Why there’s no such thing as cybersecurity

Matt Might, Ph.D.
School of Computing
University of Utah
matt.might.net
@mattmight
Why there’s no such thing as cybersecurity yet

Matt Might, Ph.D.
School of Computing
University of Utah
matt.might.net
@mattmight
Fear
Why?
What?
How?
Why?
Crime
Schnucks breach will likely cost millions

HOW A PAYMENT GOES THROUGH THE SYSTEM

Information that moves through the payment system is encrypted for most of its travels, but at some points is decrypted so different parties can accept the data. That, experts say, is where data are vulnerable. Information is also at risk if stored without being encrypted or obfuscated.

1. The consumer selects a card for payment. The cardholder information is entered into the merchant’s payment system, which could be the point-of-sale (POS) terminal/software or an e-commerce website.

2. The information is sent to an acquirer/processor, whose job it is to route the data through the payments system for processing. With e-commerce transactions, a “gateway” provider may provide the link from the merchant’s website to the acquirer.

3. The acquirer/processor sends the data to the payment brand (e.g. Visa, MasterCard, American Express, etc.) which forwards the information to the issuing bank/processor.

4. The issuing bank/processor verifies that the card is legitimate, not reported lost or stolen, and that the account has the appropriate amount of credit/funds available to pay for the transaction.

5. If so, the issuer generates an authorization number and routes this number back to the card brand. With the authorization, the issuing bank agrees to fund the purchase on the consumer’s behalf.

6. The card brand forwards the authorization code back to the acquirer/processor.

7. The acquirer/processor sends the authorization code back to the merchant.

8. The merchant concludes the sale with the customer.
South Carolina breach exposes 3.6M SSNs

Another 387,000 credit and debit cards also exposed in Department of Revenue intrusion, but most were encrypted

By Jaikumar Vijayan
October 26, 2012 07:41 PM ET    7 Comments

Computerworld - In the biggest data compromise of the year, Social Security Numbers (SSN) belonging to about 3.6 million residents in South Carolina have been exposed in an intrusion into a computer at the state's Department of Revenue.

Another 387,000 credit and debit card numbers were also exposed in the September attack, the state Department of Revenue said in a statement Friday. However, out of that number only about 16,000 of the credit and debit cards were unencrypted, the department added. The SSNs, meanwhile, do not appear to have been encrypted.
War
هوایی‌مایی بی‌شرفته جاسوسی آمریکا

RQ170

تسلط مدافعان ایران اسلامی بر پرواز کات شیطان امریکا
Hackers in China Attacked The Times for Last 4 Months

Computer Assaults Tied to Reporting on Premier

By NICOLE PERLROTH

SAN FRANCISCO — For the last four months, Chinese hackers have persistently attacked The New York Times, infiltrating its computer systems and getting passwords for its reporters and other employees.

After surreptitiously tracking the intruders to study their movements and help erect better defenses to block them, The Times and computer security executives, and Jim Yardley, The Times’s South Asia bureau chief in India, who previously worked as bureau chief in Beijing.

“Computer security experts found no evidence that sensitive e-mails or files from the reporting of our articles about the Wen family were accessed, downloaded or copied,” said Jill Abramson, executive editor of The Times. The hackers tried to cloak the
Hacktivism
ACS:Law
How?
Just Arrived: Malware Analyst’s Cookbook

DEC 1ST  Posted by Dustin  in Exploit Development

Author Michael Ligh was very gracious to send me a review copy of his new book Malware Analyst’s Cookbook and DVD: Tools and Techniques for Fighting Malicious Code. I took a quick browse through it when I opened it and it looks REALLY GOOD. If it’s anything like the articles on Michael’s website, I know I’m in for a damn good read!

