IMPACTS ON CONFIDENCE

IMPACTS ON CONFIDENCE

NEGATIVE

POSITIVE

F.U.D.

PLAN OF ACTION

F.U.D.

2.5 Million More Americans Added To The Equifax Security Leak, Wh... Do Now

Winnie Sun, WOMEN@FORBES
FULL BIO
Opinions expressed by Forbes Contributors are their own.

Everything you need to know about EternalBlue – the NSA exploit linked to Petya

WIRED

The EternalBlue vulnerability was first seen following its publication by the Shadowbrokers hacking group

Meltdown and Spectre: 'worst ever' CPU bugs affect virtually all computers

US edition

The Guardian

F.U.D.

KIM ZETTER SECURITY 10.14.14 09:01 PM

THERE IS A NEW SECURITY VULNERABILITY NAMED POODLE, AND IT IS NOT CUTE

cyberscoop

HEALTHCARE

Indiana hospital shuts down systems after ransomware attack

27 JUN 17

'Petya' Ransomware Outbreak Goes Global

KrebsonSecurity
In-depth security news and investigation

INCIDENTS

WannaCry ransomware used in widespread attacks all over the world

By GReAT on May 12, 2017. 5:30 pm

Yahoo says 2013 hack hit all 3 billion user accounts, triple initial estimates

Elizabeth Weise, USATODAY Published 4:45 p.m. ET Oct. 3, 2017 | Updated 7:55 p.m. ET Oct. 3, 2017

