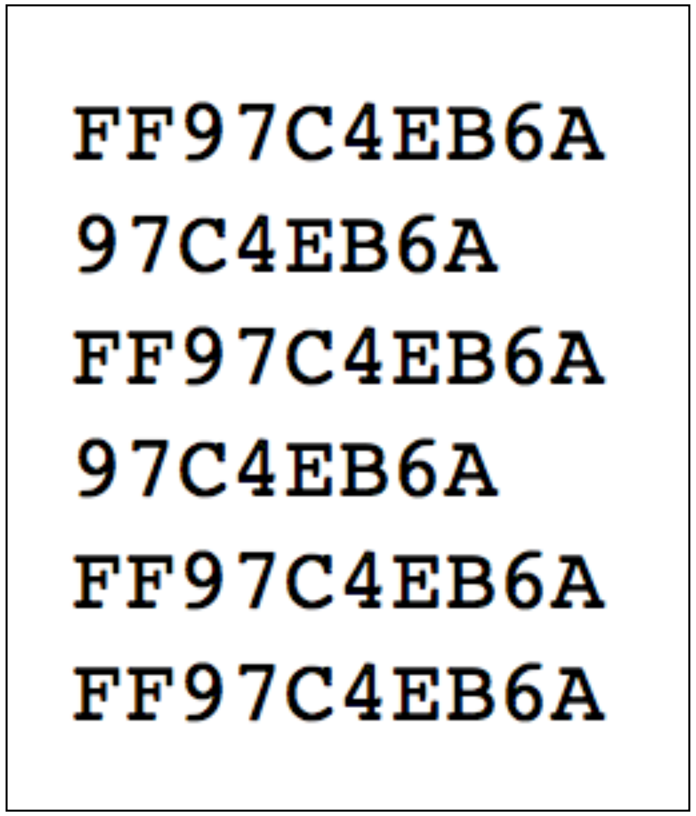
```
--Õ$
.ÍÀŽ
Æ'«ø
^W-b
ö,Ì
?án6
%¦Y
```

Hexadecimal Form



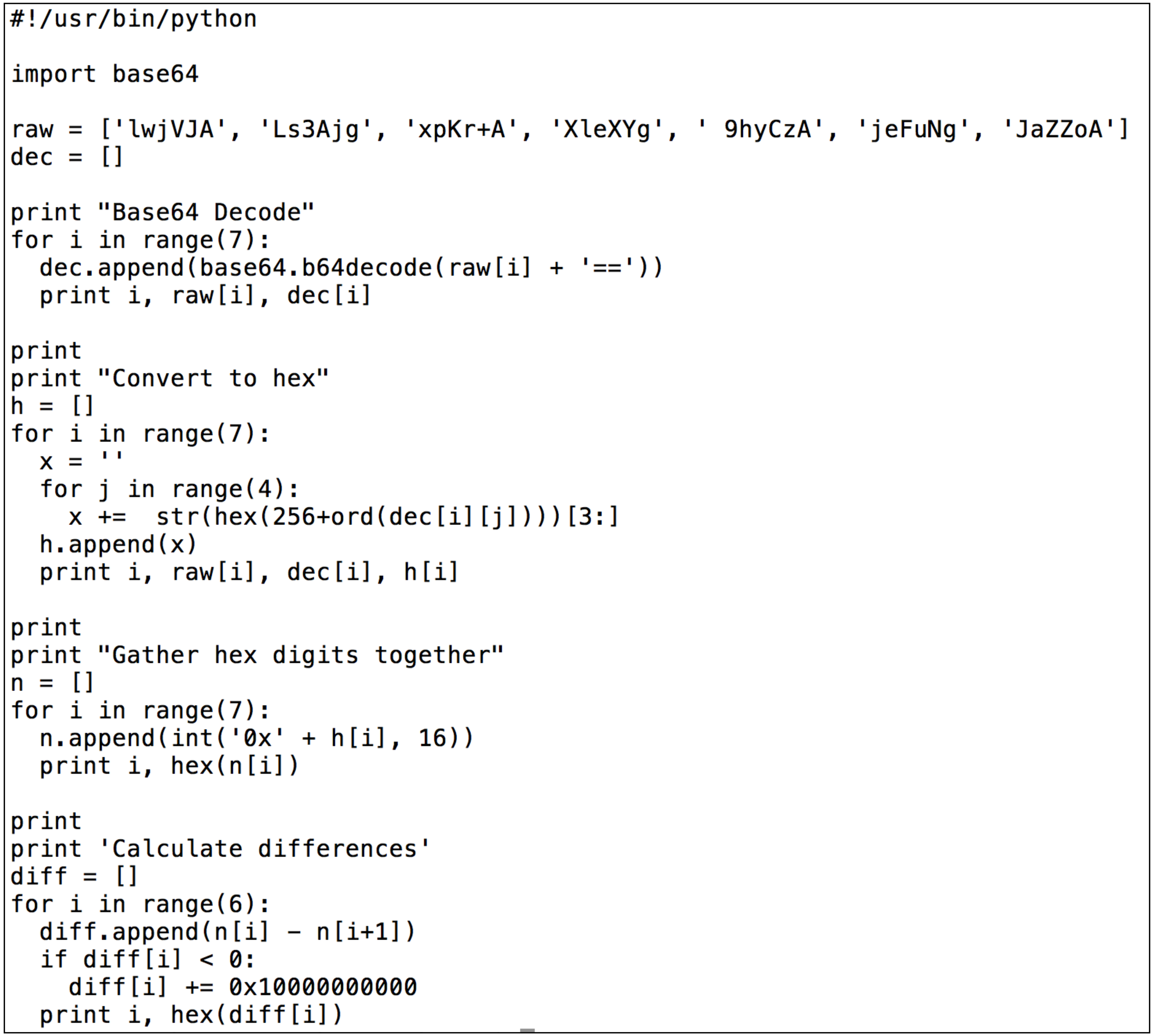
```
9708D524
2ECDC08E
C692ABF8
5E579762
F61C82CC
8DE16E36
25A659A0
```

- **Render the tokens as hexadecimal numbers**
- **Calculate difference between sequential tokens**
- **For negative differences, add 0x100000000000 so it starts with FF**



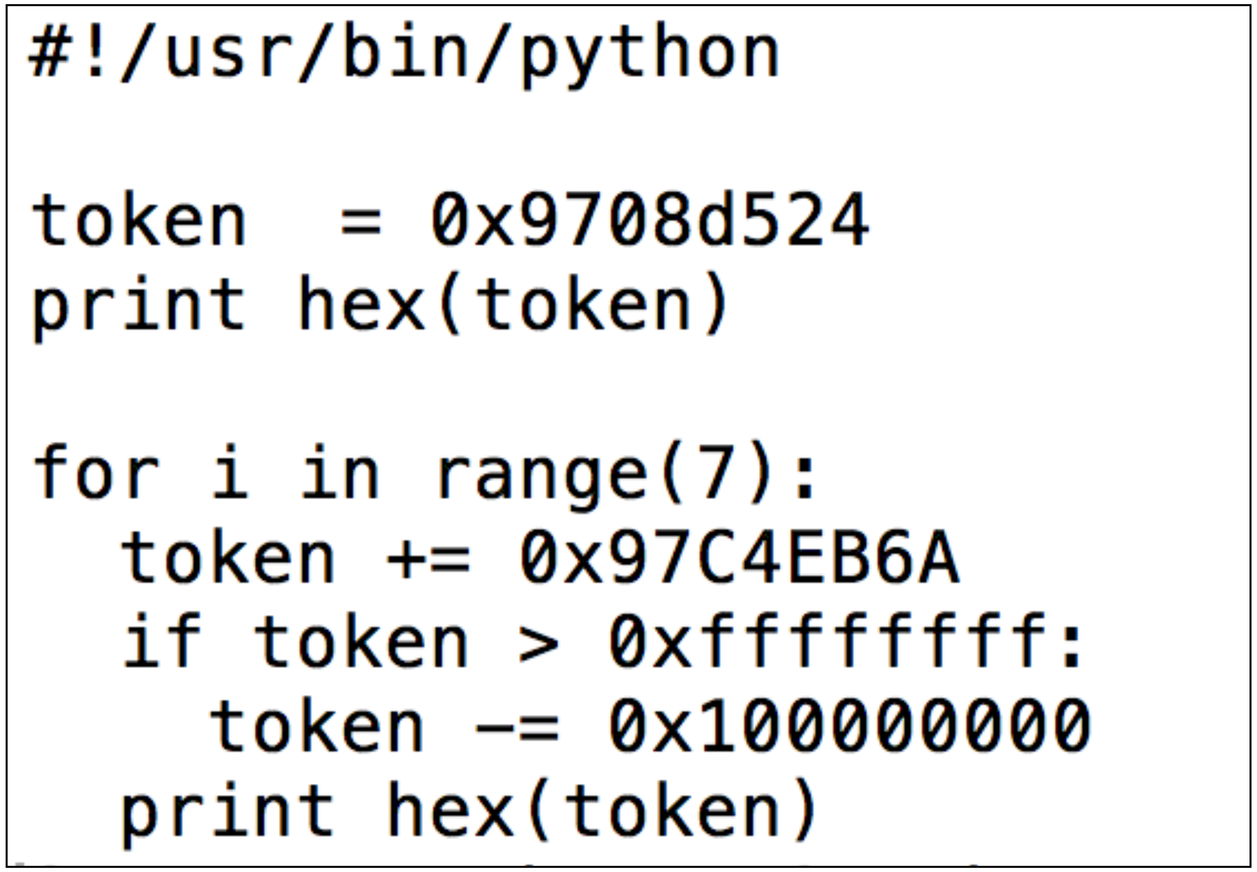
```
FF97C4EB6A
97C4EB6A
FF97C4EB6A
97C4EB6A
FF97C4EB6A
FF97C4EB6A
```

Script to Decode

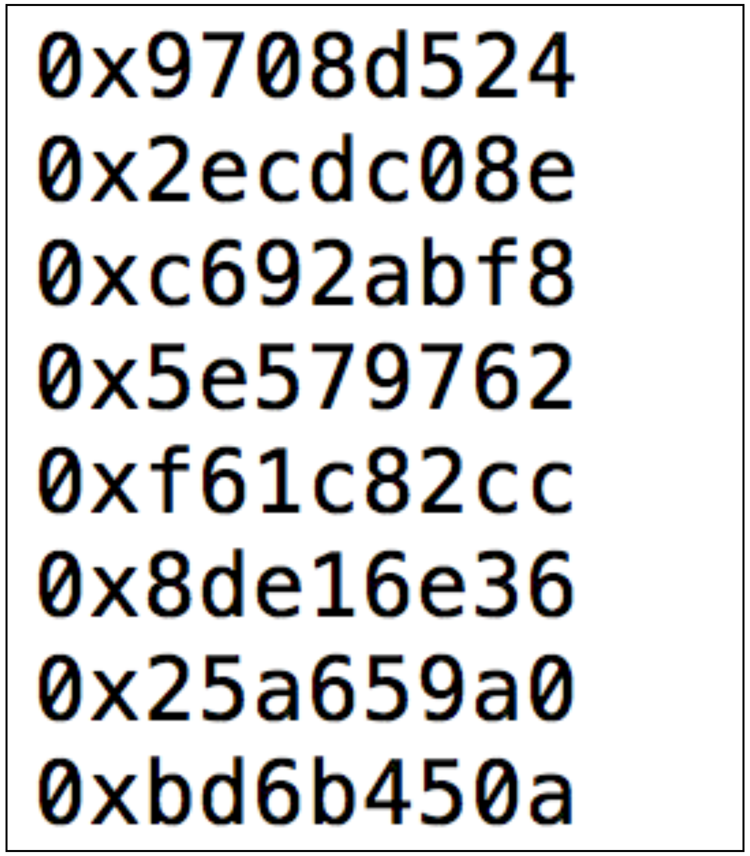


```
#!/usr/bin/python

import base64

raw = ['lwjVJA', 'Ls3Ajg', 'xpKr+A', 'XleXYg', ' 9hyCzA', 'jeFuNg', 'JaZZoA']
dec = []

print "Base64 Decode"
for i in range(7):
  dec.append(base64.b64decode(raw[i] + '=='))
  print i, raw[i], dec[i]

print
print "Convert to hex"
h = []
for i in range(7):
  x = ''
  for j in range(4):
    x +=  str(hex(256+ord(dec[i][j])))[3:]
  h.append(x)
  print i, raw[i], dec[i], h[i]

print
print "Gather hex digits together"
n = []
for i in range(7):
  n.append(int('0x' + h[i], 16))
  print i, hex(n[i])

print
print 'Calculate differences'
diff = []
for i in range(6):
  diff.append(n[i] - n[i+1])
  if diff[i] < 0:
    diff[i] += 0x100000000
  print i, hex(diff[i])
```

Generate Valid Tokens

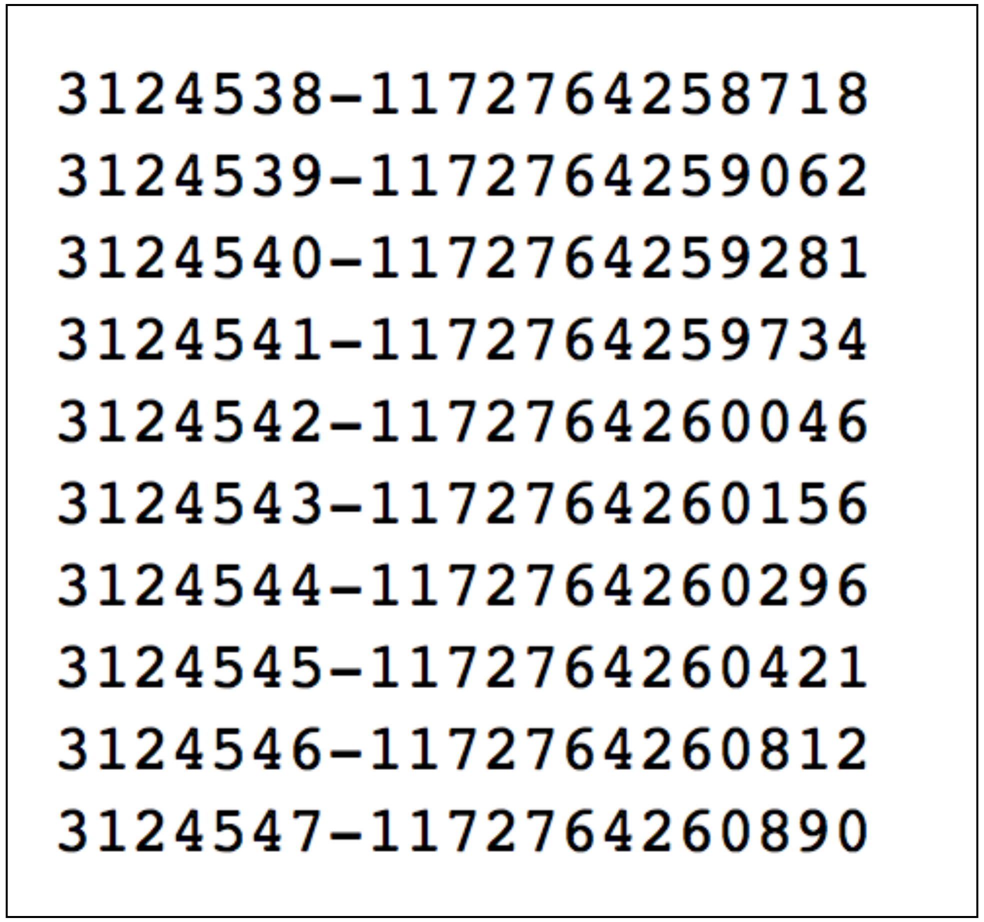


```
#!/usr/bin/python

token  = 0x9708d524
print hex(token)

for i in range(7):
  token += 0x97C4EB6A
  if token > 0xffffffff:
    token -= 0x100000000
  print hex(token)
```



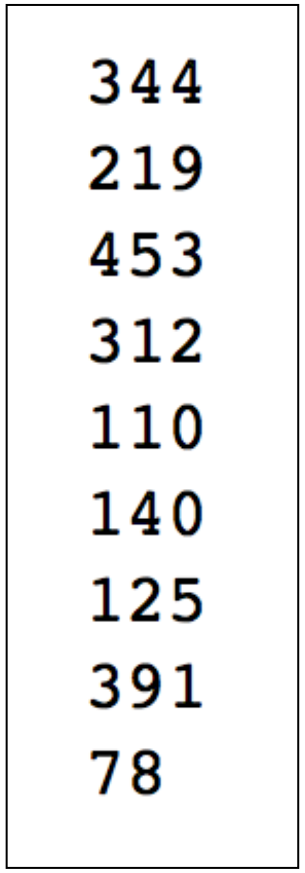
```
0x9708d524
0x2ecdc08e
0xc692abf8
0x5e579762
0xf61c82cc
0x8de16e36
0x25a659a0
0xbd6b450a
```

Time Dependency



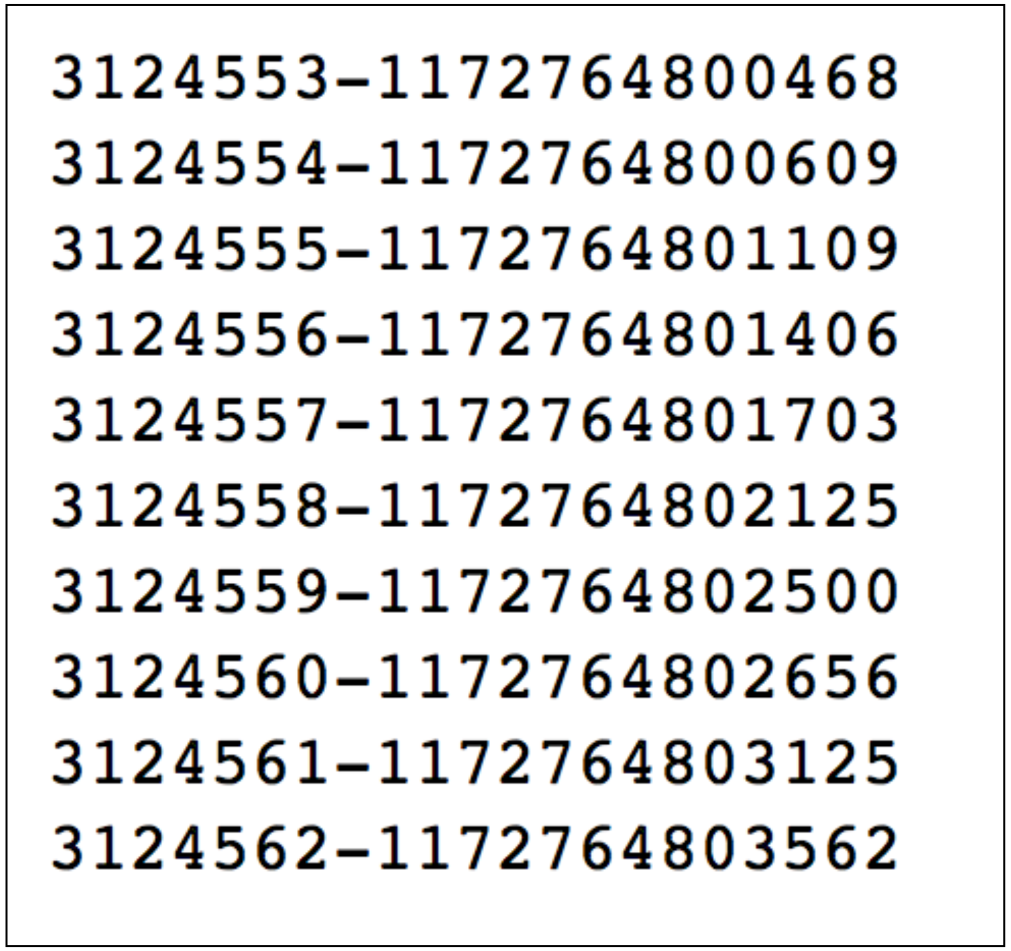
```
3124538-1172764258718
3124539-1172764259062
3124540-1172764259281
3124541-1172764259734
3124542-1172764260046
3124543-1172764260156
3124544-1172764260296
3124545-1172764260421
3124546-1172764260812
3124547-1172764260890
```

- **Left number simply increments by 1 each time**
- **Right number moves up by a varying value, as shown on the right**



```
344
219
453
312
110
140
125
391
78
```

Another Sample



```
3124553-1172764800468
3124554-1172764800609
3124555-1172764801109
3124556-1172764801406
3124557-1172764801703
3124558-1172764802125
3124559-1172764802500
3124560-1172764802656
3124561-1172764803125
3124562-1172764803562
```