I’m planning on starting it this Saturday due to some other priorities so heads up for a review post in the future or check it out for yourself

Malware Analyst’s Cookbook and DVD: Tools and Techniques for Fighting Malicious Code (Paperback)
By (author) Michael Ligh, Steven Adair, Blake Hartstein, Matthew Richard
List Price: $59.99 USD
<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>506</td>
<td>152.25091</td>
<td>192.168.12.21</td>
<td>209.132.177.50</td>
<td>TCP</td>
<td>http &gt; 48890 [SYN] Seq=0 Len=0 MSS=1460 TSV=1535</td>
</tr>
<tr>
<td>507</td>
<td>152.31125</td>
<td>209.132.177.50</td>
<td>192.168.12.21</td>
<td>TCP</td>
<td>http &gt; 48890 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=5840</td>
</tr>
<tr>
<td>508</td>
<td>152.31132</td>
<td>209.132.177.50</td>
<td>192.168.12.21</td>
<td>TCP</td>
<td>http &gt; 48890 [ACK] Seq=1 Ack=1 Win=8576 Len=0</td>
</tr>
<tr>
<td>509</td>
<td>152.31154</td>
<td>192.168.12.21</td>
<td>209.132.177.50</td>
<td>HTTP</td>
<td>GET / HTTP/1.1</td>
</tr>
<tr>
<td>510</td>
<td>152.38737</td>
<td>209.132.177.50</td>
<td>192.168.12.21</td>
<td>TCP</td>
<td>[TCP segment of a reassembled PDU]</td>
</tr>
<tr>
<td>511</td>
<td>152.40516</td>
<td>209.132.177.50</td>
<td>192.168.12.21</td>
<td>TCP</td>
<td>[TCP segment of a reassembled PDU]</td>
</tr>
<tr>
<td>512</td>
<td>152.40520</td>
<td>192.168.12.21</td>
<td>209.132.177.50</td>
<td>TCP</td>
<td>[TCP segment of a reassembled PDU]</td>
</tr>
<tr>
<td>513</td>
<td>152.41351</td>
<td>192.168.12.21</td>
<td>209.132.177.50</td>
<td>TCP</td>
<td>[TCP segment of a reassembled PDU]</td>
</tr>
<tr>
<td>514</td>
<td>152.41356</td>
<td>192.168.12.21</td>
<td>209.132.177.50</td>
<td>TCP</td>
<td>[TCP segment of a reassembled PDU]</td>
</tr>
<tr>
<td>515</td>
<td>152.45058</td>
<td>192.168.12.21</td>
<td>209.132.177.50</td>
<td>TCP</td>
<td>[TCP segment of a reassembled PDU]</td>
</tr>
<tr>
<td>516</td>
<td>152.47685</td>
<td>209.132.177.50</td>
<td>192.168.12.21</td>
<td>TCP</td>
<td>[TCP segment of a reassembled PDU]</td>
</tr>
<tr>
<td>517</td>
<td>152.47690</td>
<td>209.132.177.50</td>
<td>192.168.12.21</td>
<td>TCP</td>
<td>[TCP segment of a reassembled PDU]</td>
</tr>
</tbody>
</table>

Frame 507 (74 bytes on wire, 74 bytes captured)
Ethernet II, Src: Amit_04:ae:54 (00:50:18:04:ae:54), Dst: Intel_e3:01:f5 (00:0c:fe:e3:01:f5)
Internet Protocol, Src: 209.132.177.50 (209.132.177.50), Dst: 192.168.12.21 (192.168.12.21)
Transmission Control Protocol, Src Port: http (80), Dst Port: 48890 (48890), Seq: 0, Ack: 1, Len: 0

Destination port: 48890 (48890)
Sequence number: 0 (relative sequence number)
Acknowledgement number: 1 (relative ack number)
Header length: 40 bytes
Flags: 0x12 (SYN, ACK)
Window size: 5792
Checksum: 0x99df [correct]
Options: (20 bytes)

Source Port (tcp.srcport), 2: P: 1096 D: 1096 M: 0 Drops: 0
<table>
<thead>
<tr>
<th>IP Address</th>
<th>Name</th>
<th>OS Name</th>
<th>Version</th>
<th>Purpose</th>
<th>Services</th>
<th>Vulns</th>
<th>Notes</th>
<th>Updated</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.0.23</td>
<td>metasploitable</td>
<td>Linux (Ubuntu)</td>
<td>server</td>
<td>14</td>
<td>1</td>
<td>9</td>
<td></td>
<td>about 1 hour ago</td>
<td>Looted</td>
</tr>
</tbody>
</table>
Why so vulnerable?
Programmers screw up!
vulnerability = mistake
no mistake = no vulnerability
Example: Buffer overflow
John Day
John Day$*
John Day $ *
takeover 42
Defense
Discovery → Private disclosure → Patch release → Exploit creation → Patched
Discovery