VULNERABILITY

THREAT / EXPLOIT

BREACH

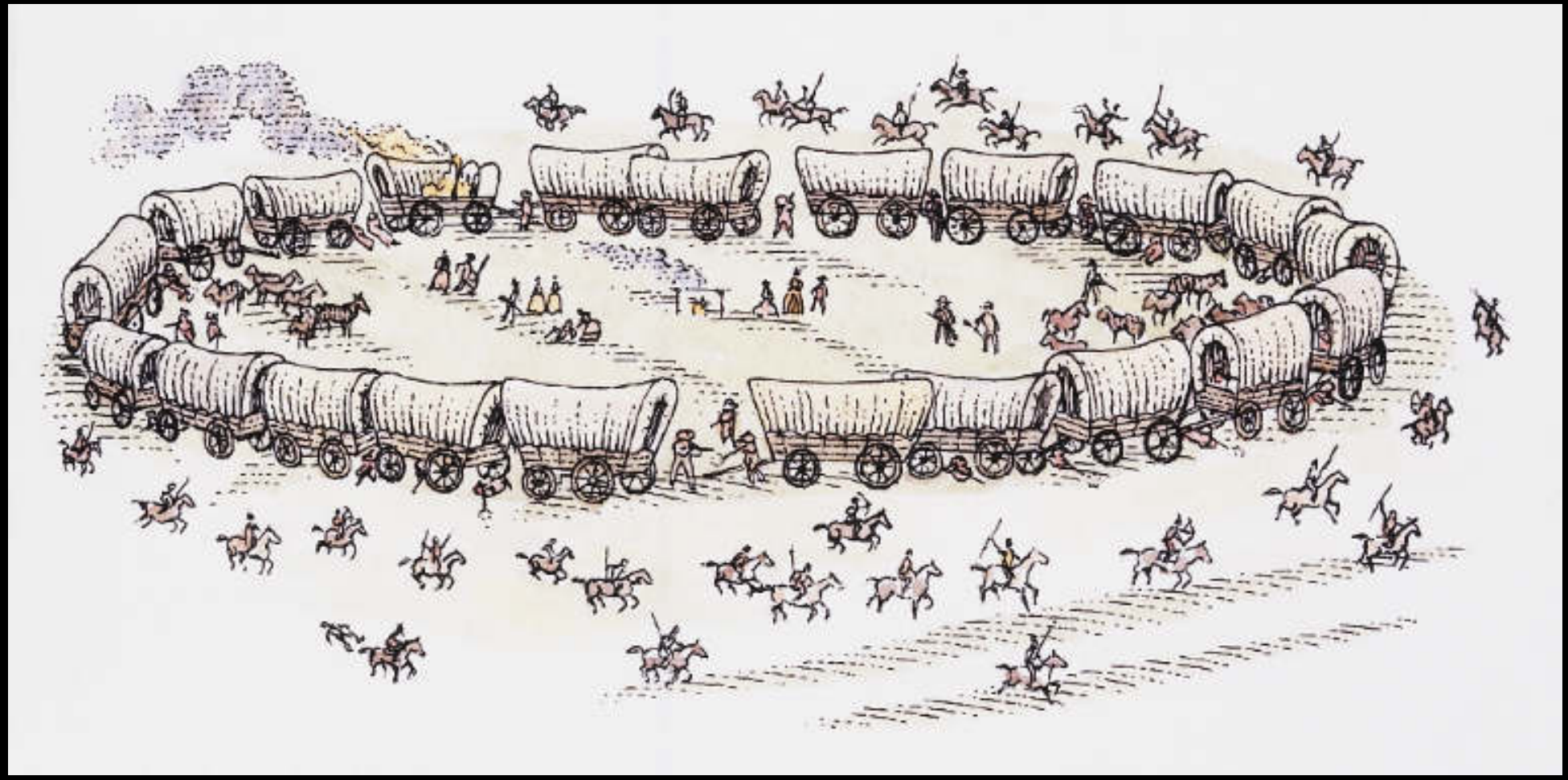


Photo Credit: themoderatevoice.com

2.5 Million More Americans Added To The Equifax Security Leak, Wh To Do Now

Winnie Sun, WOMEN@FORBES
FULL BIO
Opinions expressed by Forbes Contributors are their own.

Everything you need to know about EternalBlue – the NSA exploit linked to Petya

WIRED

The EternalBlue vulnerability was first seen following its publication by the Shadowbrokers hacking group

Meltdown and Spectre: 'worst ever' CPU bugs affect virtually all computers

US edition
The Guardian

F.U.D.

KIM ZETTER SECURITY 10.14.14 09:01 PM

THERE IS A NEW SECURITY ~~VULNERABILITY~~ NAMED POODLE, AND IT IS NOT CUTE

ATTACK

cyberscoop

HEALTHCARE

Indiana hospital shuts down systems after ransomware attack

27 JUN 17 'Petya' Ransomware Outbreak Goes Global

KrebsonSecurity
In-depth security news and investigation

INCIDENTS

WannaCry ransomware used in widespread attacks all over the world

By GReAT on May 12, 2017. 5:30 pm

Yahoo says 2013 hack hit all 3 billion user accounts, triple initial estimates

Elizabeth Weise, USATODAY Published 4:45 p.m. ET Oct. 3, 2017 | Updated 7:55 p.m. ET Oct. 3, 2017

PLAN OF ACTION

Reverse engineering for malware analysis cheat sheet

by @r00tbsd

[WinDBG commands]

```
.attach PID      : Attach to a process
.detach          : End the debugging
.restart         : Restart target application
!sym             : Get state of symbol loading
.reload          : Reload symbol information
g                : Go
p                : Single step
t                : Single trace
pt               : Step to next return
tt               : Trace to next return
pc               : Step to next call
tc               : Trace to next call
pa StopAddr      : Step to address
ta StopAddr      : Trace to address
lm               : List modules
!dlls            : All loaded modules
!peb             : Display formatted view of PEB
!teb             : Display formatted view of TEB
!error 0xN       : Display error
!address         : Display information about memory
~                : List threads
bl               : List breakpoints
bc               : Cancel breakpoints
be               : Enable breakpoints
bd               : Disable breakpoints
bp [Addr]        : Set breakpoint at the address
bm SymPattern    : Set breakpoint at the symbol
ba [r|w|e] Addr  : Set breakpoint on Access
k                : Display call stack
r                : Dump all registers
u                : Disassemble
dN               : Display where N:
                   a: asci chars   | u: Unicode char
                   b: byte + ascii | w: word
                   W: word + ascii | d: dword
                   c: dword + asci | q: qword
                   b: bin + byte   | d: bin + dword
eN Addr Value    : Edit memory
.writemem f A S  : Dump memory
                   f: file name
                   A: Address
                   S: Size (Lx)
```