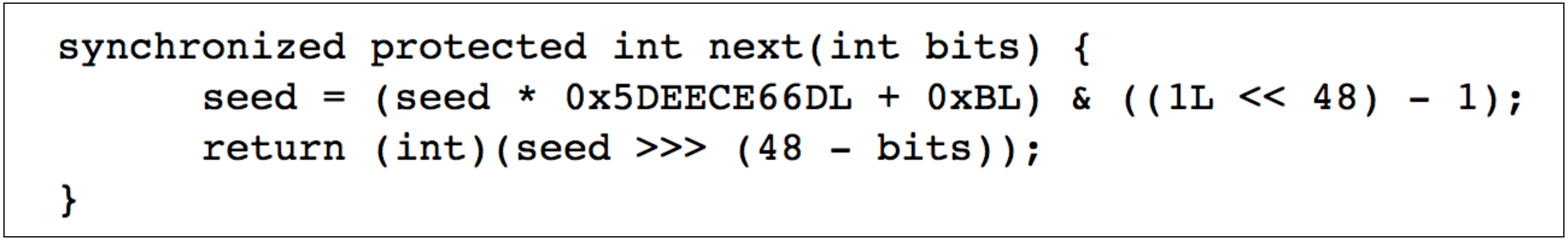
- **Ten minutes later**
- **First number has jumped by 6**
- **Second number has jumped by 539578**
- **Ten minutes = 600 sec = 600,000 milliseconds**

Attack Steps

- **Poll the server frequently to gather session tokens**
- **When first number increases by more than 1, there's another user in between**
- **We know the second number will be between the two numbers we have**
  - **Simply brute-force the value**

Weak Random Number Generation

- **Jetty is a Java-based Web server**
- **Calculates pseudorandom session tokens with a "linear congruential generator"**
- **Multiplies previous number by a constant, adds another constant, truncates to 48 bits**
- **Given one value, all others can be predicted**



```
synchronized protected int next(int bits) {
      seed = (seed * 0x5DEECE66DL + 0xBL) & ((1L << 48) - 1);
      return (int)(seed >>> (48 - bits));
}
```

PHP 5.3.2 and Earlier

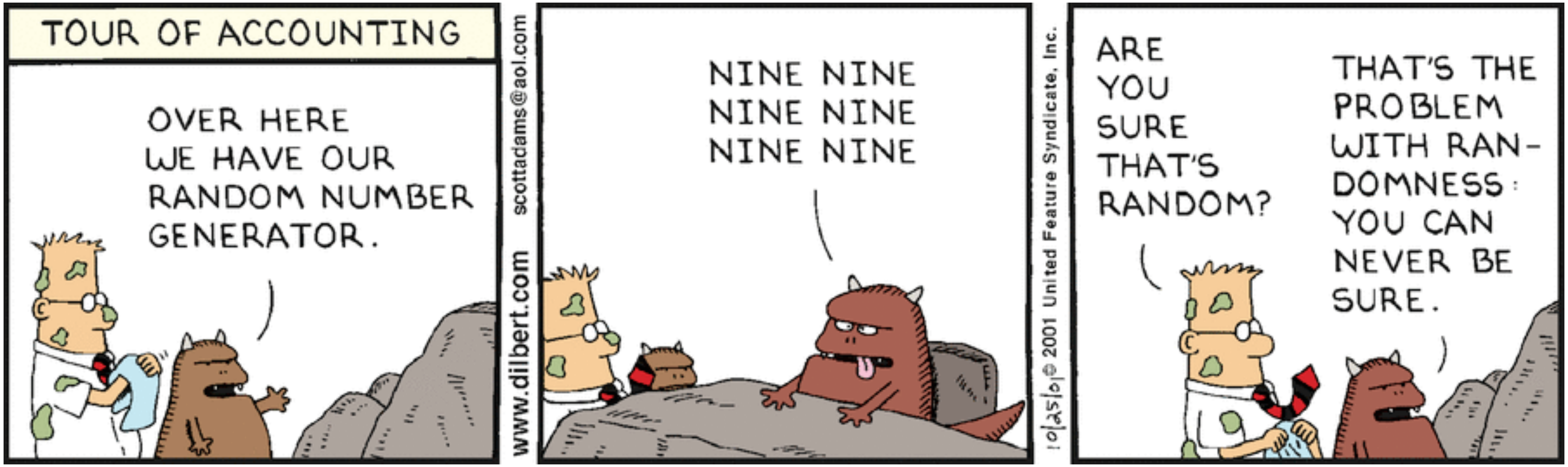
- **Session token generated from**
  - **Client's IP address**
  - **Epoch time at token creation**
  - **Microseconds at token creation**
  - **Linear congruential generator**

phpwn

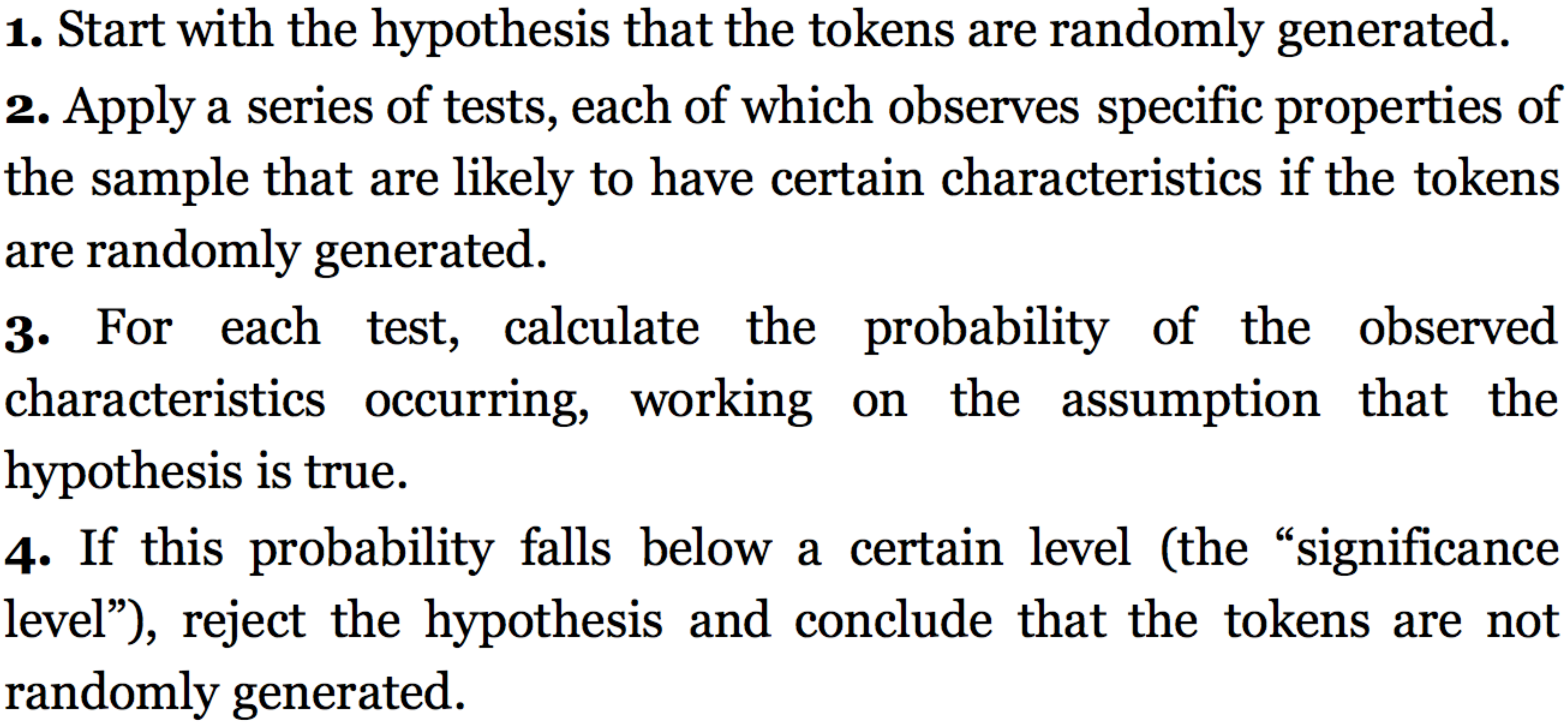
- **The vulnerability was found in 2001**
- **But no one wrote a practical attack tool until Samy Kamkar in 2010**
  - **Link Ch 7b**

Samy Kamkar became a notorious hacker when he was 19. He created the fastest-spreading computer worm of its time. Unfortunately, it also crashed MySpace, and Kamkar was banned from the Internet.

Testing the Quality of Randomness


TOUR OF ACCOUNTING
OVER HERE WE HAVE OUR RANDOM NUMBER GENERATOR.
scottadams@aol.com
www.dilbert.com
NINE NINE NINE NINE NINE NINE
10/25/01 © 2001 United Feature Syndicate, Inc.
ARE YOU SURE THAT'S RANDOM?
THAT'S THE PROBLEM WITH RANDOMNESS: YOU CAN NEVER BE SURE.

Algorithm



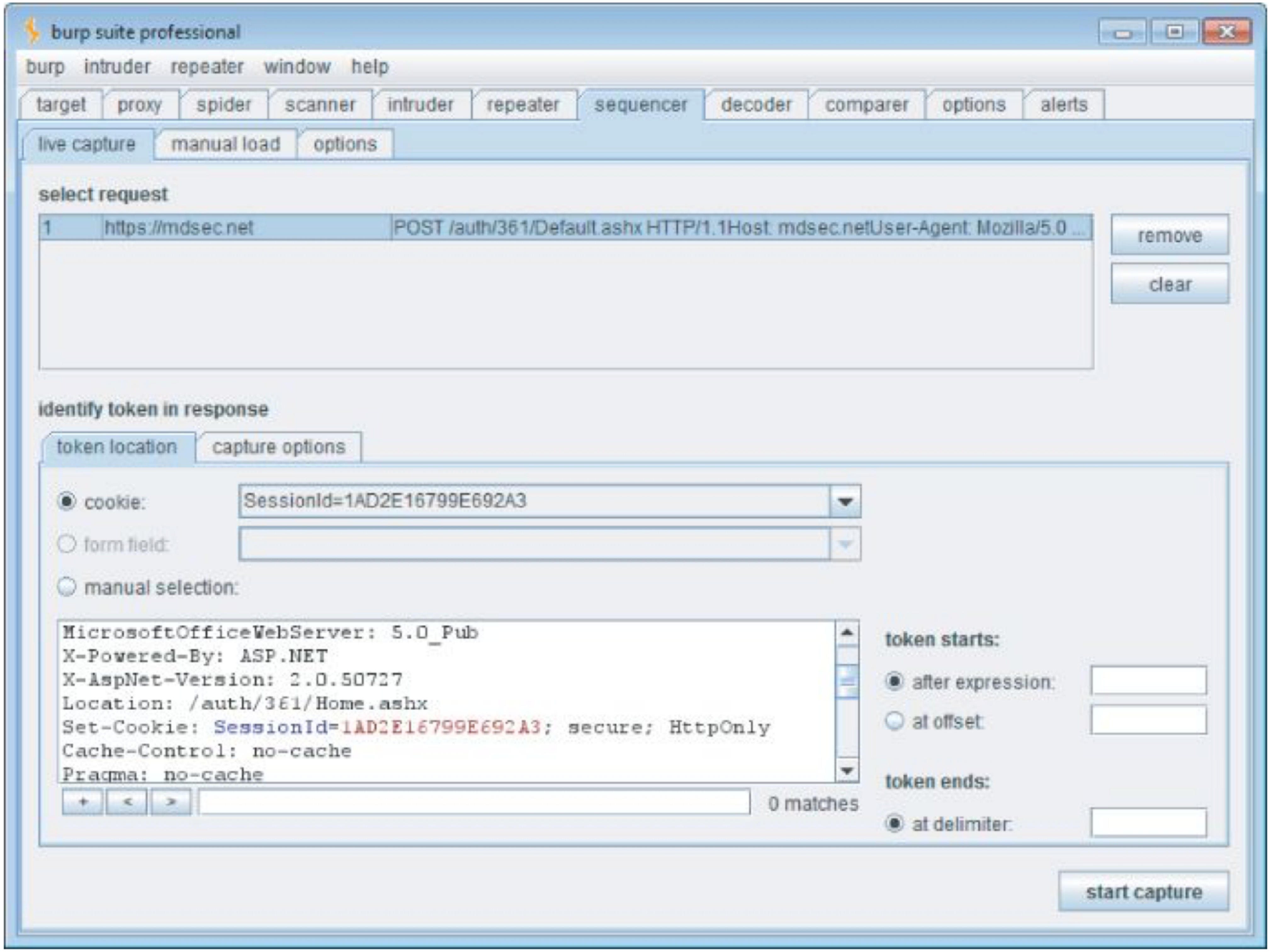
1. Start with the hypothesis that the tokens are randomly generated.

2. Apply a series of tests, each of which observes specific properties of the sample that are likely to have certain characteristics if the tokens are randomly generated.

3. For each test, calculate the probability of the observed characteristics occurring, working on the assumption that the hypothesis is true.

4. If this probability falls below a certain level (the "significance level"), reject the hypothesis and conclude that the tokens are not randomly generated.

Burp Sequencer



burp suite professional
burp intruder repeater window help
target proxy spider scanner intruder repeater sequencer decoder comparer options alerts
live capture manual load options
remove
clear
token location capture options
start capture

select request

1	https://mdsec.net	POST /auth/361/Default.ashx HTTP/1.1Host: mdsec.netUser-Agent: Mozilla/5.0 ...

identify token in response

● cookie: SessionId=1AD2E16799E692A3

○ form field:

○ manual selection:

```
MicrosoftOfficeWebServer: 5.0_Pub
X-Powered-By: ASP.NET
X-AspNet-Version: 2.0.50727
Location: /auth/361/Home.ashx
Set-Cookie: SessionId=1AD2E16799E692A3; secure; HttpOnly
Cache-Control: no-cache
Pragma: no-cache
```

+ < > 0 matches

token starts:

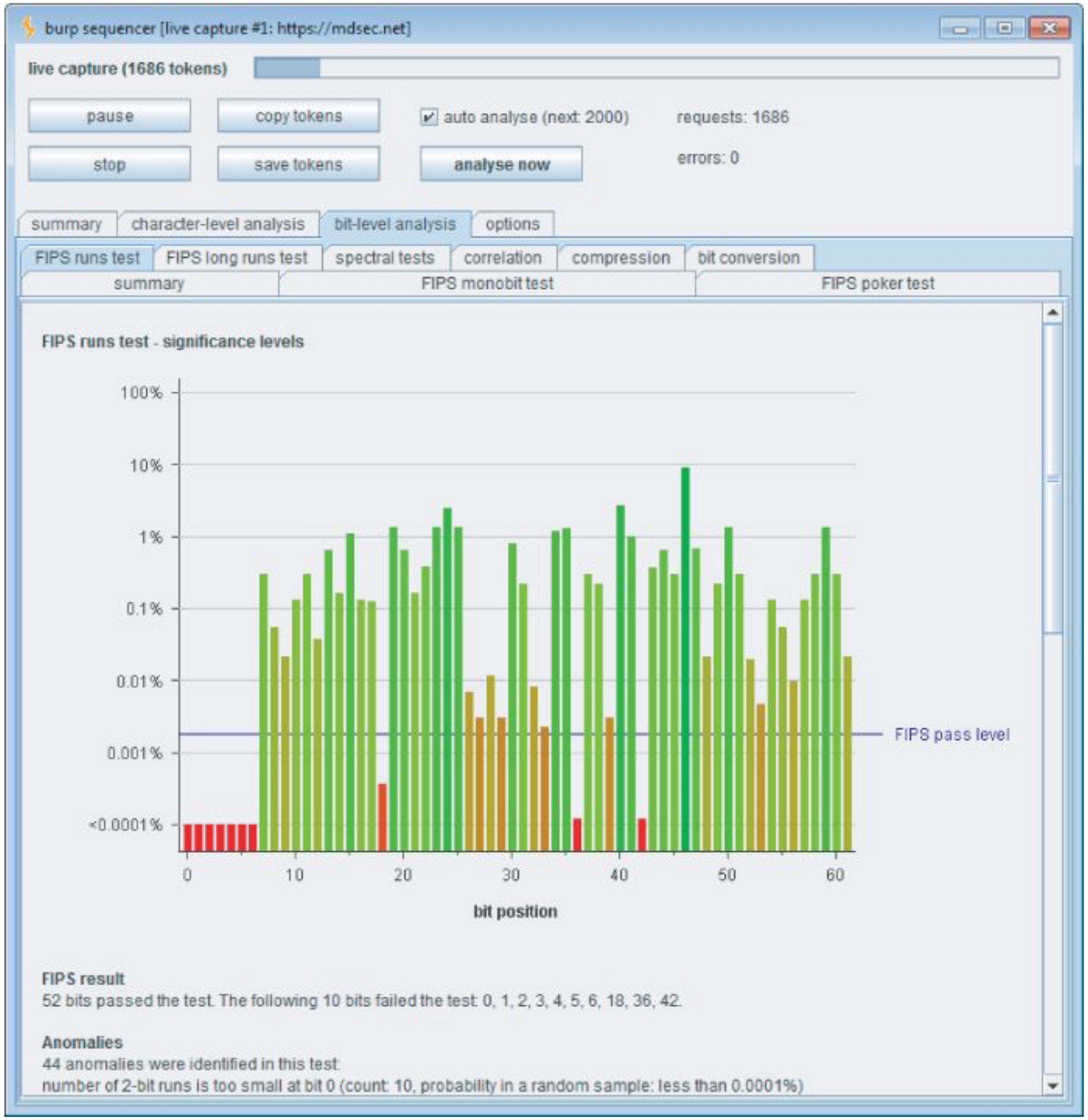
● after expression:

○ at offset:

token ends:

● at delimiter:

Results: Red Bits are Non-Random



FIPS result

52 bits passed the test. The following 10 bits failed the test: 0, 1, 2, 3, 4, 5, 6, 18, 36, 42.