Public disclosure

Patch release

Exploit creation

Patched
Discovery → Exploit creation → Attack
Discovery → Weapon creation
6 P’s
Passwords.
Patches.
Popularity.
Protection.
Preparation.
Prayer.
Future
How?
Math.
Lots of Math.
\[a^2 + b^2 = c^2\]
\[ \forall \sigma \in \{ \sigma' \mid I(program) \Rightarrow^* \sigma' \} : \sigma \in WF \]
How?
.class MyActivity

.method public MyActivity
    invokedynamic activateMic
.end method
.end class
Vulnerability!
```
.class MyActivity
.method public MyActivity
   invokedynamic activateMic
.end method
.end class
```
ψ
\[ F = ma \]
\[
\begin{align*}
\langle \text{nop} :: \tilde{\text{stmt}}, fp, \sigma, \kappa \rangle & \mapsto \langle \tilde{\text{stmt}}, fp, \sigma, \kappa \rangle \\
\langle \text{move-object}(r_d, r_s) :: \tilde{\text{stmt}}, fp, \sigma, \kappa \rangle & \mapsto \langle \tilde{\text{stmt}}, fp, \sigma[(r_d, fp) \mapsto \sigma(r_s, fp)], \kappa \rangle \\
\langle \text{return-void :: \tilde{\text{stmt}}, fp', \sigma, \text{fink}(\tilde{\text{stmt}}, fp, \kappa) \rangle & \mapsto \langle \tilde{\text{stmt}}, fp, \sigma \rangle \\
\langle \text{return-object}(r) :: \tilde{\text{stmt}}, fp', \sigma, \text{fink}(\tilde{\text{stmt}}, fp, \kappa) \rangle & \mapsto \langle \tilde{\text{stmt}}, fp, \sigma[(\text{ret}, fp) \mapsto \sigma(n, fp')], \kappa \rangle \\
\langle \text{const}(r, c) :: \tilde{\text{stmt}}, fp, \sigma, \kappa \rangle & \mapsto \langle \tilde{\text{stmt}}, fp, \sigma[(r, fp) \mapsto c], \kappa \rangle \\
\langle \text{throw}'(r) :: \tilde{\text{stmt}}, fp, \sigma, \kappa \rangle & \mapsto \langle \text{S}(\ell'), fp', \sigma[(\text{exn}, fp') \mapsto \sigma(r, fp)], \kappa' \rangle \\
& \quad \text{where } (\ell', fp', \kappa') = \mathcal{H}(\ell, fp, \kappa) \\
\langle \text{goto}(\ell) :: \tilde{\text{stmt}}, fp, \sigma, \kappa \rangle & \mapsto \langle \text{S}(\ell), fp, \sigma, \kappa \rangle \\
\langle \text{new-instance}(r, \tau) :: \tilde{\text{stmt}}, fp, \sigma, \kappa \rangle & \mapsto \langle \tilde{\text{stmt}}, fp, \sigma[(r, fp) \mapsto \sigma], \kappa \rangle \\
& \quad \text{where } \sigma = \text{new}(\varsigma) \\
\langle \text{if-eq}(r, r', \ell) :: \tilde{\text{stmt}}, fp, \sigma, \kappa \rangle & \mapsto \langle \text{S}(\ell), fp, \sigma, \kappa \rangle \text{ if } \sigma(r, fp) = \sigma(r', fp) \\
\langle \text{if-neq}(r, r', \ell) :: \tilde{\text{stmt}}, fp, \sigma, \kappa \rangle & \mapsto \langle \tilde{\text{stmt}}, fp, \sigma, \kappa \rangle \text{ if } \sigma(r, fp) \neq \sigma(r', fp) \\
\langle \text{i-get}(r_d, r_s, \text{field}) :: \tilde{\text{stmt}}, fp, \sigma, \kappa \rangle & \mapsto \langle \tilde{\text{stmt}}, fp, \sigma[(r_d, fp) \mapsto \sigma(\text{field})], \kappa \rangle \\