dec	hex	char	dec	hex	char	dec	hex	char	dec	hex	char
0	0x00	NUL	32	0x20	SPACE	64	0x40	@	96	0x60	`
1	0x01	SOH	33	0x21	!	65	0x41	A	97	0x61	a
2	0x02	STX	34	0x22	"	66	0x42	B	98	0x62	b
3	0x03	ETX	35	0x23	#	67	0x43	C	99	0x63	c
4	0x04	EOT	36	0x24	$	68	0x44	D	100	0x64	d
5	0x05	ENQ	37	0x25	%	69	0x45	E	101	0x65	e
6	0x06	ACK	38	0x26	&	70	0x46	F	102	0x66	f
7	0x07	BEL	39	0x27	'	71	0x47	G	103	0x67	g
8	0x08	BS	40	0x28	(	72	0x48	H	104	0x68	h
9	0x09	TAB	41	0x29	)	73	0x49	I	105	0x69	i
10	0x0A	LF	42	0x2A	*	74	0x4A	J	106	0x6A	j
11	0x0B	VT	43	0x2B	+	75	0x5B	K	107	0x7B	k
12	0x0C	FF	44	0x2C	,	76	0x5C	L	108	0x7C	l
13	0x0D	CR	45	0x2D	-	77	0x5D	M	109	0x7D	m
14	0x0E	SO	46	0x2E	.	78	0x5E	N	110	0x7E	n
15	0x0F	SI	47	0x2F	/	79	0x5F	O	111	0x7F	o
16	0x10	DLE	48	0x30	0	80	0x50	P	112	0x70	p
17	0x11	DC1	49	0x31	1	81	0x51	Q	113	0x71	q
18	0x12	DC2	50	0x32	2	82	0x52	R	114	0x72	r
19	0x13	DC3	51	0x33	3	83	0x53	S	115	0x73	s
20	0x14	DC4	52	0x34	4	84	0x54	T	116	0x74	t
21	0x15	NAK	53	0x35	5	85	0x55	U	117	0x75	u
22	0x16	SYN	54	0x36	6	86	0x56	V	118	0x76	v
23	0x17	ETB	55	0x37	7	87	0x57	W	119	0x77	w
24	0x18	CAN	56	0x38	8	88	0x58	X	120	0x78	x
25	0x19	EM	57	0x39	9	89	0x59	Y	121	0x79	y
26	0x1A	SUB	58	0x3A	:	90	0x5A	Z	122	0x7A	z
27	0x1B	ESC	59	0x3B	;	91	0x5B	[	123	0x7B	{
28	0x1C	FS	60	0x3C	<	92	0x5C	\	124	0x7C	|
29	0x1D	GS	61	0x3D	=	93	0x5D	]	125	0x7D	}
30	0x1E	RS	62	0x3E	>	94	0x5E	^	126	0x7E	~
31	0x1F	US	63	0x3F	?	95	0x5F	_	127	0x7F	DEL

[Registers]

```
x86
EAX       :   return value       |   ESP       :   top of stack
ECX       :   loopcounter, param |   EBP       :   frame pointer
EDX       :   params, data, math |   ESI       :   source
EBX       :   generic            |   EDI       :   destination
EIP       :   instruction point  |
Arguments :   stack (push)

x64
RAX       :   return value       |   RCX       :   argument 1
RDX       :   argument 2         |   R8        :   argument 3
R9        :   argument 4         |   R10:R11   :   for
syscall/sysret
R12:15    :   Must be preserved  |   RDI       :   Must be preserved
RSI       :   Must be preserved  |   RBX       :   Must be preserved
RSP       :   Stack pointer      |

0x1234567012345678
  ================ rax (64 bits)
          ======== eax (32 bits)
              ====  ax (16 bits)
              ==    ah (8 bits)
                ==  al (8 bits)
```

[IDA Pro shortcuts]

```
Navigation:
Enter     :   Jump to operand    |   ESC       :   Jump to previous position
G         :   Go to address      |   Ctrl+L    :   Jump by name
CTRL+P    :   Jump to function   |   X         :   xref
CTRL+E    :   Jump to entry point

Search
Alt+C     :   Next code          |   Ctrl+D    :   Next data
Alt+I     :   Immediate value    |   Crtl+I    :   Next immediate value
Alt+T     :   Text               |   Ctrl+T    :   Next text
Alt+B     :   Sequence of bytes  |   Ctrl+B    :   Next sequence of bytes

Graphing
F12       :   Flow chart         |   Ctrl+F12  :   Function calls

Subviews
Shift+F4  :   Name               |   Shift+F3  :   Functions
Shift+F12 :   Strings            |   Shift+F7  :   Segments

Debugger
F9        :   Start              |   Ctrl+F2   :   Stop process
F7        :   Step into          |   F8        :   Step over
Ctrl+F7   :   Run until return   |   Ctrl+Alt+B:   List breakpoints

Other
C         :   Code               |   D         :   Data
U         :   Undefine           |   N         :   Rename
Shift+;   :   Enter comment      |   ;         :   Enter repeatable comment
P         :   Create function    |   Alt+P     :   Edit function
E         :   Set function end   |   Y         :   Declare function type
M         :   Member enumeration |   Shift+F2  :   Run script
```