Anomalies

44 anomalies were identified in this test:

number of 2-bit runs is too small at bit 0 (count: 10, probability in a random sample: less than 0.0001%)

Encrypted Tokens

Design Goal

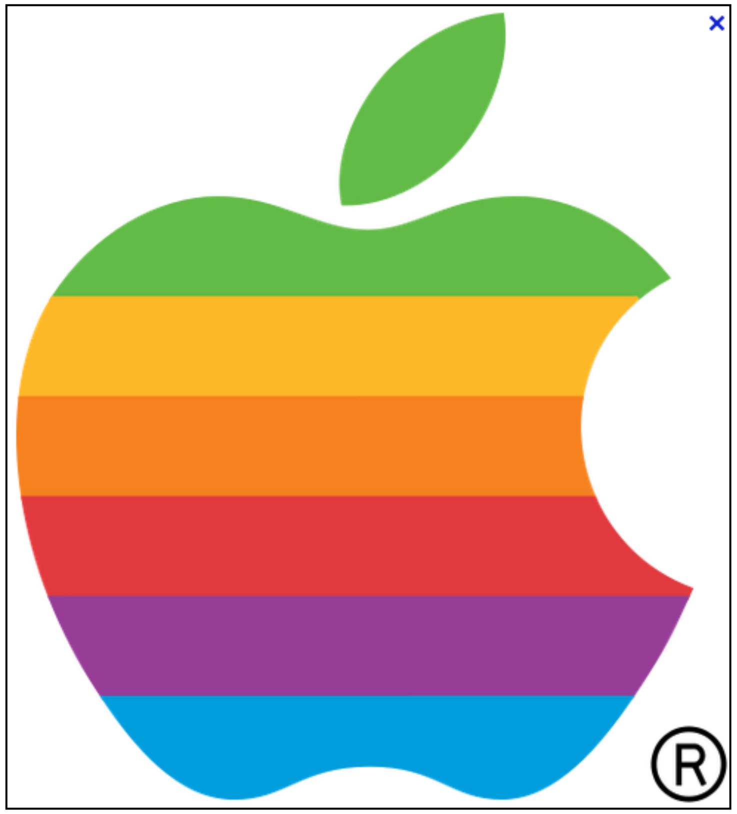
- **Token built from meaningful content, such as username**
- **Encrypted with a secret key not known to the attacker**
- **Sounds good, but sometimes the attacker can tamper with token's meaningful values without decrypting them**

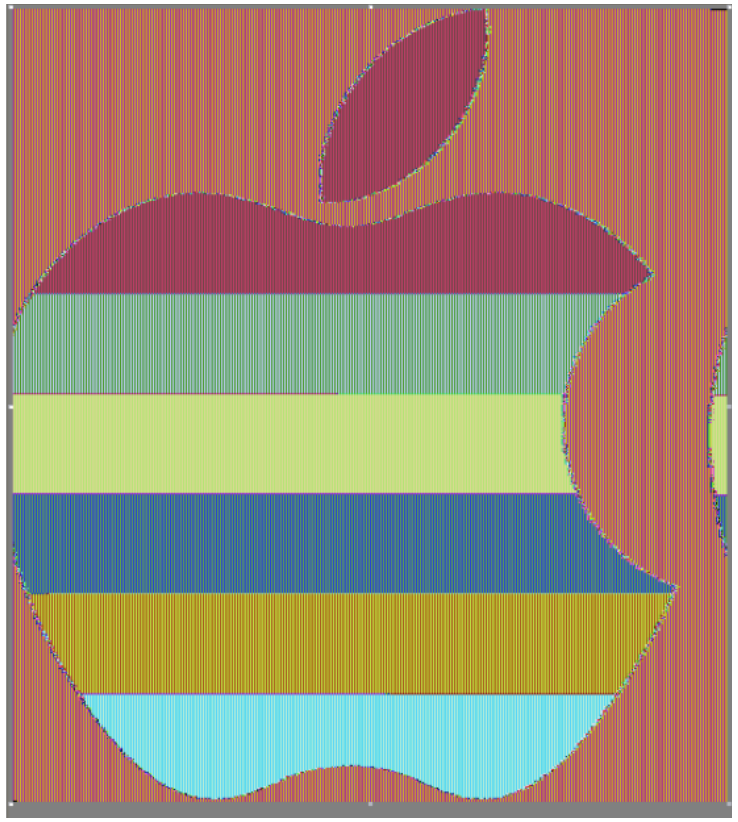
ECB Ciphers

- **Electronic Code Book**
- **Input broken up into blocks, often 8 bytes long**
- **Symmetric encryption, no randomness**
- **Each input block encodes to a single output block**
- **This preserves patterns in input**

Image Encrypted with ECB

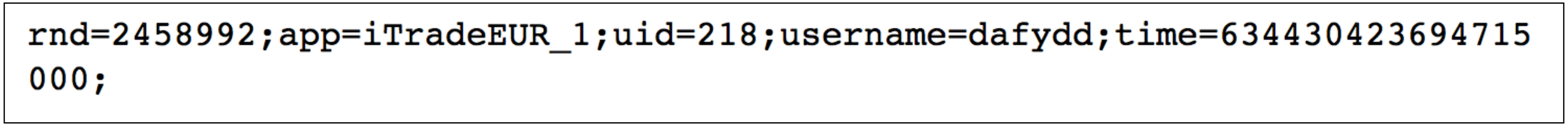
- **Patterns preserved in output**





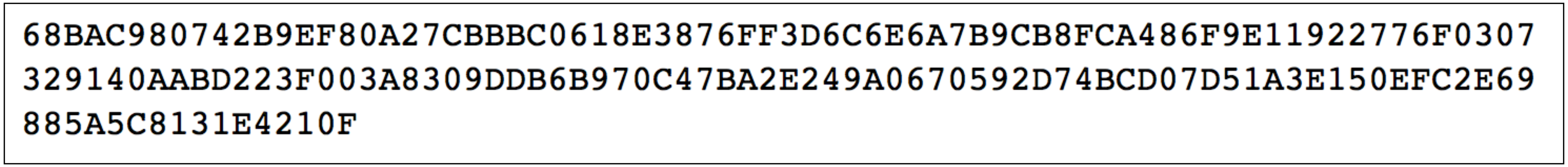
Example of ECB

- **Token contains several meaningful fields, including numeric user id**



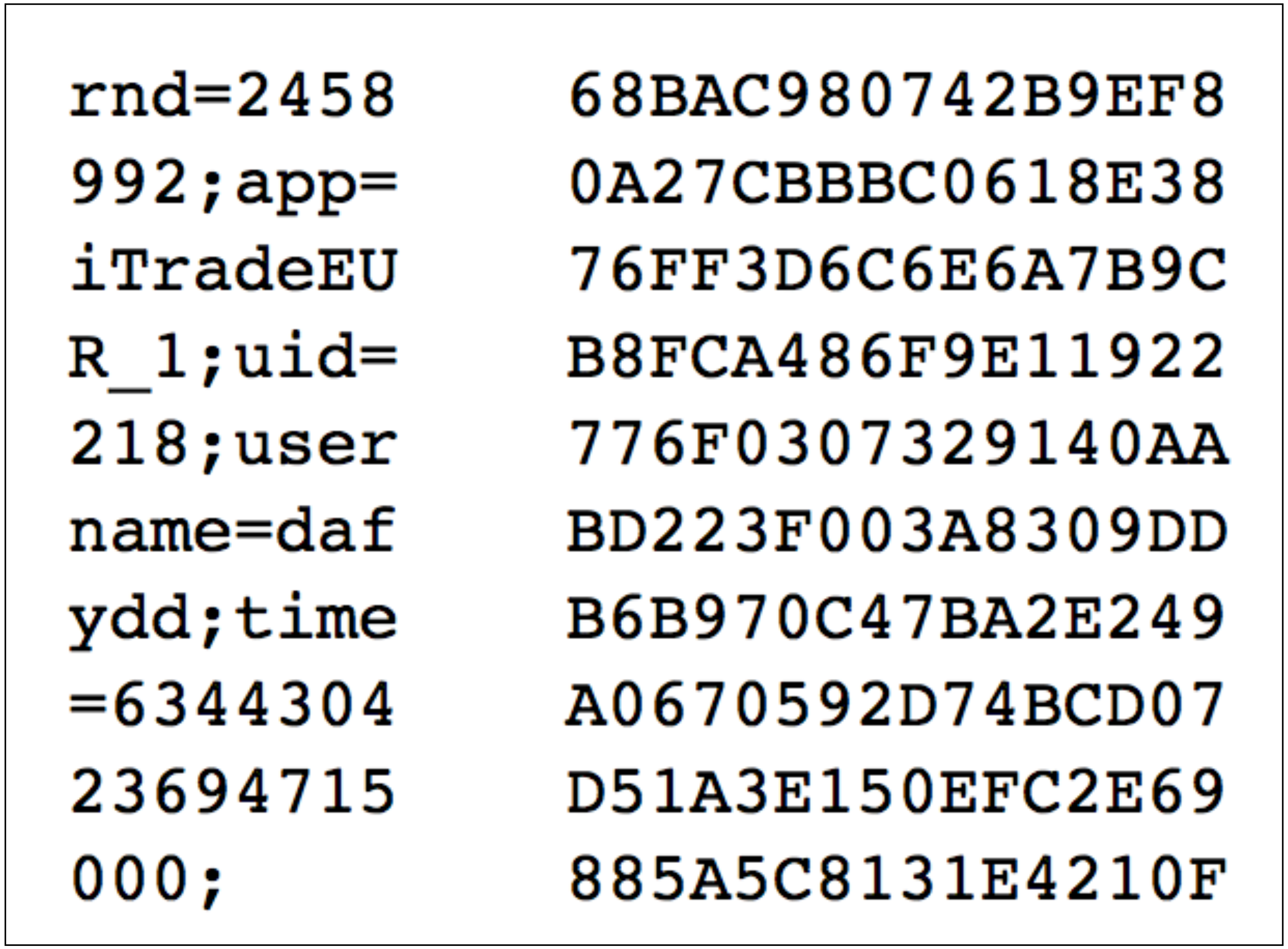
```
rnd=2458992;app=iTradeEUR_1;uid=218;username=dafydd;time=634430423694715
000;
```

- **Encrypted form appears meaningless**



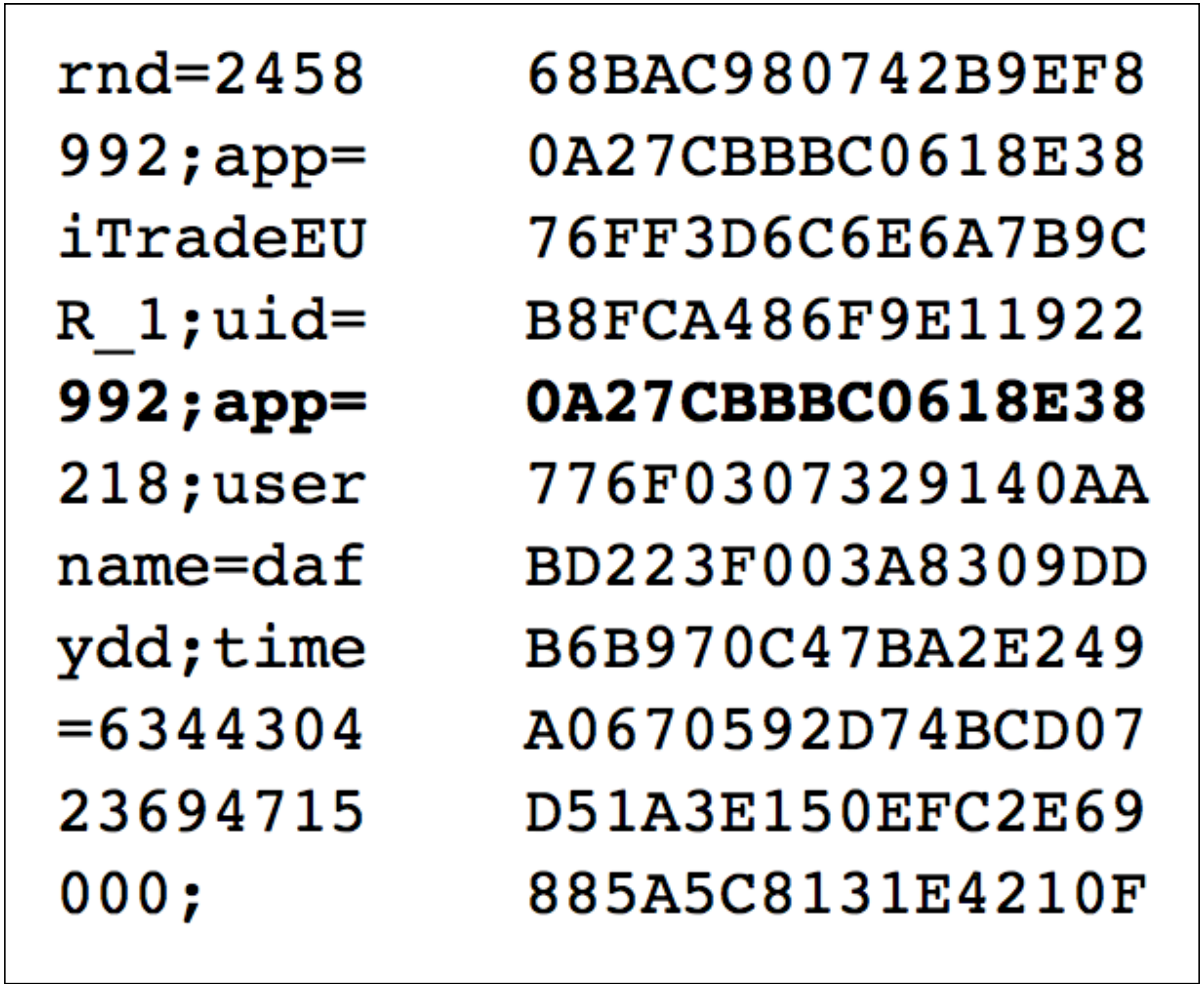
```
68BAC980742B9EF80A27CBBBC0618E3876FF3D6C6E6A7B9CB8FCA486F9E11922776F0307
329140AABD223F003A8309DDB6B970C47BA2E249A0670592D74BCD07D51A3E150EFC2E69
885A5C8131E4210F
```

8-Byte Blocks



```
rnd=2458      68BAC980742B9EF8
992;app=      0A27CBBBC0618E38
iTradeEU      76FF3D6C6E6A7B9C
R_1;uid=      B8FCA486F9E11922
218;user      776F0307329140AA
name=daf      BD223F003A8309DD
ydd;time      B6B970C47BA2E249
=6344304      A0670592D74BCD07
23694715      D51A3E150EFC2E69
000;          885A5C8131E4210F
```

Copy a Whole Block

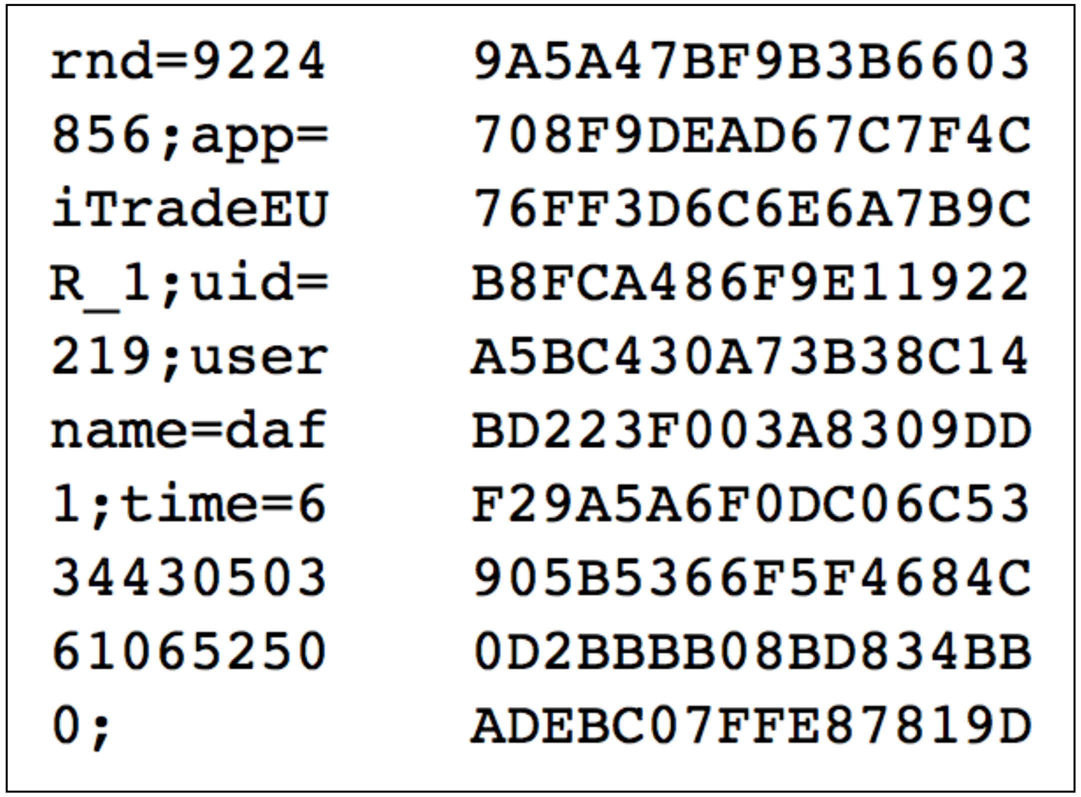


```
rnd=2458       68BAC980742B9EF8
992;app=       0A27CBBBC0618E38
iTradeEU       76FF3D6C6E6A7B9C
R_1;uid=       B8FCA486F9E11922
992;app=       0A27CBBBC0618E38
218;user       776F0307329140AA
name=daf       BD223F003A8309DD
ydd;time       B6B970C47BA2E249
=6344304       A0670592D74BCD07
23694715       D51A3E150EFC2E69
000;           885A5C8131E4210F
```

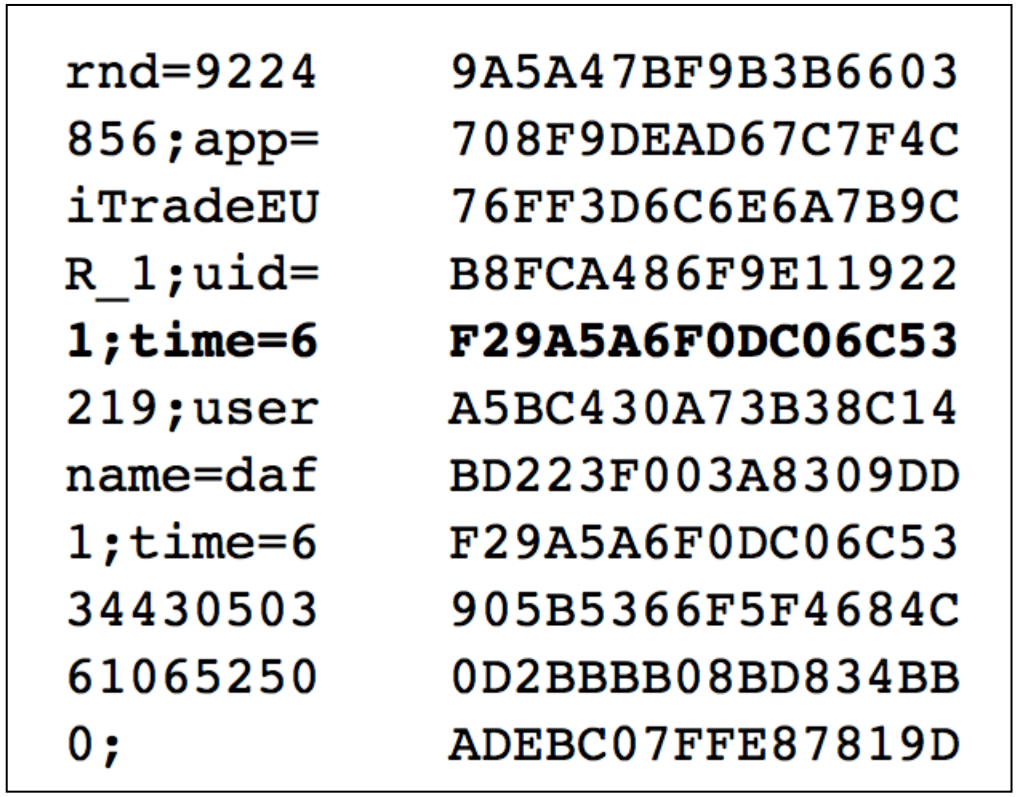
- **Token is now for user 992**

Register a New User "daf1"

- **Now attacker can target uid=1**



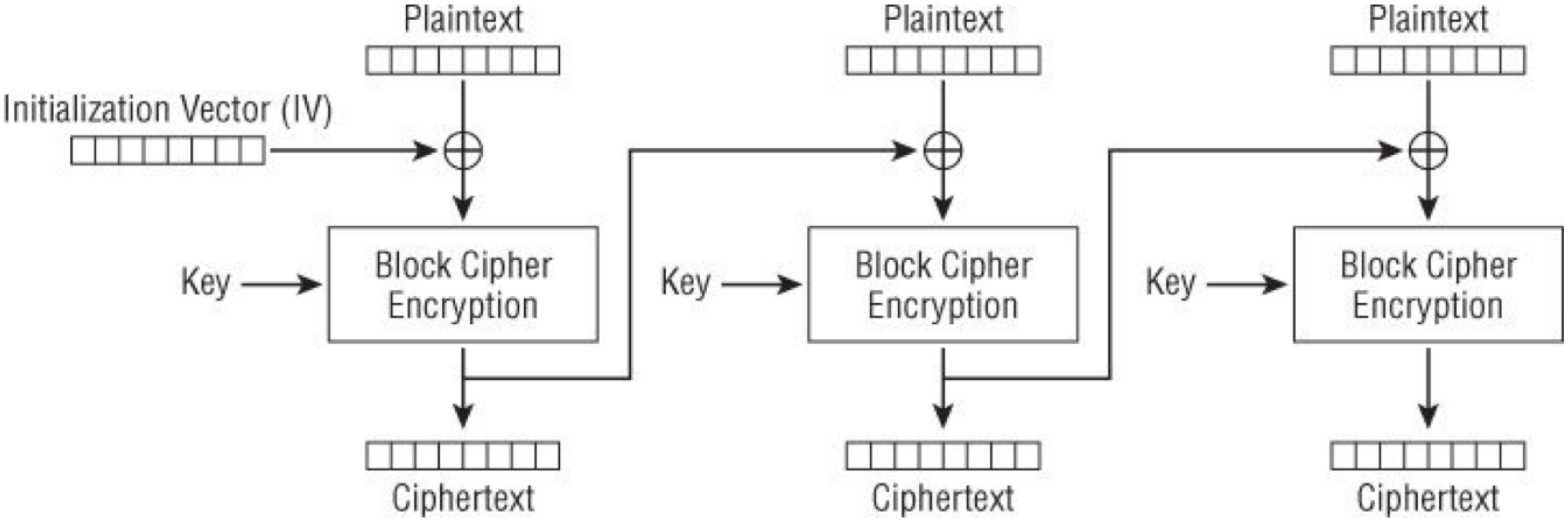
```
rnd=9224     9A5A47BF9B3B6603
856;app=     708F9DEAD67C7F4C
iTradeEU     76FF3D6C6E6A7B9C
R_1;uid=     B8FCA486F9E11922
219;user     A5BC430A73B38C14
name=daf     BD223F003A8309DD
1;time=6     F29A5A6F0DC06C53
34430503     905B5366F5F4684C
61065250     0D2BBBB08BD834BB
0;           ADEBC07FFE87819D
```



```
rnd=9224     9A5A47BF9B3B6603
856;app=     708F9DEAD67C7F4C
iTradeEU     76FF3D6C6E6A7B9C
R_1;uid=     B8FCA486F9E11922
1;time=6     F29A5A6F0DC06C53
219;user     A5BC430A73B38C14
name=daf     BD223F003A8309DD
1;time=6     F29A5A6F0DC06C53
34430503     905B5366F5F4684C
61065250     0D2BBBB08BD834BB
0;           ADEBC07FFE87819D
```

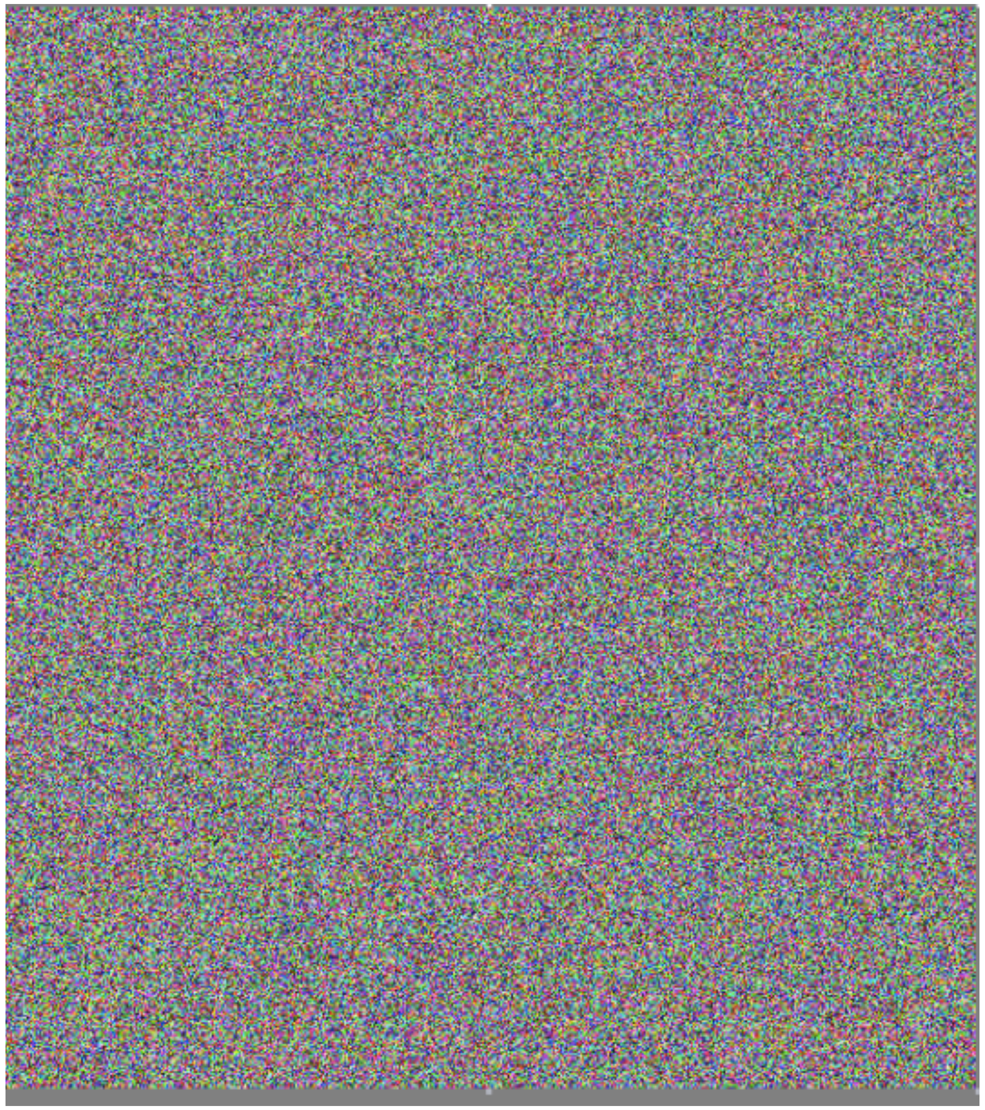
CBC Ciphers

- **Cipher Block Chain mode**
- **XORs each block of plaintext with the preceding block of ciphertext**



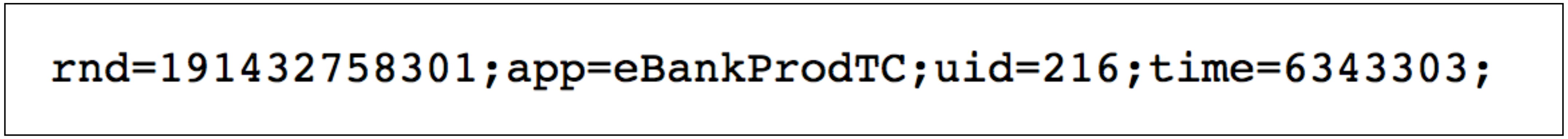
CBC Ciphers

- **Removes all visible patterns from the apple logo**



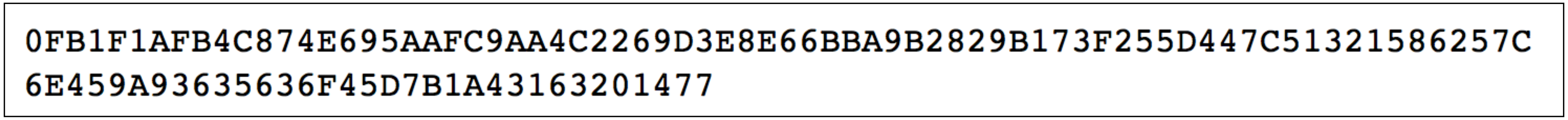
Example: CBC

- **Token contains uid and other fields**



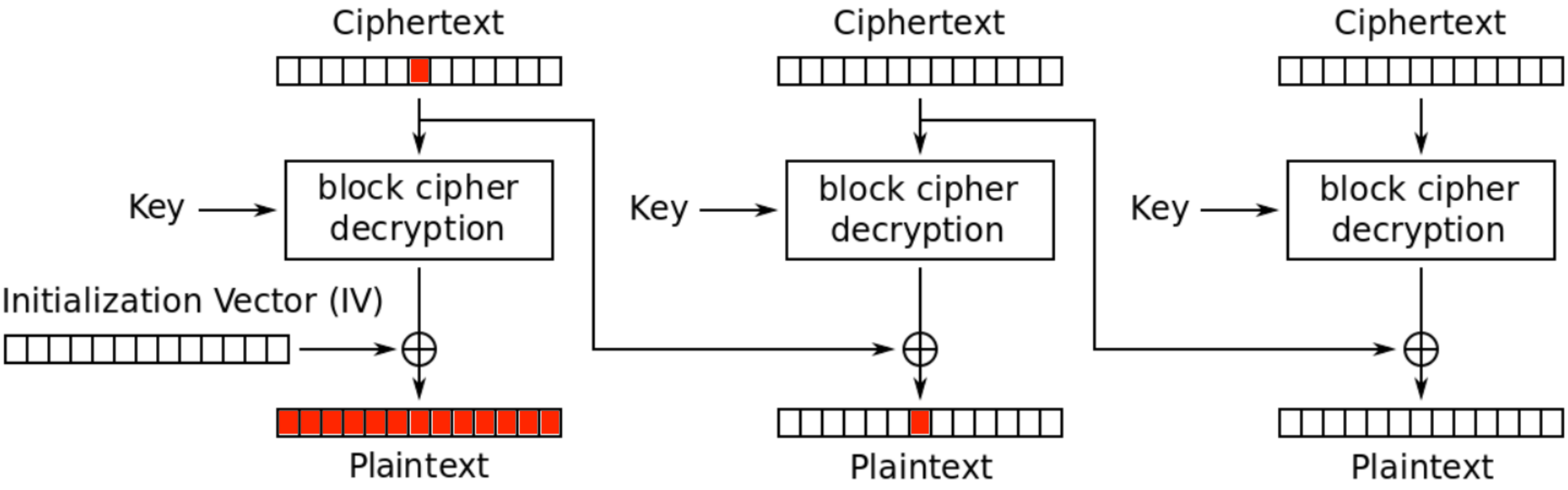
```
rnd=191432758301;app=eBankProdTC;uid=216;time=6343303;
```

- **Encrypted version appears random**



```
0FB1F1AFB4C874E695AAFC9AA4C2269D3E8E66BBA9B2829B173F255D447C51321586257C
6E459A93635636F45D7B1A43163201477
```

Modify a Single Byte of Ciphertext



Cipher Block Chaining (CBC) mode decryption

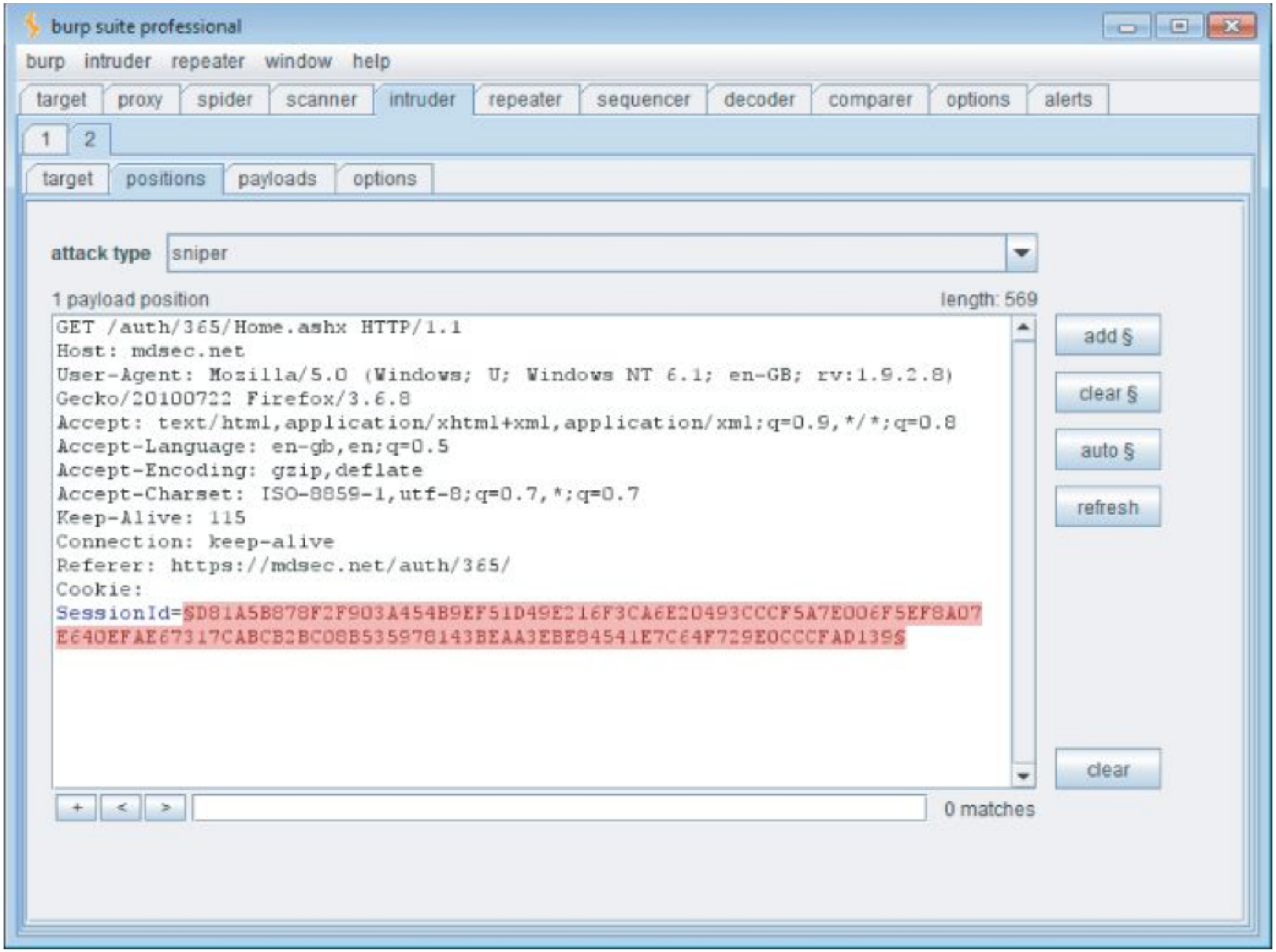
Modify a Single Byte of Ciphertext

- **That block will decrypt to junk**
- **But the next block will remain meaningful, only slightly altered by the XOR**
- **Some of the altered blocks will have valid uid values**

Example Altered Values

```
????????32858301;app=eBankProdTC;uid=216;time=6343303;
????????32758321;app=eBankProdTC;uid=216;time=6343303;
rnd=1914????????;aqp=eBankProdTC;uid=216;time=6343303;
rnd=1914????????;app=eAankProdTC;uid=216;time=6343303;
rnd=191432758301????????nkPqodTC;uid=216;time=6343303;
rnd=191432758301????????nkProdUC;uid=216;time=6343303;
rnd=191432758301;app=eBa????????;uie=216;time=6343303;
rnd=191432758301;app=eBa????????;uid=226;time=6343303;
rnd=191432758301;app=eBankProdTC????????;timd=6343303;
rnd=191432758301;app=eBankProdTC????????;time=6343503;
```

Modify a Session Token



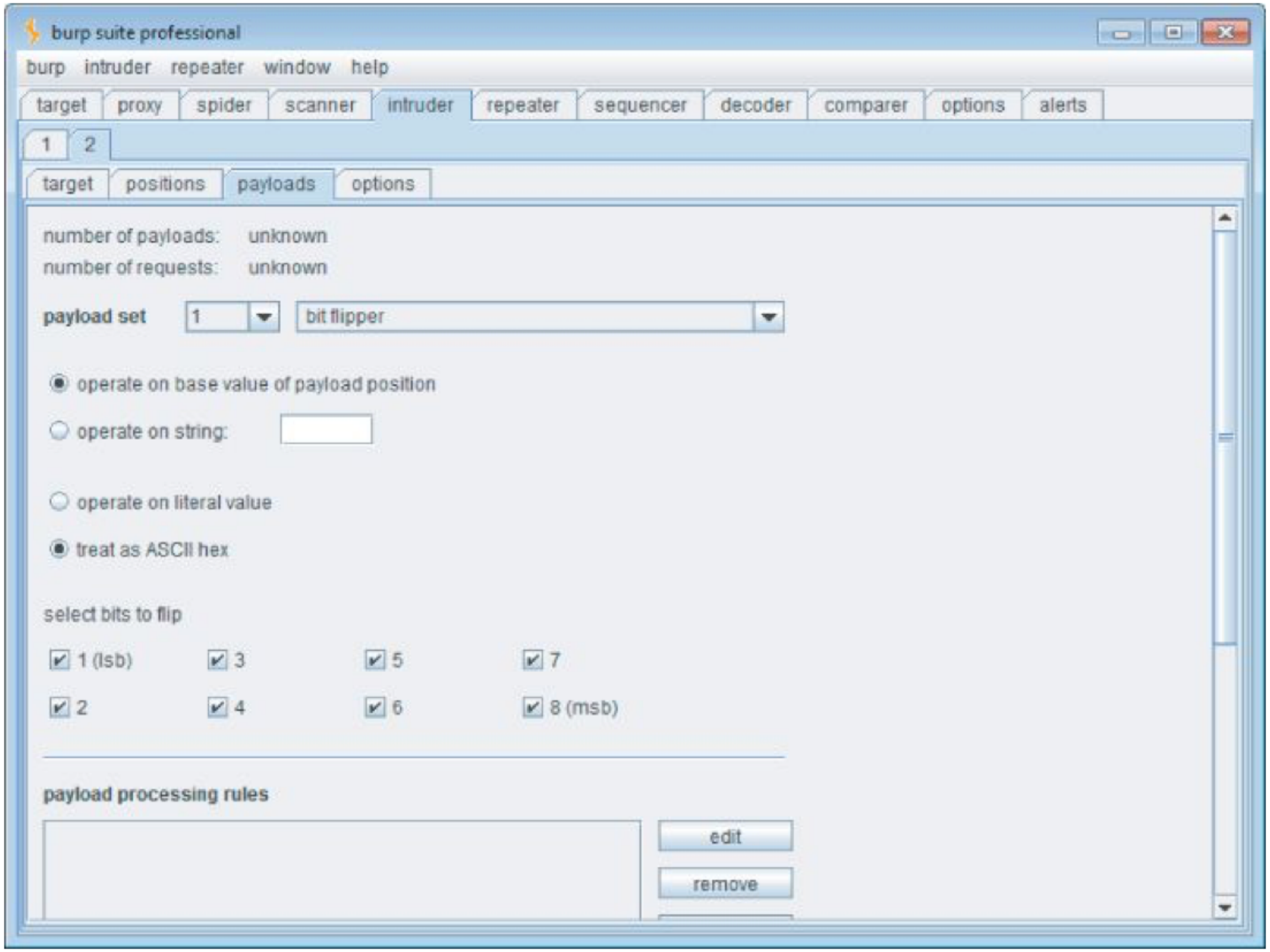
burp suite professional
burp intruder repeater window help
target proxy spider scanner intruder repeater sequencer decoder comparer options alerts
1 2
target positions payloads options

attack type: sniper

1 payload position — length: 569

```
GET /auth/365/Home.ashx HTTP/1.1
Host: mdsec.net
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-GB; rv:1.9.2.8)
Gecko/20100722 Firefox/3.6.8
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-gb,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Referer: https://mdsec.net/auth/365/
Cookie:
SessionId=§D81A5B878F2F903A454B9EF51D49E216F3CA6E20493CCCF5A7E006F5EF8A07
E640EFAE67317CABCB2BC08B535978143BEAA3EBE84541E7C64F729E0CCCFAD139§
```

Flip Each Bit


burp suite professional
burp intruder repeater window help
target proxy spider scanner intruder repeater sequencer decoder comparer options alerts
1 2
target positions payloads options
edit
remove

number of payloads: unknown

number of requests: unknown

payload set 1 bit flipper

● operate on base value of payload position

○ operate on string:

○ operate on literal value

● treat as ASCII hex

select bits to flip

☑ 1 (lsb) ☑ 3 ☑ 5 ☑ 7

☑ 2 ☑ 4 ☑ 6 ☑ 8 (msb)

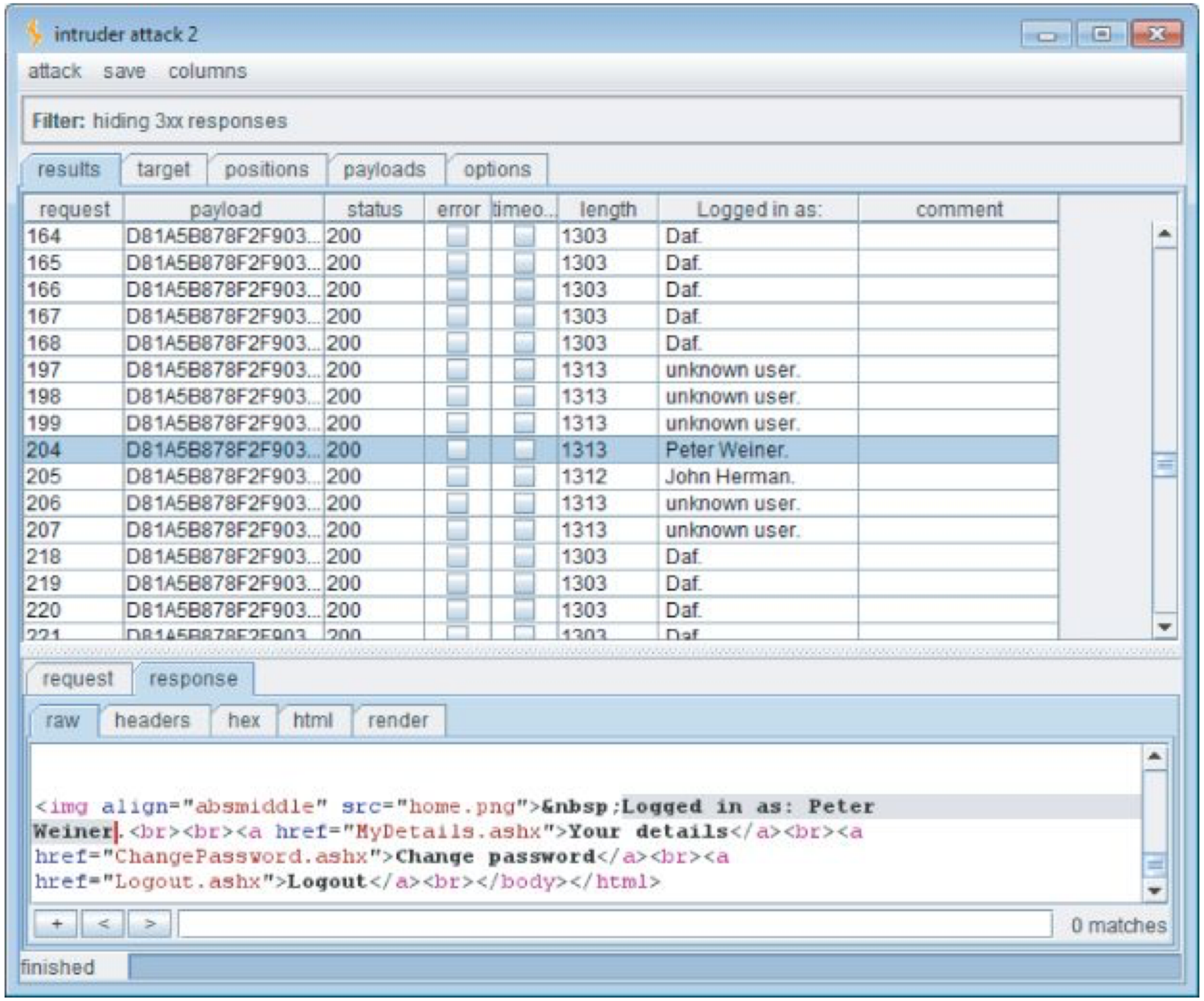
payload processing rules

Fast Method

- **8 requests per byte**
- **Won't take long to try all single bit flips**
- **Will confirm whether application is vulnerable**