[Immunity Debugger shortcuts]

```
F2        :   Set breakpoint     |   F9        :   run
F7        :   Step into          |   F8        :   Step over
Ctrl+F9   :   Execute till ret   |   F12       :   Pause
Alt+B     :   Open breakpoint wi |   Alt+C     :   Open CPU window
Alt+E     :   Open module window |   Alt+L     :   Open log window
Alt+M     :   Open memory window |   Alt+O     :   Open option window
```

[ASM]

```
MOV  : Move (copy)
XCHG : Exchange
PUSH : Push onto stack
POP  : Pop from stack
ADD  : Add
SUB  : Subtract
DIV  : Divide
IDIV : Signed integer divide
MUL  : Multiply
IMUL : Signed integer multiply
INC  : Increment
DEC  : Decrement
SAL  : Shift left
SAR  : Shift right
ROL  : Rotate left
ROR  : Rotate right
NOT  : Invert each bit
AND  : Logical and
OR   : Logical or
XOR  : Logical exclusive or
SHL  : Shift logical left
SHR  : Shift logical right
NOP  : No operation (0x90)
INT  : Interrupt
CALL : Call subroutine
JMP  : Jump
JE   : Jump if equal
JZ   : Jump if zero
JCXZ : Jump if CX zero
JNE  : Jump if not equal
JNZ  : Jump if not zero
JECXZ: Jump if ECX zero
RET  : Return from subroutine
JA   : Jump if above
JAE  : Jump if above or equal
JB   : Jump if below
JBE  : Jump if below or equal
JNA  : Jump if not above
JNAE : Jump if not above or equal
JNB  : Jump if not below
JNBE : Jump if not below or equal
JC   : Jump if carry
JNC  : Jump if no carry
JG   : Jump if greater
JGE  : Jump if greater or equal
JL   : Jump if less
JLE  : Jump if less or equal
JNG  : Jump if not greater
JNGE : Jump if not greater or equal
JNL  : Jump if not less
JNLE : Jump if not less or equal
JO   : Jump if overflow
JNO  : Jump if no overflow
JS   : Jump if sign (= negative)
JNS  : Jump if no sign (= positive)
```

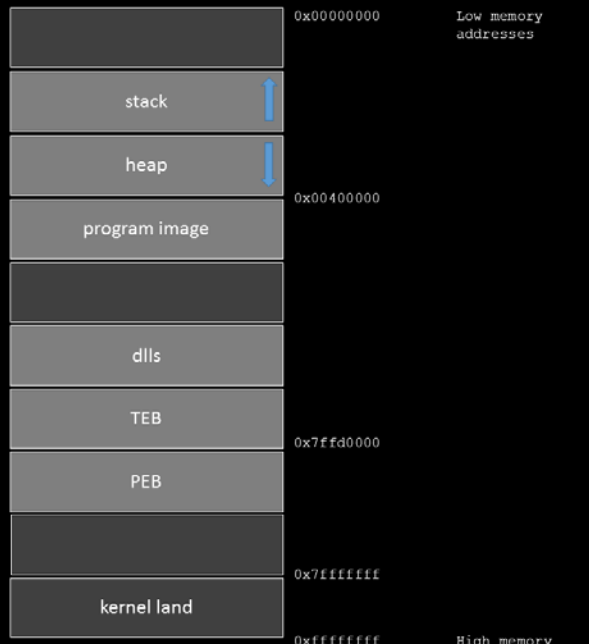
[IDA Pro plugins]

IDA Scope
Diaphora
IDA Tool Bag
IDA Signsrch

[WinDBG plugin]

pykd

[Win32 memory map]