Results

- **Some flips change user id to "invalid"**
- **Others reach real accounts for other users!**



Filter: hiding 3xx responses

request	payload	status	error	timeo...	length	Logged in as:	comment
164	D81A5B878F2F903...	200	□	□	1303	Daf.	
165	D81A5B878F2F903...	200	□	□	1303	Daf.	
166	D81A5B878F2F903...	200	□	□	1303	Daf.	
167	D81A5B878F2F903...	200	□	□	1303	Daf.	
168	D81A5B878F2F903...	200	□	□	1303	Daf.	
197	D81A5B878F2F903...	200	□	□	1313	unknown user.	
198	D81A5B878F2F903...	200	□	□	1313	unknown user.	
199	D81A5B878F2F903...	200	□	□	1313	unknown user.	
204	D81A5B878F2F903...	200	□	□	1313	Peter Weiner.	
205	D81A5B878F2F903...	200	□	□	1312	John Herman.	
206	D81A5B878F2F903...	200	□	□	1313	unknown user.	
207	D81A5B878F2F903...	200	□	□	1313	unknown user.	
218	D81A5B878F2F903...	200	□	□	1303	Daf.	
219	D81A5B878F2F903...	200	□	□	1303	Daf.	
220	D81A5B878F2F903...	200	□	□	1303	Daf.	
221	D81A5B878F2F903	200	□	□	1303	Daf	

```
<img align="absmiddle" src="home.png">&nbsp;Logged in as: Peter
Weiner.<br><br><a href="MyDetails.ashx">Your details</a><br><a
href="ChangePassword.ashx">Change password</a><br><a
href="Logout.ashx">Logout</a><br></body></html>
```

Information Leakage

- **Application may re-use the encryption code elsewhere**
- **It may allow user to submit arbitrary input and see the ciphertext**
- **Such as to create a download link for an uploaded file**
- **Submit desired plaintext as a filename, such as**
  - **uid=1**

Kahoot!

Ch 7b

Weaknesses in Session Token Handling

Common Myths

- **"SSL protects tokens in transmission"**
  - **Some attacks still work, such as XSS**
- **"Our platform uses mature, sound cryptography so the token is not vulnerable"**
  - **But cookie may be stolen in transit**

Disclosure on the Network

- **Tokens transmitted without encryption**
- **Can be sniffed from many locations**
  - **User's local network**
  - **Within ISP**
  - **Within IT department of server hosting the application**

Credential Theft v. Token Theft

- **Stealing a user's password may not work**
  - **Two-factor authentication, requiring a PIN in addition to the password**
  - **Login performed over HTTPS**
- **Stealing a session token and hijacking an authenticated session may still work**
  - **And the user won't be notified of a extra successful login**

Not Always HTTPS

- **An app may use HTTPS during login**
- **But use HTTP after login to see authorized content**
- **Gmail did this until 2012**
- **Eavesdropper cannot steal password, but can steal session token**

Upgradeable Token

- **App may use HTTP for preauthenticated pages**
  - **Such as the site's front page**
- **And use HTTPS for login and all subsequent pages**
  - **But continue to use the same token; upgrade to an authenticated session**
- **Attacker can steal the token before login**

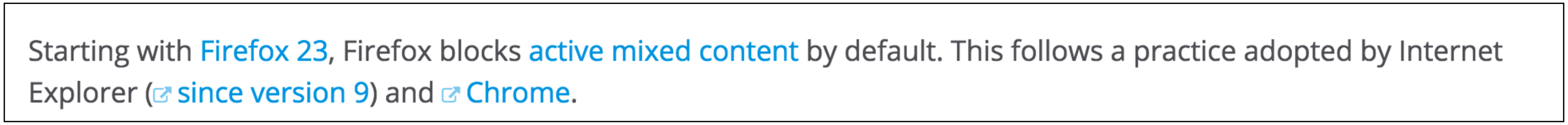
Back Button

- **App uses HTTPS for login and all subsequent pages**
- **With a new token**
- **But user navigates back to an HTTP page with the Back button**
- **Exposing the token**

sslstrip

- "Log In" link goes to an HTTPS page
- Attacker in the middle alters the page to use HTTP instead
- And forwards requests to the server via HTTPS
- User won't see any obvious difference in the page

Mixed Content



Starting with Firefox 23, Firefox blocks active mixed content by default. This follows a practice adopted by Internet Explorer (since version 9) and Chrome.

- HTTPS page with some HTTP content
  - Such as images, style sheets, etc.
- This can send the session token over HTTP
- Browsers now block some mixed-content by default

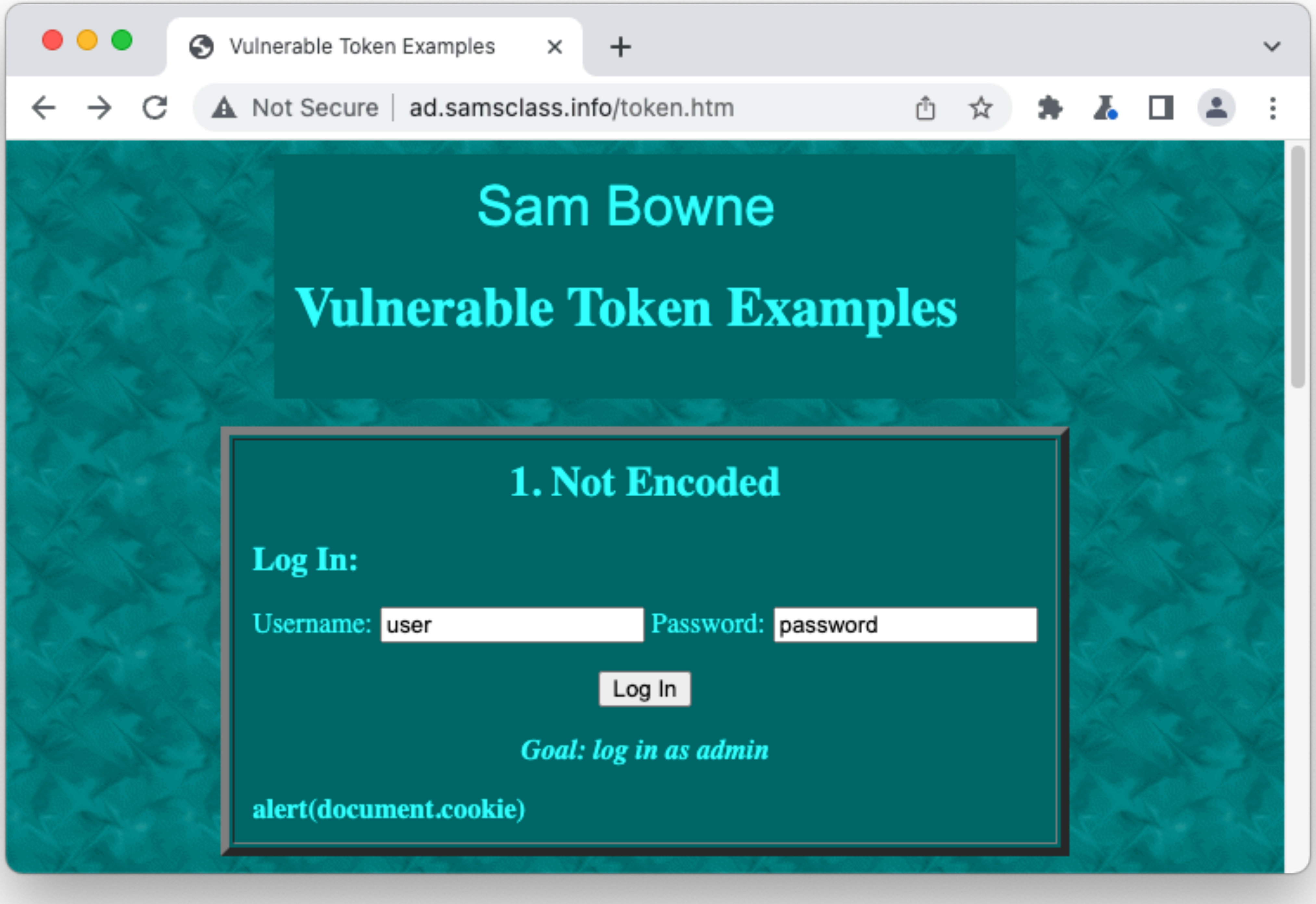
Social Engineering

- **App uses HTTPS for every page**
- **Send user an HTTP link via email or IM, or added to some page the user views**
  - **To http://target.com or http://target.com:443**
- **Clicking that link may send a session token over HTTP**

Hack Steps

- **Walk through the app, from start page, through login, and all its functionality**
- **Record every URL and note every session token you receive**
- **Note transitions between HTTP and HTTPS**

Demo



Sam Bowne

Vulnerable Token Examples

1. Not Encoded

Log In:

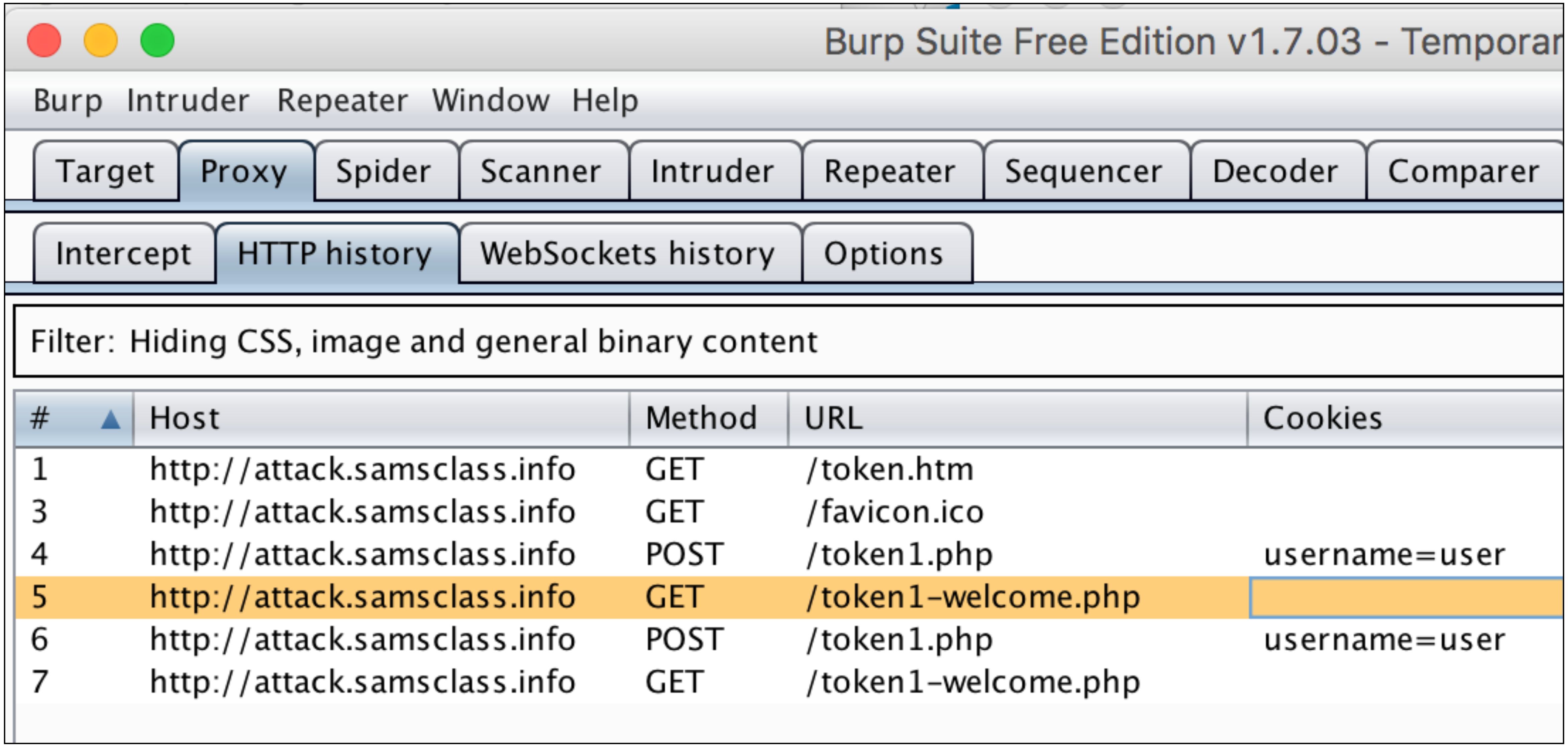
Username: user Password: password

Log In

Goal: log in as admin

alert(document.cookie)

- **After login, send welcome page to repeater**
- **Username is in the cookie**

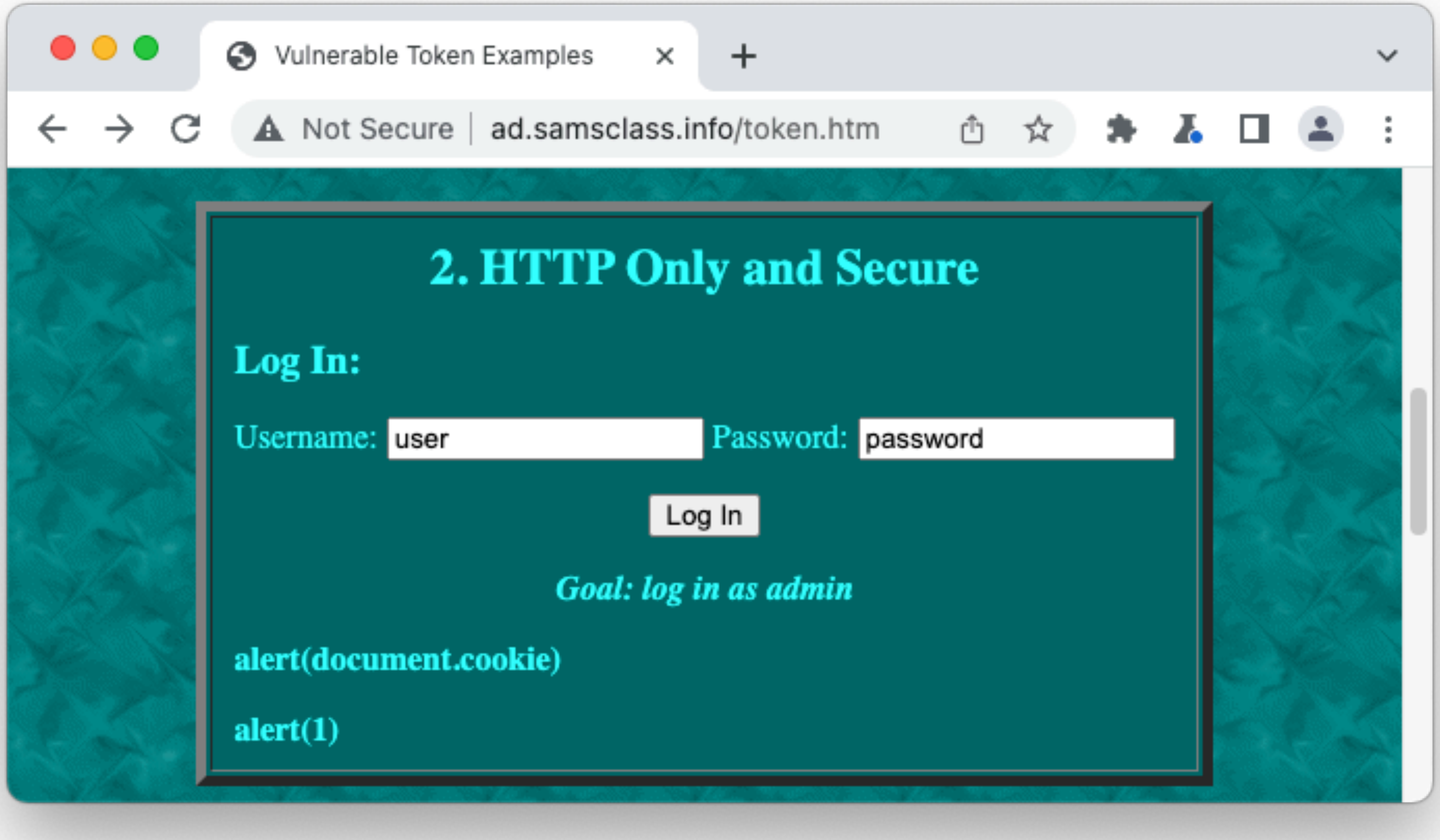


Filter: Hiding CSS, image and general binary content

# ▲	Host	Method	URL	Cookies
1	http://attack.samsclass.info	GET	/token.htm	
3	http://attack.samsclass.info	GET	/favicon.ico	
4	http://attack.samsclass.info	POST	/token1.php	username=user
5	http://attack.samsclass.info	GET	/token1-welcome.php	
6	http://attack.samsclass.info	POST	/token1.php	username=user
7	http://attack.samsclass.info	GET	/token1-welcome.php	

Secure Cookies

- **If secure flag is set on a cookie, browser will only send it via HTTPS**
- **If the connection is only HTTP, that cookie won't be sent**



2. HTTP Only and Secure

Log In:

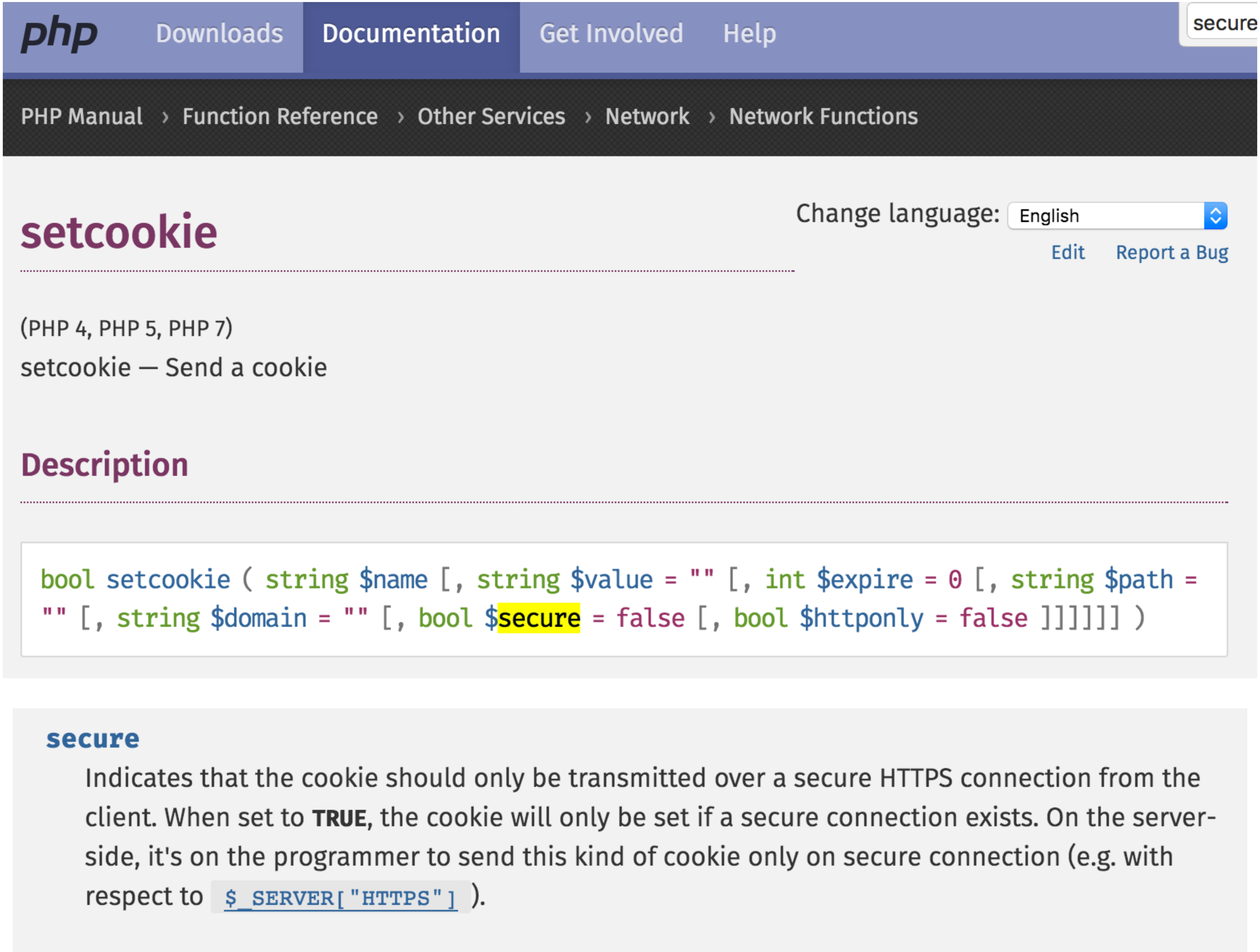
Username: user Password: password

Goal: log in as admin

alert(document.cookie)

alert(1)

```
<?php
$p = strip_tags($_POST['password']);
$u = strip_tags($_POST['username']);
$date_of_expiry = time() + 3000 ;

if ( ($u == "user") && ($p == "password") ) {
    setcookie( "username", $u, $date_of_expiry, '', '', true, true );
    header ("Location: token2-welcome.php");
}
```



setcookie

(PHP 4, PHP 5, PHP 7)

setcookie — Send a cookie

Description

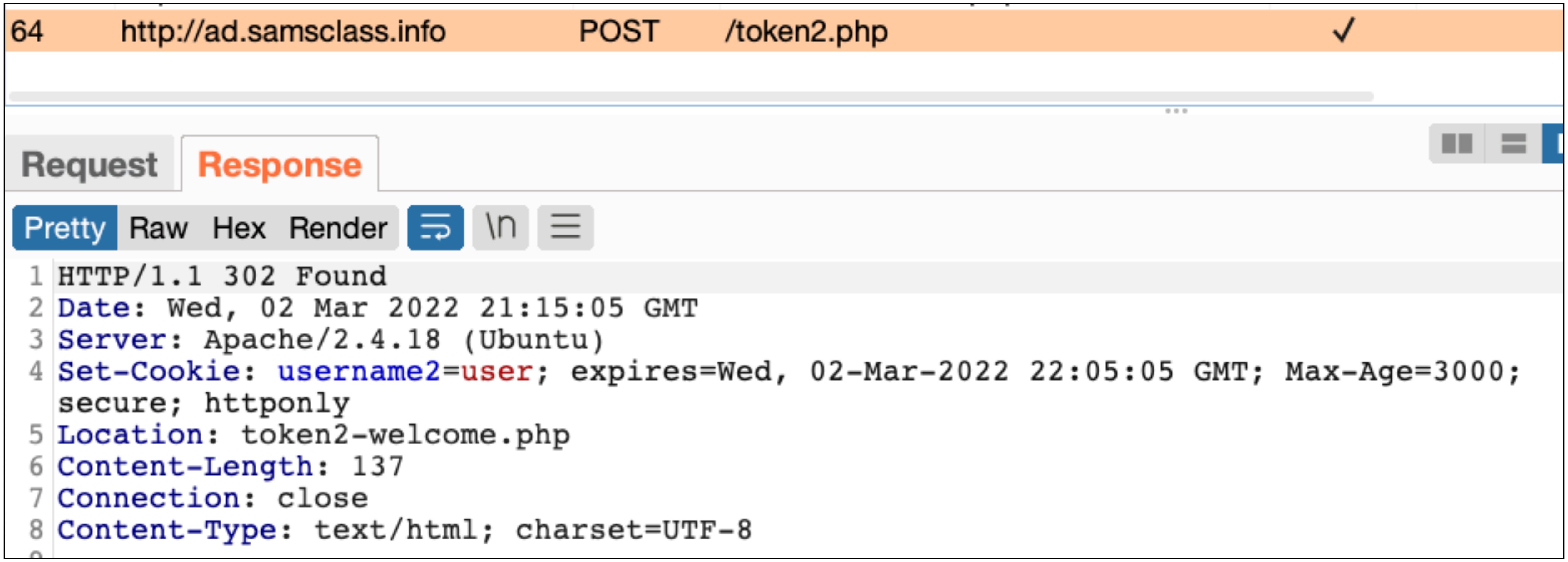
```
bool setcookie ( string $name [, string $value = "" [, int $expire = 0 [, string $path =
"" [, string $domain = "" [, bool $secure = false [, bool $httponly = false ]]]]]] )
```

secure

Indicates that the cookie should only be transmitted over a secure HTTPS connection from the client. When set to **TRUE**, the cookie will only be set if a secure connection exists. On the server-side, it's on the programmer to send this kind of cookie only on secure connection (e.g. with respect to `$_SERVER["HTTPS"]`).

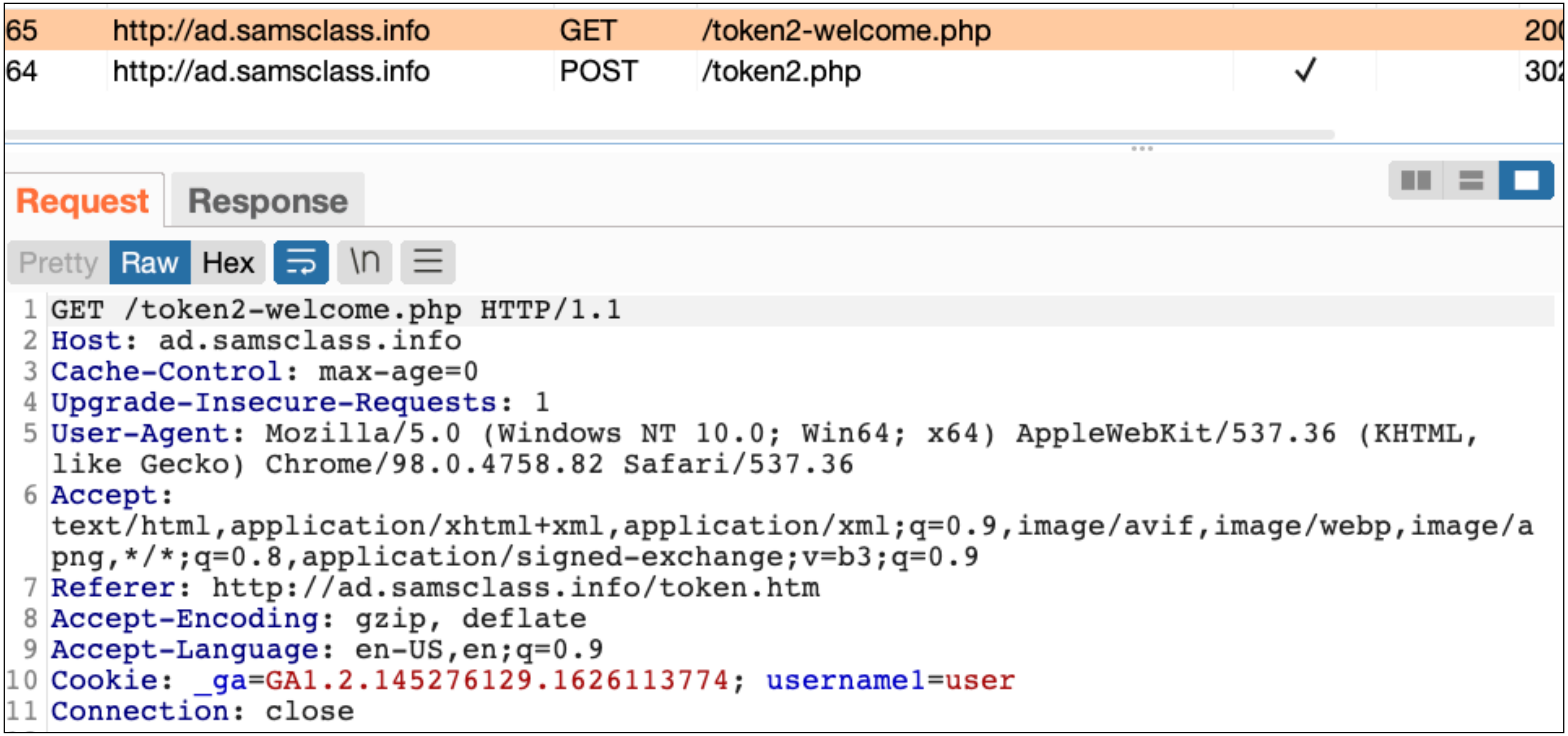
Setting Secure Cookie

- **Valid username and password**
- **Server sets username2, secure, httponly**



```
HTTP/1.1 302 Found
Date: Wed, 02 Mar 2022 21:15:05 GMT
Server: Apache/2.4.18 (Ubuntu)
Set-Cookie: username2=user; expires=Wed, 02-Mar-2022 22:05:05 GMT; Max-Age=3000;
secure; httponly
Location: token2-welcome.php
Content-Length: 137
Connection: close
Content-Type: text/html; charset=UTF-8
```

Transmitting Secure Cookie

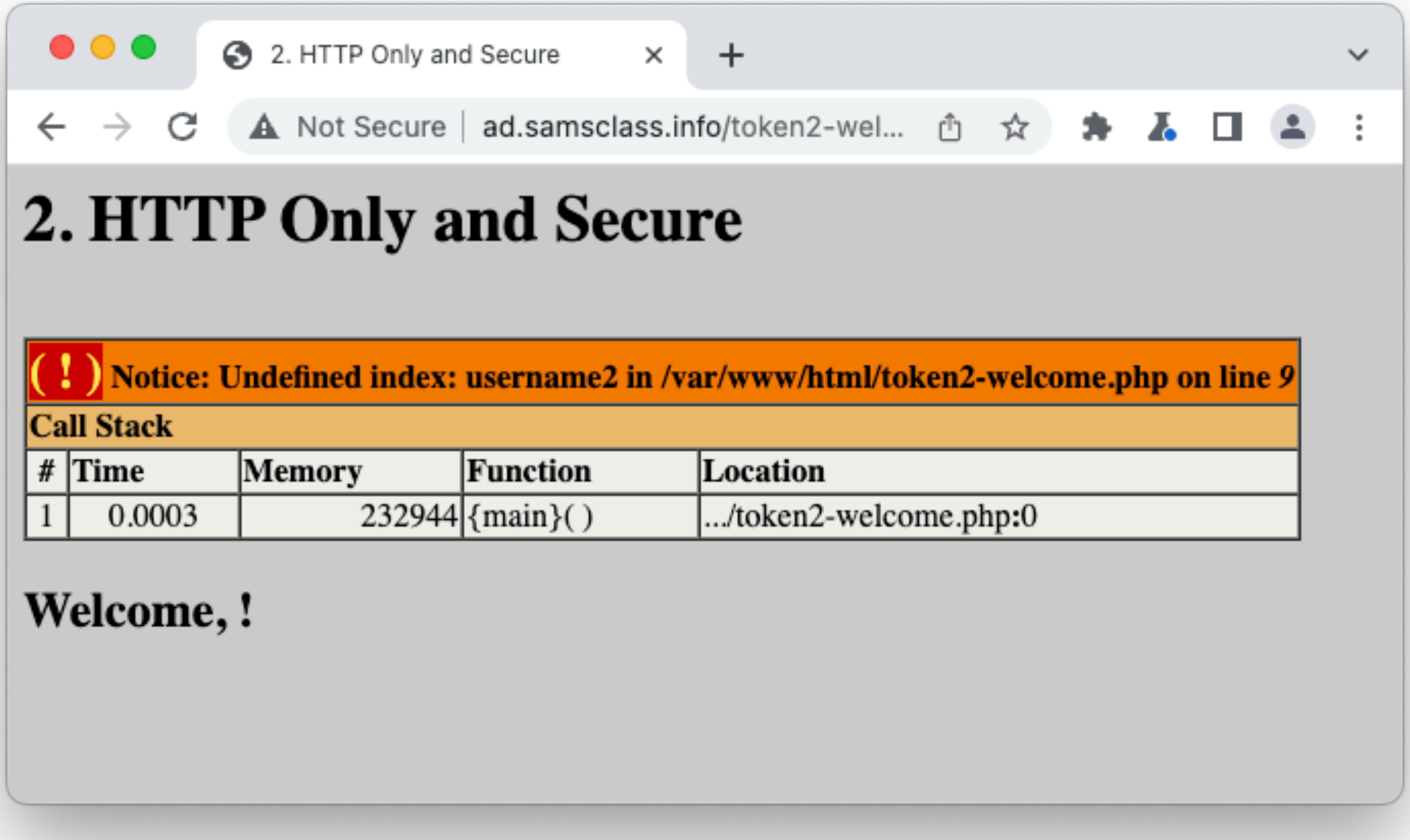


65	http://ad.samsclass.info	GET	/token2-welcome.php		20
64	http://ad.samsclass.info	POST	/token2.php	✓	30

```
GET /token2-welcome.php HTTP/1.1
Host: ad.samsclass.info
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/98.0.4758.82 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://ad.samsclass.info/token.htm
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: _ga=GA1.2.145276129.1626113774; username1=user
Connection: close
```

- **http session, so username2 is NOT sent**

Welcome Page Can't See Username



2. HTTP Only and Secure

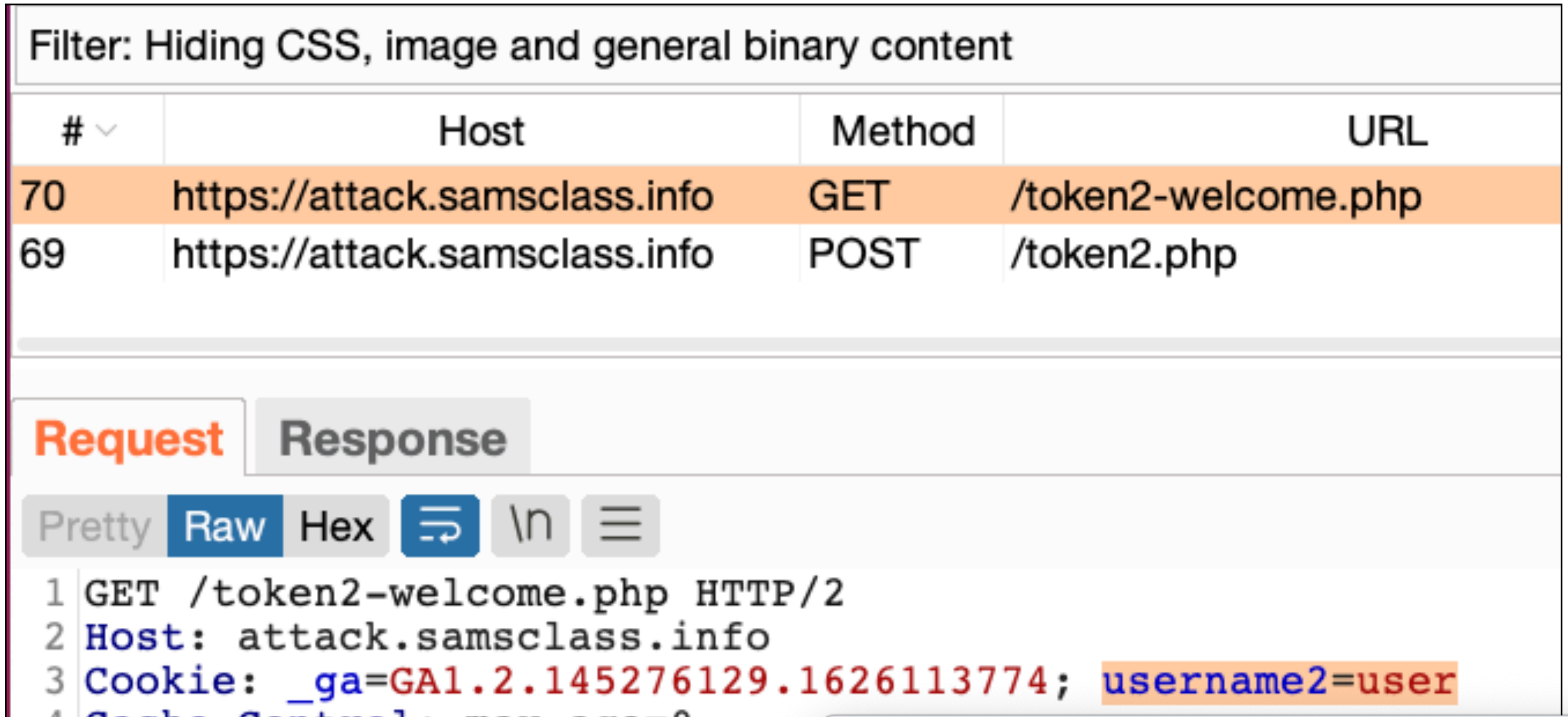
(!) Notice: Undefined index: username2 in /var/www/html/token2-welcome.php on line *9*

Call Stack				
#	**Time**	**Memory**	**Function**	**Location**
1	0.0003	232944	{main}( )	.../token2-welcome.php:0

Welcome, !