[Python]

```
>>> a="A"
>>> print ord(a)
65
>>> print hex(ord(a))
0x41

>>> b=0x42
>>> print str(b)
66
>>> print chr(b)
B

>>> c="\x41"
>>> print c
A
>>> print ord(c)
65

>>> c="\x90"
>>> print c
▒
>>> import sys
>>> sys.stdout.write(c)
▒

>>> int("0x100", 16)
256
```

```
#XOR
>>> d="\xBE"
>>> x="\xFF"
>>> print hex(ord(d)^ord(x))
0x41
>>> print chr(ord(d)^ord(x))
A

def rol32(num, count):
  num1 = (num << count) & 0xFFFFFFFF
  num2 = (num >> (0x20-count)) &0xFFFFFFFF
  return num1 | num2

def ror32(num, count):
   num1 = (num >> count) & 0xFFFFFFFF
   num2 = (num << (0x20-count)) &0xFFFFFFFF
   return num1 | num2

def ror8(num, count):
   num1 = (num >> count) & 0xFF
   num2 = (num << (0x08 - count)) & 0xFF
   return num1 | num2

def rol8(num, count):
   num1 = (num << count) & 0xFF
   num2 = (num >> (0x08 - count)) & 0xFF
   return num1 | num2

def shl(dest, count):
   return hex(dest << count)

def shr(dest, count):
   return hex(dest >> count)
```

NIST

CYBERSECURITY FRAMEWORK

Function Unique Identifier	Function	Category Unique Identifier	Category
ID	Identify	ID.AM	Asset Management
		ID.BE	Business Environment
		ID.GV	Governance
		ID.RA	Risk Assessment
		ID.RM	Risk Management Strategy
PR	Protect	PR.AC	Access Control
		PR.AT	Awareness and Training
		PR.DS	Data Security
		PR.IP	Information Protection Processes and Procedures
		PR.MA	Maintenance
		PR.PT	Protective Technology
DE	Detect	DE.AE	Anomalies and Events
		DE.CM	Security Continuous Monitoring
		DE.DP	Detection Processes
RS	Respond	RS.RP	Response Planning
		RS.CO	Communications
		RS.AN	Analysis
		RS.MI	Mitigation
		RS.IM	Improvements
RC	Recover	RC.RP	Recovery Planning
		RC.IM	Improvements
		RC.CO	Communications

Source: NIST Framework For Improving Critical Infrastructure Cybersecurity, v. 1.0 (Feb. 12, 2014) at p. 19.

CIS Controls

First 5 CIS Controls
Eliminate the vast majority of your organization's vulnerabilities

1: **Inventory of Authorized and Unauthorized Devices**
2: **Inventory of Authorized and Unauthorized Software**
3: **Secure Configurations for Hardware and Software**
4: **Continuous Vulnerability Assessment and Remediation**
5: **Controlled Use of Administrative Privileges**

All 20 CIS Controls
Secure your entire organization against today's most pervasive threats

6: **Maintenance, Monitoring, and Analysis of Audit Logs**
7: **Email and Web Browser Protections**
8: **Malware Defenses**
9: **Limitation and Control of Network Ports**
10: **Data Recovery Capability**
11: **Secure Configurations for Network Devices**
12: **Boundary Defense**
13: **Data Protection**
14: **Controlled Access Based on the Need to Know**
15: **Wireless Access Control**
16: **Account Monitoring and Control**
17: **Security Skills Assessment and Appropriate Training to Fill Gaps**
18: **Application Software Security**
19: **Incident Response and Management**
20: **Penetration Tests and Red Team Exercises**

Source: Center for Internet Security (www.cisecurity.org/controls)

Australian Government
Department of Defence

Australian Signals Directorate
Reveal Their Secrets – Protect Our Own

STRATEGIES to MITIGATE TARGETED CYBER INTRUSIONS

ASD TOP 4 PREVENTS OVER 85% OF INTRUSIONS

Mitigation 1: Application Whitelisting

Mitigation 2: Patch Applications

Mitigation 3: Patch Operating Systems

Mitigation 4: Minimise Administrative Privileges

Basic Security Hygiene

Basic Security Hygiene

- Patch, Patch, and Make Sure You Did!
- Principle of Least Privilege
- Control Administrative Privileges
- Strong, Well-Protected Passwords
- Multi-Factor Authentication
- Security Awareness Training

PLAN OF ACTION

F.U.D.

Questions?