Try a Secure Connection

- **Go to https://attack.samsclass.info/token.htm**

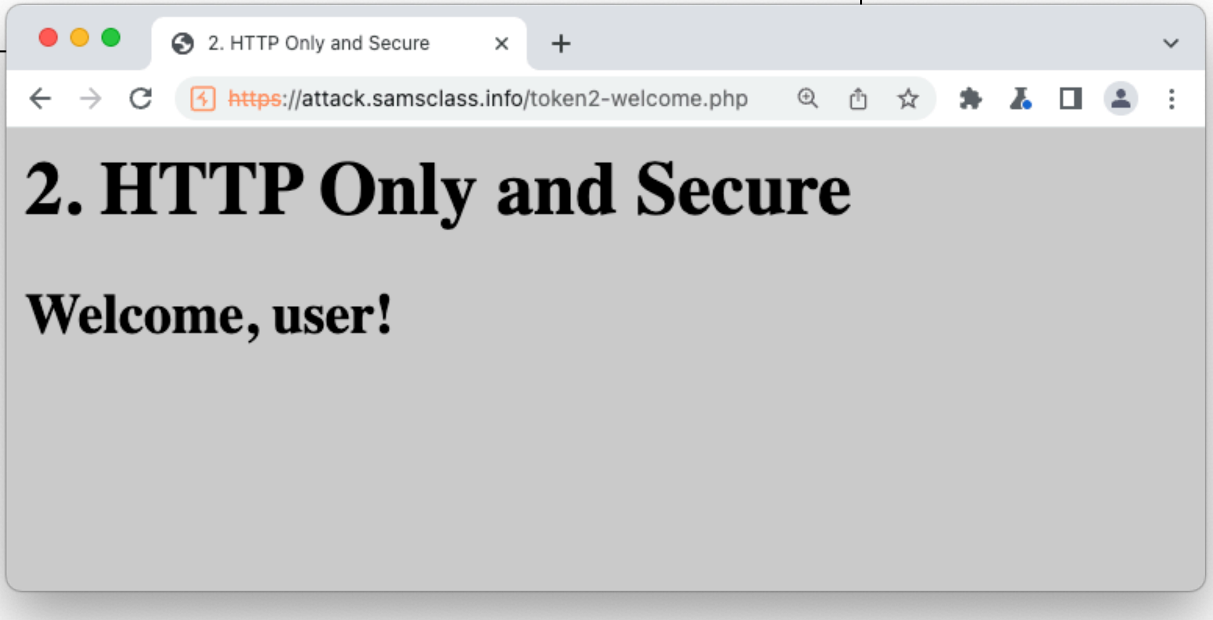


Filter: Hiding CSS, image and general binary content

#	Host	Method	URL
70	https://attack.samsclass.info	GET	/token2-welcome.php
69	https://attack.samsclass.info	POST	/token2.php

```
GET /token2-welcome.php HTTP/2
Host: attack.samsclass.info
Cookie: _ga=GA1.2.145276129.1626113774; username2=user
Cache-Control: max-age=0
```

- **Cookie sent**
- **Welcome page knows my name**

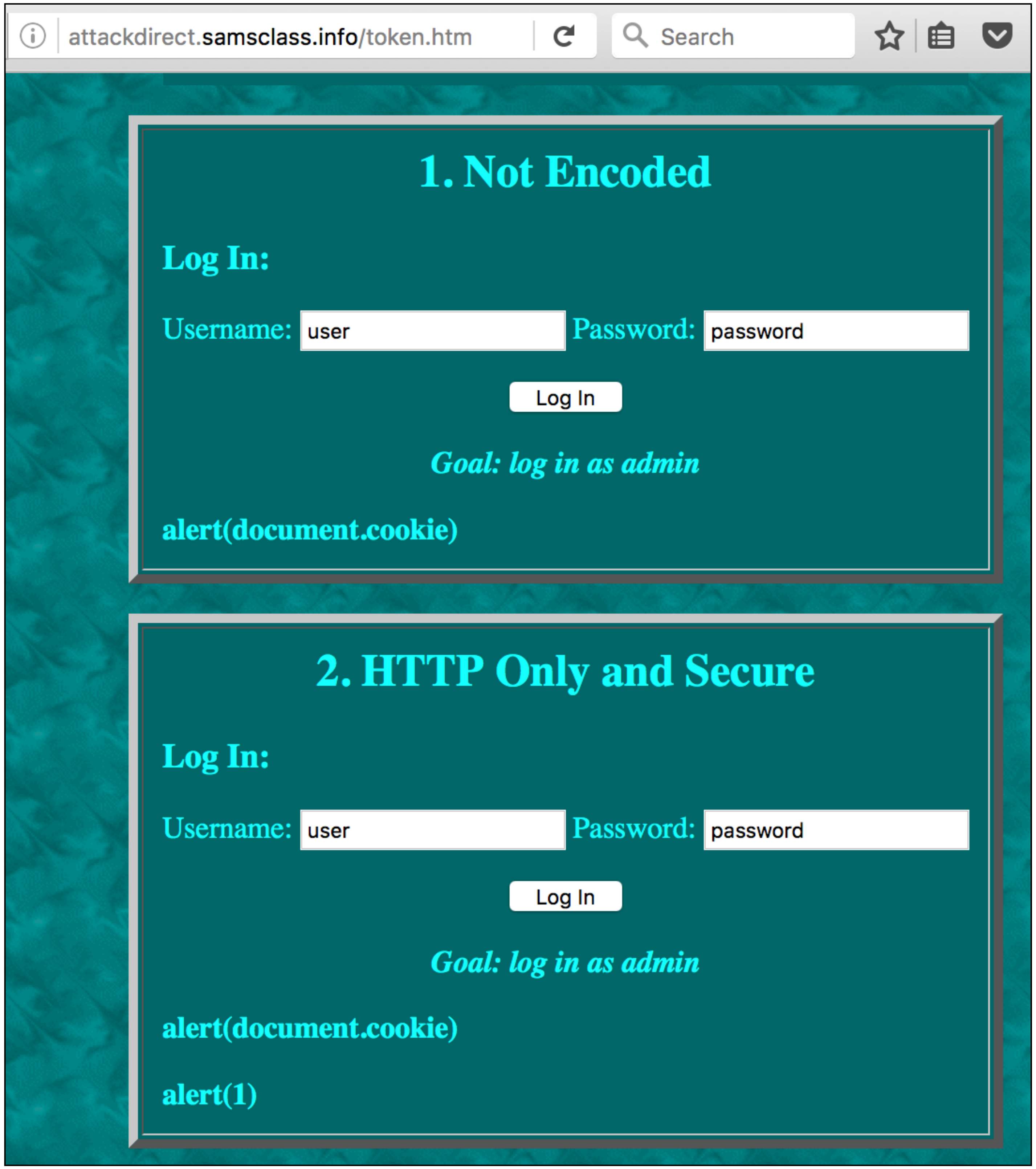


2. HTTP Only and Secure

Welcome, user!

httponly

- **httponly cookie cannot be used by JavaScript**



attackdirect.samsclass.info/token.htm

1. Not Encoded

Log In:

Username: user Password: password

Goal: log in as admin

alert(document.cookie)

2. HTTP Only and Secure

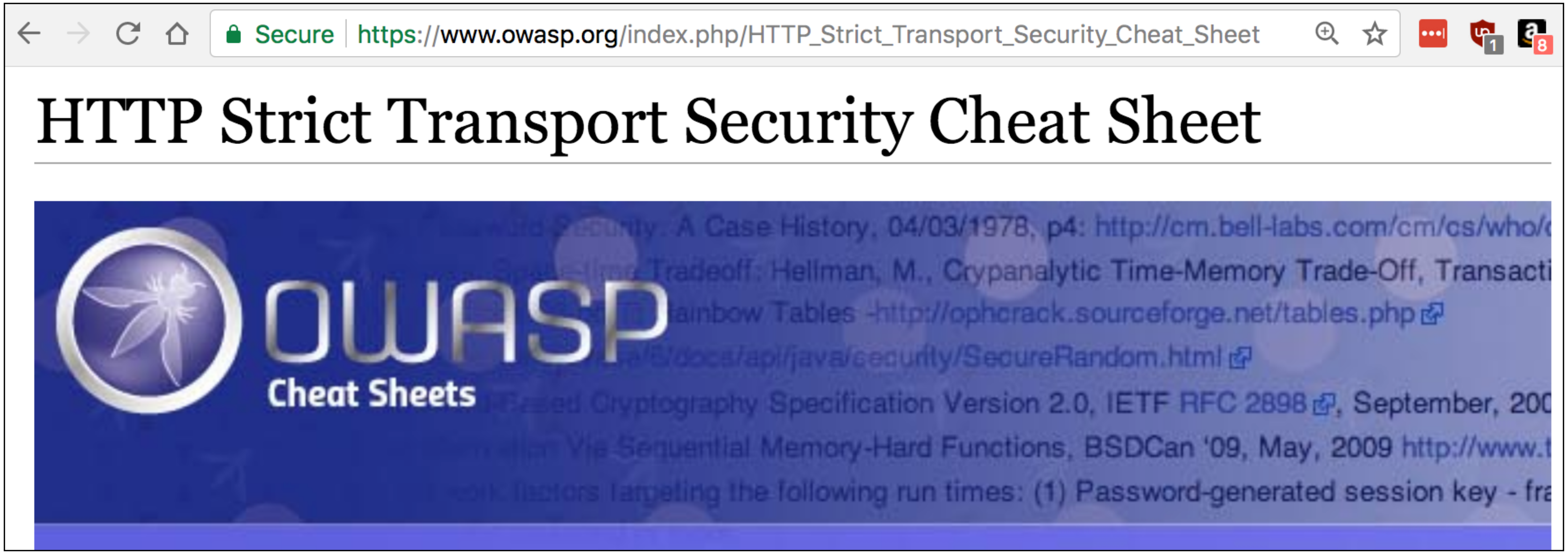
Log In:

Username: user Password: password

Goal: log in as admin

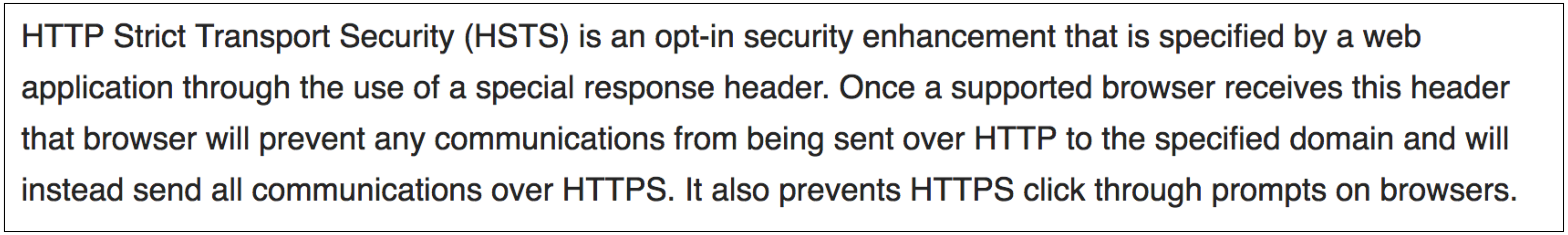
alert(document.cookie)

alert(1)



Secure https://www.owasp.org/index.php/HTTP_Strict_Transport_Security_Cheat_Sheet

HTTP Strict Transport Security Cheat Sheet



HTTP Strict Transport Security (HSTS) is an opt-in security enhancement that is specified by a web application through the use of a special response header. Once a supported browser receives this header that browser will prevent any communications from being sent over HTTP to the specified domain and will instead send all communications over HTTPS. It also prevents HTTPS click through prompts on browsers.

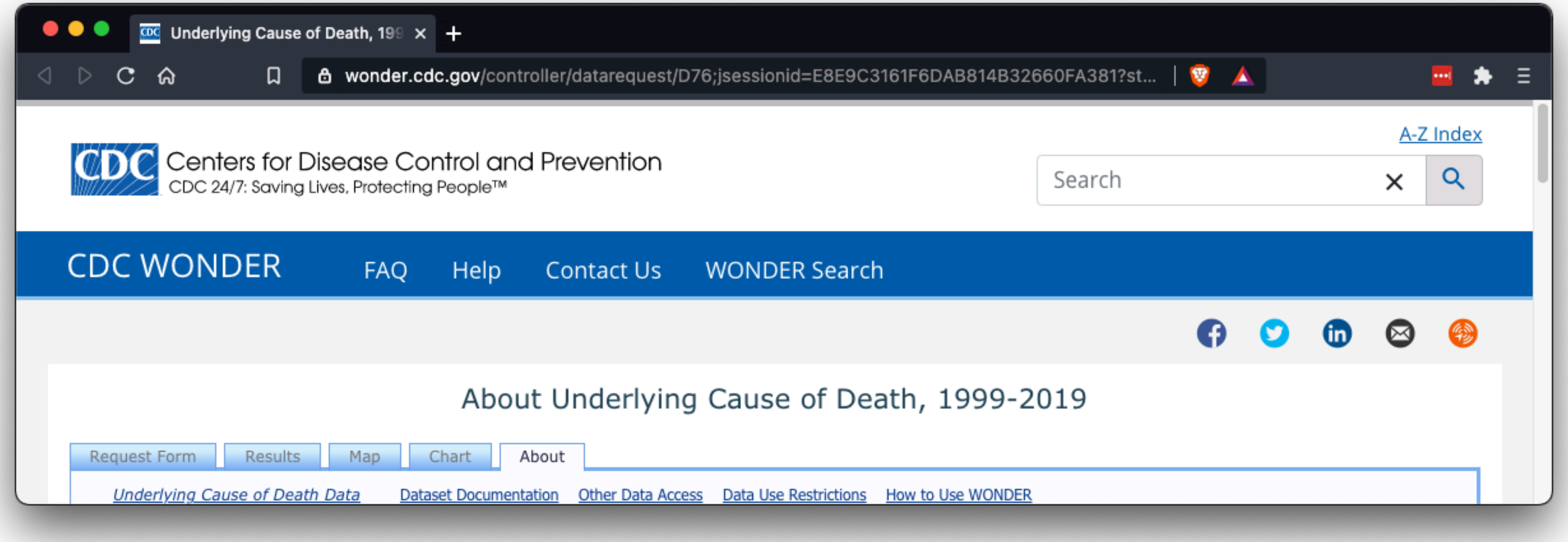
- **23% of websites use HSTS (link Ch 7h)**

Disclosure of Tokens in Logs

- **Less common as unencrypted network traffic but more serious**
- **Logs often visible to more attackers**

Google:

inurl:jsessionid

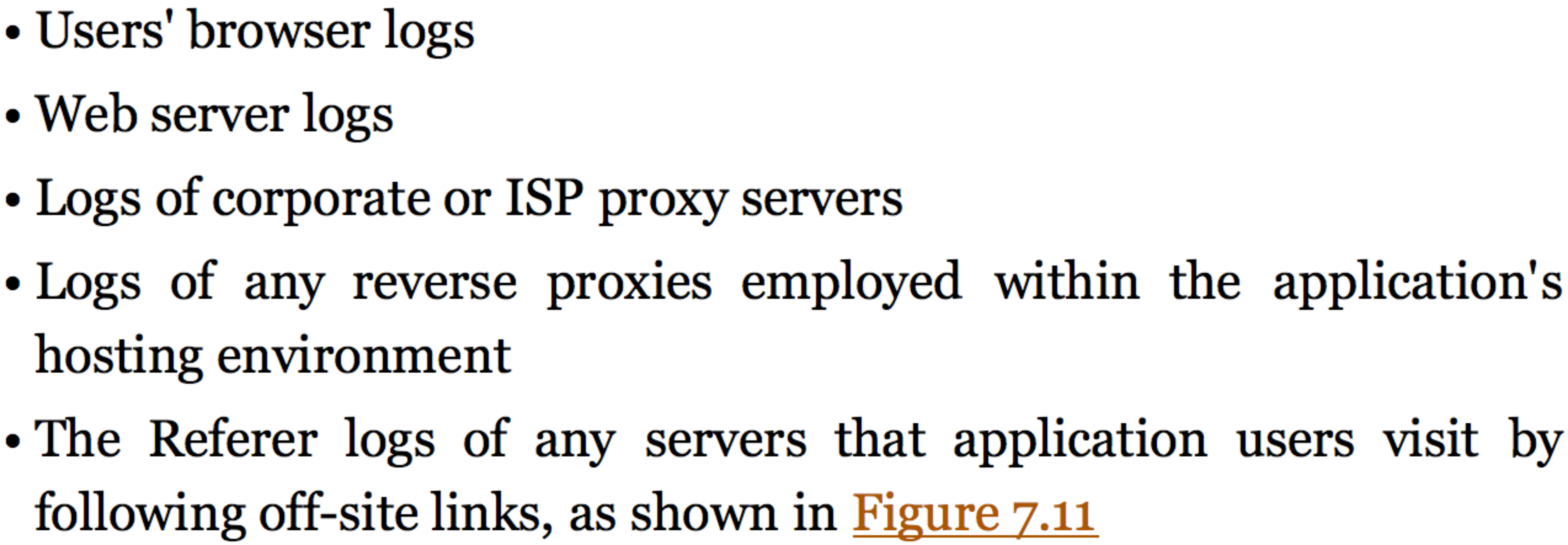

Underlying Cause of Death, 199

wonder.cdc.gov/controller/datarequest/D76;jsessionid=E8E9C3161F6DAB814B32660FA381?st...

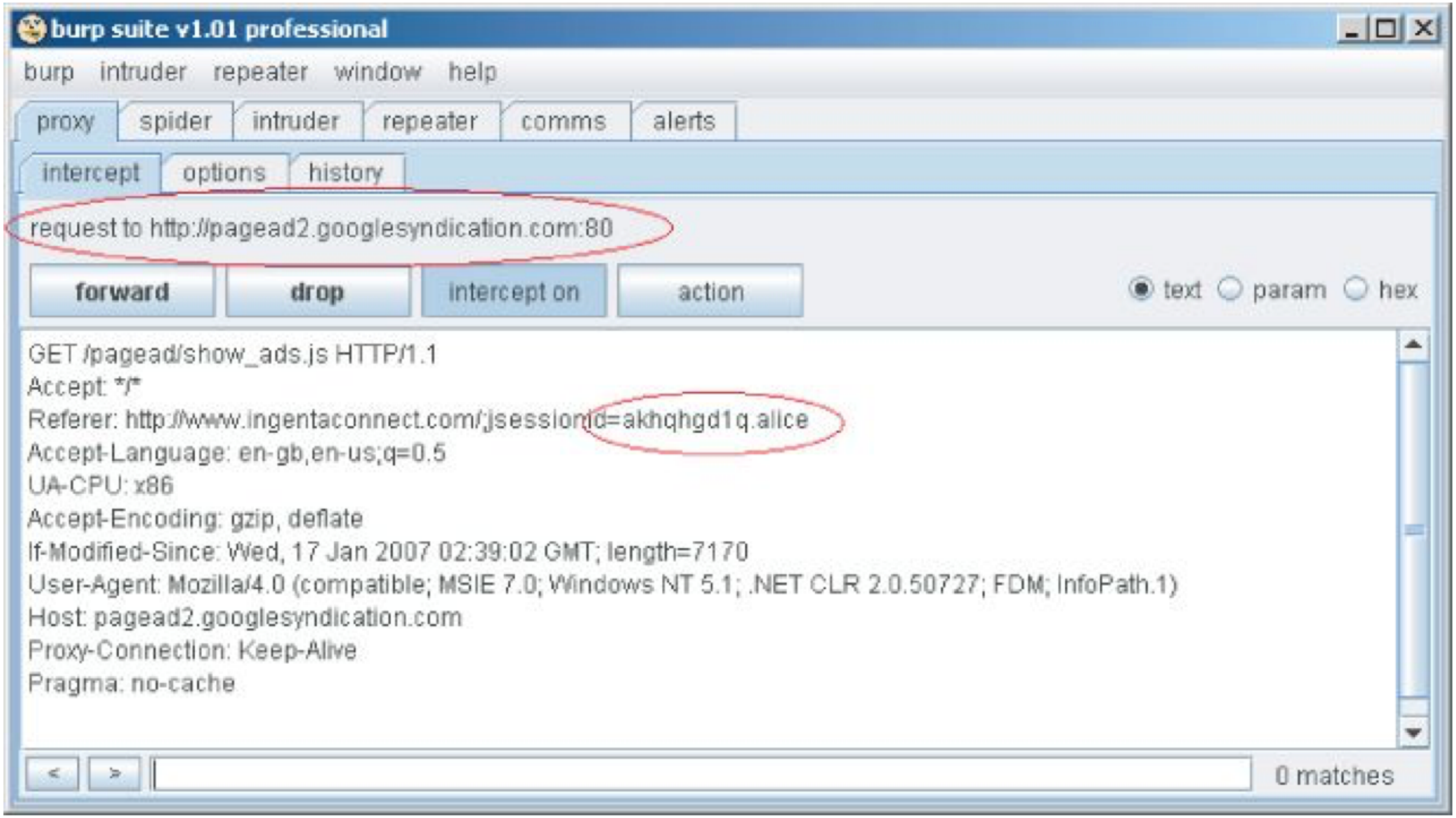
Centers for Disease Control and Prevention
CDC 24/7: Saving Lives, Protecting People™

About Underlying Cause of Death, 1999-2019

Where SessionIDs in URL Will Appear



- Users' browser logs
- Web server logs
- Logs of corporate or ISP proxy servers
- Logs of any reverse proxies employed within the application's hosting environment
- The Referer logs of any servers that application users visit by following off-site links, as shown in Figure 7.11



request to http://pagead2.googlesyndication.com:80

● text ○ param ○ hex

```
GET /pagead/show_ads.js HTTP/1.1
Accept: */*
Referer: http://www.ingentaconnect.com/;jsessionid=akhqhgd1q.alice
Accept-Language: en-gb,en-us;q=0.5
UA-CPU: x86
Accept-Encoding: gzip, deflate
If-Modified-Since: Wed, 17 Jan 2007 02:39:02 GMT; length=7170
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 2.0.50727; FDM; InfoPath.1)
Host: pagead2.googlesyndication.com
Proxy-Connection: Keep-Alive
Pragma: no-cache
```

Referer Attack

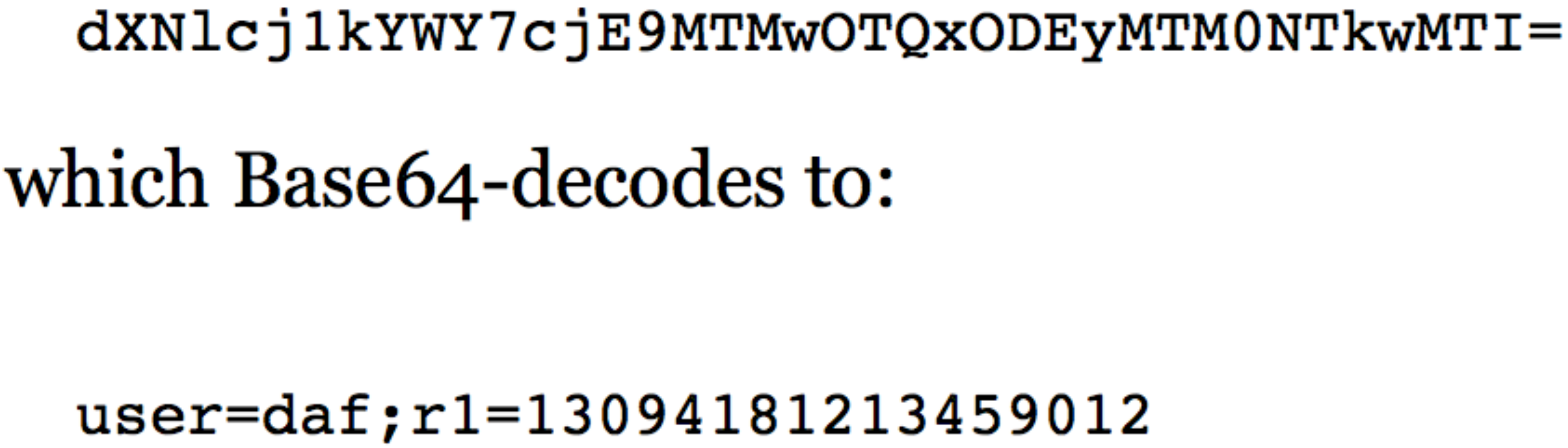
- **A web mail application transmits session tokens in the URL**
- **Send email to targets containing a URL on the attacker's Web server**
- **The Referer headers from people who click will appear in the server logs**

Vulnerable Mapping of Tokens to Sessions

- **Allowing users to have two sessions open at the same time**
- **Using static tokens (same token is sent to the user each time they log in)**
  - **Misunderstanding of what a session is**

Flawed Logic

- **Token value**



```
dXNlcj1kYWY7cjE9MTMwOTQxODEyMTM0NTkwMTI=
```

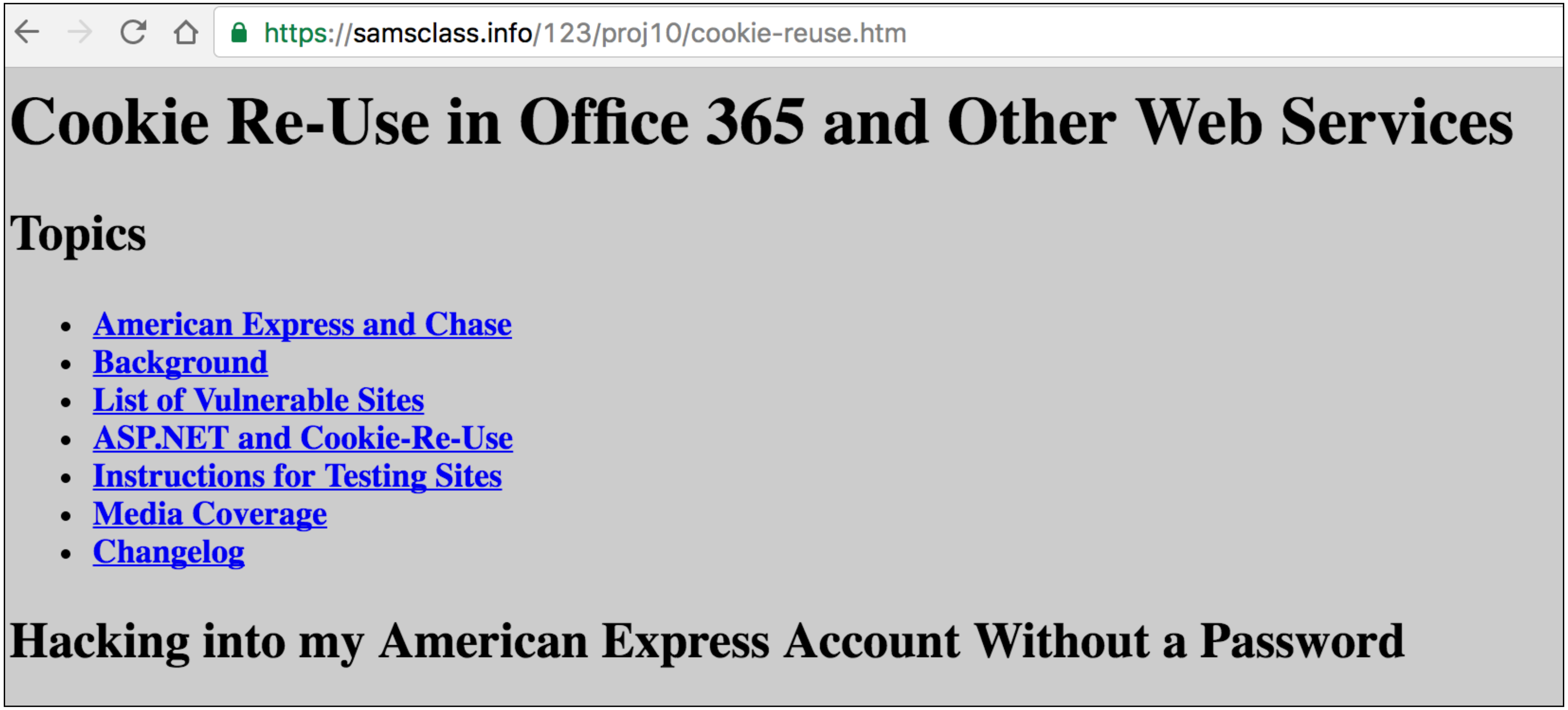
which Base64-decodes to:

```
user=daf;r1=13094181213459012
```

- **But app accepts the same "r1" with a different "user"**

Vulnerable Session Termination

- **A session may remain valid for days after the last request is received**
- **Ineffective logout functionality**
- **Logout merely deletes cookie but does not invalidate it**
  - **Logout merely runs a client-side script, server doesn't know a logout has occurred**



https://samsclass.info/123/proj10/cookie-reuse.htm

Cookie Re-Use in Office 365 and Other Web Services

Topics

- American Express and Chase
- Background
- List of Vulnerable Sites
- ASP.NET and Cookie-Re-Use
- Instructions for Testing Sites
- Media Coverage
- Changelog

Hacking into my American Express Account Without a Password

Token Hijacking

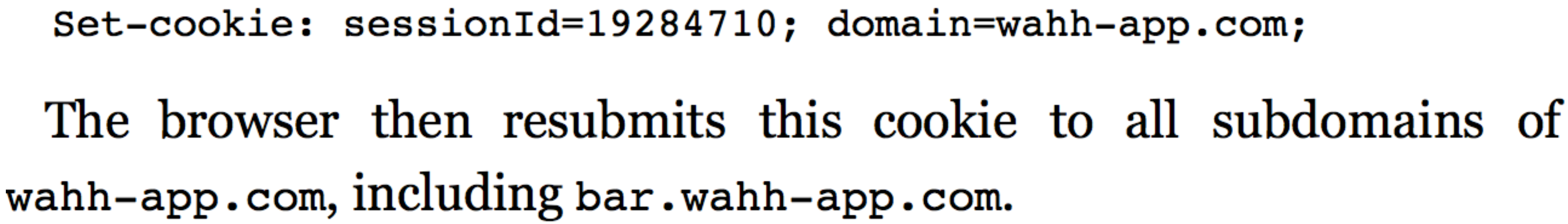
- **Cookie theft**
- **Session fixation: attacker feeds a token to the user, then user logs in, then attacker hijacks the session**
- **Cross-Site Request Forgery (CSRF)**
  - **Tricks user into submitting a request containing a cookie that goes to an attacker's server**

Liberal Cookie Scope

- **When a cookie is set, the server can set the *domain* and *url* (path) the cookie is used for**
- **By default, all subdomains are included**
  - **Cookie set by games.samsclass.info**
  - **Will be sent to foo.games.samsclass.info**
  - **But NOT to samsclass.info**

Specifying the Domain

- **App at foo.wahh-app.com sets this cookie:**



```
Set-cookie: sessionId=19284710; domain=wahh-app.com;
```

The browser then resubmits this cookie to all subdomains of `wahh-app.com`, including `bar.wahh-app.com`.

- **A domain can only set a cookie for the same domain or a parent domain**
  - **And not a top-level domain like .com**

Example

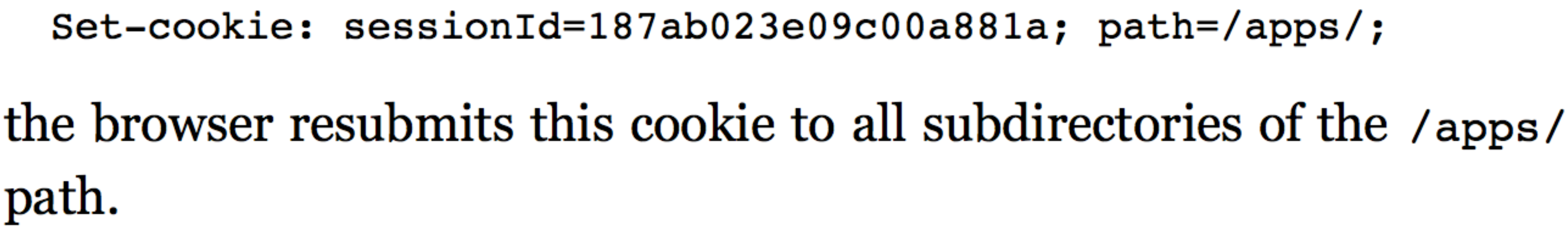
- blogs.com sets a cookie for each user
- Each user can create blogs
    - joe.blogs.com
    - sally.blogs.com
- A blog with JavaScript can steal tokens of other users who read the attacker's blog

Fix

- **There is no way to prevent cookies for an application from being sent to subdomains**
- **Solution: use a different domain name for main app, and scope the domain to this fully qualified name**
  - **www.blogs.com**
- **Cookie won't be sent to joe.blogs.com**

Path

- **Application returns this HTTP header:**



```
Set-cookie: sessionId=187ab023e09c00a881a; path=/apps/;
```

the browser resubmits this cookie to all subdirectories of the `/apps/` path.

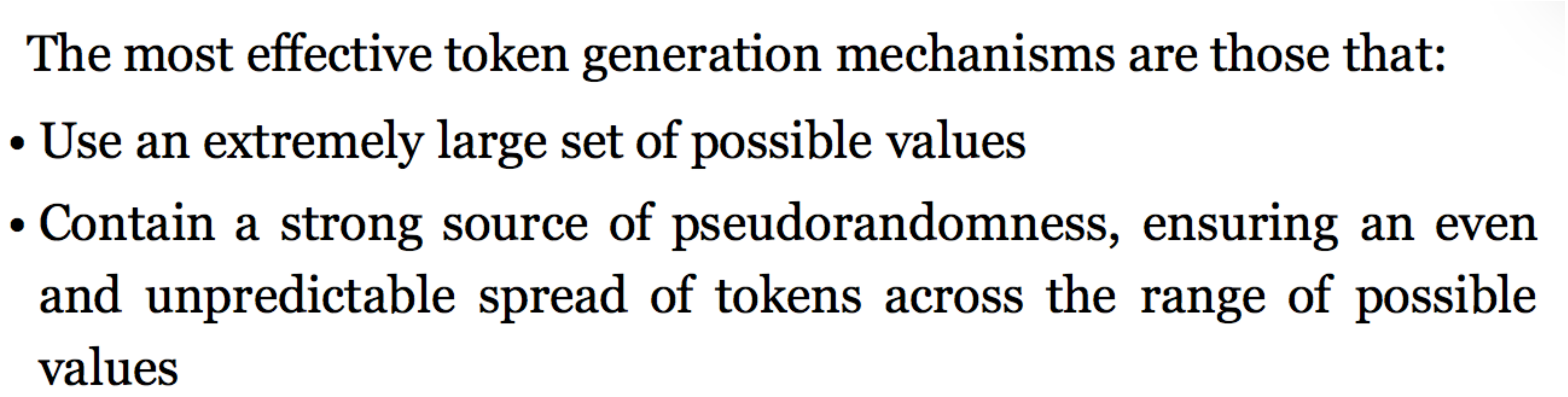
- **Easily defeated by an malicious page on the same domain (Link Ch 7f)**

Securing Session Management

Securing Session Management

- **Generate Strong Tokens**
- **Protect them throughout life cycle, from creation to disposal**

Strong Tokens



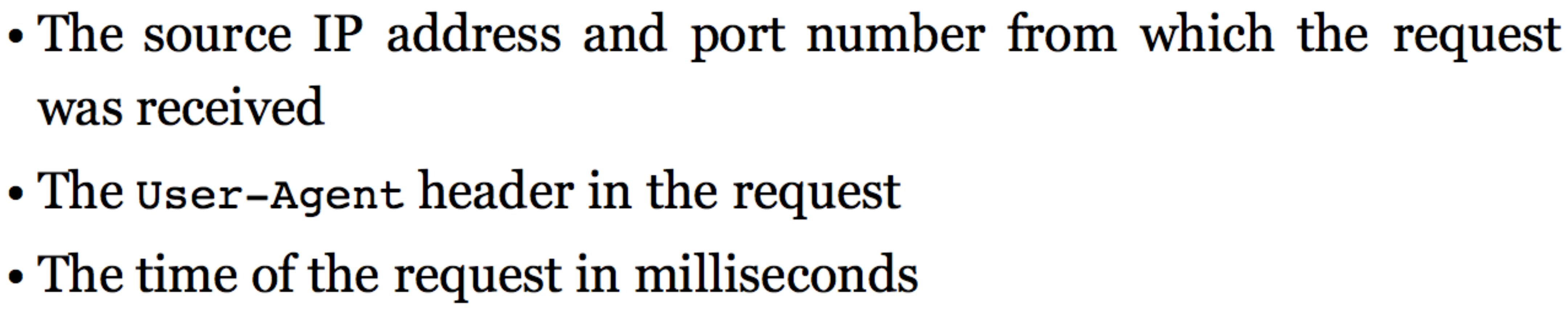
The most effective token generation mechanisms are those that:

- Use an extremely large set of possible values
- Contain a strong source of pseudorandomness, ensuring an even and unpredictable spread of tokens across the range of possible values

Strong Tokens

- **Tokens should contain no meaning or structure**
- **All data about the session's owner and status should be stored on the server in a session object**
- **Some random functions, like java.util.Random, are predictable from a single value**

Sources of Entropy (Randomness)



- The source IP address and port number from which the request was received
- The `User-Agent` header in the request
- The time of the request in milliseconds

- **Good method:**
  - **Add a secret known only to the server, then hash it all**
  - **Change the secret on each reboot**

Protecting Tokens Throughout Their Life Cycle

- **Only transmit over HTTPS**
  - **secure, httponly**
  - **Use HTTPS for every page in application**
- **Don't put session tokens in the URL**
- **Implement a logout function that invalidates session token on the server**

Protecting Tokens Throughout Their Life Cycle

- **Session should expire after a brief period of inactivity, such as 10 minutes**
- **Don't allow concurrent logins**
  - **If a user starts a new session, a new token should be generated, and the old one invalidated**
- **Protect diagnostic or administrative functions that save tokens**
  - **Or don't save tokens in them**

Protecting Tokens Throughout Their Life Cycle

- **Restrict domain and path for session cookies**
- **Audit codebase and remove XSS vulnerabilities**
- **Don't accept tokens submitted by unrecognized users**
  - **Cancel tokens after such a request**

Protecting Tokens Throughout Their Life Cycle

- **Use two-factor authentication**
  - **Makes cross-site request forgery and other session-hijacking methods more difficult**
- **Use hidden fields rather than session cookies**
  - **More difficult to steal because they aren't sent in every request**

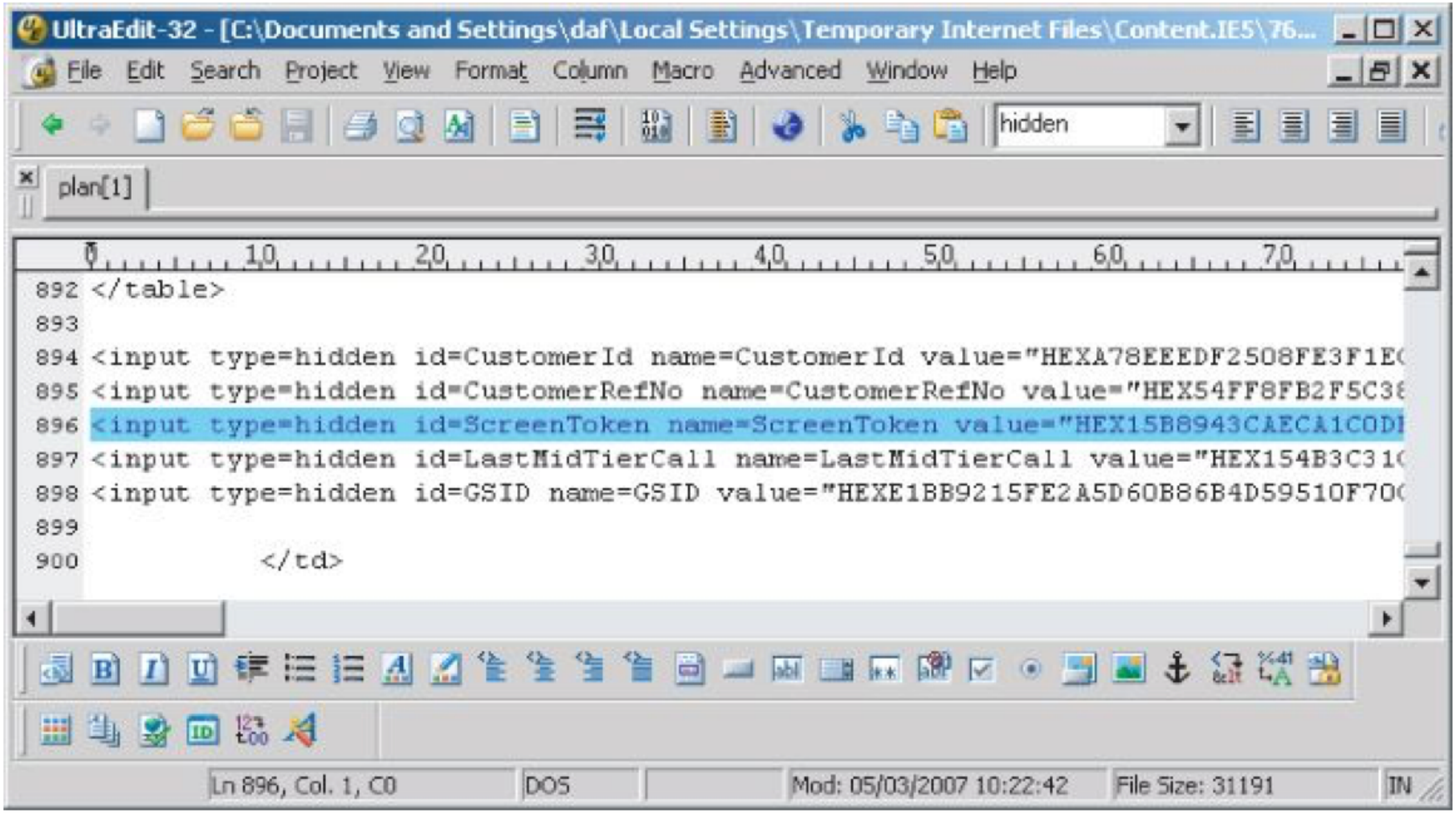
Protecting Tokens Throughout Their Life Cycle

- **Create a fresh session after every authentication**
  - **To prevent session fixation attacks**

Per-Page Tokens

- **A new page token is created every time the user requests an application page**
- **Page token verified on every request**
  - **If it doesn't match, the session is terminated**
- **Prevents pages being used out of order**
- **And blocks an attacker from using a page at the same time as a real user**

Figure 7.12 Per-page tokens used in a banking application



UltraEdit-32 - [C:\Documents and Settings\daf\Local Settings\Temporary Internet Files\Content.IE5\76...
File Edit Search Project View Format Column Macro Advanced Window Help
hidden
plan[1]

```
</table>

<input type=hidden id=CustomerId name=CustomerId value="HEXA78EEEDF2508FE3F1E
<input type=hidden id=CustomerRefNo name=CustomerRefNo value="HEX54FF8FB2F5C3
<input type=hidden id=ScreenToken name=ScreenToken value="HEX15B8943CAECA1C0D
<input type=hidden id=LastMidTierCall name=LastMidTierCall value="HEX154B3C31
<input type=hidden id=GSID name=GSID value="HEXE1BB9215FE2A5D60B86B4D59510F70

          </td>
```

Log, Monitor, and Alert

- **Requests with invalid tokens should raise IDS alerts**
- **Alert users of incidents relating to their sessions**

Reactive Session Termination

- **Terminate session if a request has**
  - **Modified hidden form field or URL query string parameter**
  - **Strings associated with SQL injection or XSS**
  - **Input that violates validation checks, such as length restrictions**

Kahoot!

Ch 7c