& \quad \text{where } \sigma(r_s, fp) = \sigma \text{ and } \sigma.\text{field} = a \\
\langle \text{i-put}(r_v, r_s, \text{field}) :: \tilde{\text{stmt}}, fp, \sigma, \kappa \rangle & \mapsto \langle \tilde{\text{stmt}}, fp, \sigma[a \mapsto \sigma(r_v, fp)], \kappa \rangle \\
& \quad \text{where } \sigma(r_s, fp) = \sigma \text{ and } \sigma.\text{field} = a \\
\langle \text{invoke-direct}(r_0, \ldots, r_n, id) :: \tilde{\text{stmt}}, fp, \sigma, \kappa \rangle & \mapsto \langle \text{M}(id), fp', \sigma', \text{fink}(\tilde{\text{stmt}}, fp, \kappa) \rangle \\
& \quad \text{where } \sigma' = \sigma[(0, fp') \mapsto \sigma(r_0, fp), \ldots, (n, fp') \mapsto \sigma(r_n, fp)] \\
& \quad \quad fp' = \text{alloc}(\varsigma) \\
\langle \text{invoke-virtual}(r_0, \ldots, r_n, id) :: \tilde{\text{stmt}}, fp, \sigma, \kappa \rangle & \mapsto \langle \text{V}(id, \sigma(r_0, fp)), fp', \sigma', \text{fink}(\tilde{\text{stmt}}, fp, \kappa) \rangle \\
& \quad \text{where } \sigma' = \sigma[(0, fp') \mapsto \sigma(r_0, fp), \ldots, (n, fp') \mapsto \sigma(r_n, fp)] \\
& \quad \quad fp' = \text{alloc}(\varsigma) \\
\langle \text{unop}(r_d, r_s) :: \tilde{\text{stmt}}, fp, \sigma, \kappa \rangle & \mapsto \langle \tilde{\text{stmt}}, fp, \sigma[(r_d, fp) \mapsto v], \kappa \rangle \\
& \quad \text{where } v = \delta(\text{unop}, \sigma(r_s, fp)) \\
\langle \text{binop}(r_d, r_1, r_2) :: \tilde{\text{stmt}}, fp, \sigma, \kappa \rangle & \mapsto \langle \tilde{\text{stmt}}, fp, \sigma[(r_d, fp) \mapsto v], \kappa \rangle \\
& \quad \text{where } v = \delta(\text{binop}, \sigma(r_1, fp), \sigma(r_2, fp))
\end{align*}
\]
\begin{align*}
\text{(nop :: }& \text{ stnt, fp, } \sigma, k, t) \rightarrow (\text{stnt, fp, } \sigma, k, t) \\
\text{(move-object }&(r_d, r_s) :: \text{ stnt, fp, } \sigma, k, t) \rightarrow (\text{stnt, fp, } \sigma \cup [(r_d, fp) \rightarrow \sigma(r_s, fp)], k, t) \\
\text{(return-void :: }& \text{ stnt', fp', } \sigma, \text{ fnk(stnt, fp, } \tilde{a}_n)) \rightarrow (\text{stnt, fp, } \sigma, k) \text{ if } k \in \sigma(\tilde{a}_n) \\
\text{(return-object }&(r :: \text{ stnt', fp', } \sigma, \text{ fnk(stnt, fp, } \tilde{a}_n)) \rightarrow (\text{stnt, fp, } \sigma \cup [(r, \tilde{fp}) \rightarrow \sigma(n, \tilde{fp}'), k] \text{ if } k \in \sigma(\tilde{a}_n) \\
\text{(const(r, c) :: }& \text{ stnt, fp, } \sigma, k, t) \rightarrow (\text{stnt, fp, } \sigma \cup [(r, \tilde{fp}) \rightarrow c], k, t) \\
\text{(throw'(r :: stnt, fp, } \sigma, k, t) \rightarrow (S(t'), fp', \sigma \cup [(\text{exn, } fp') \rightarrow \sigma(r, \tilde{fp})], k') \\
\text{ where } (t', fp', k') = \hat{H}_0(t, fp, k) \\
\text{(goto(l :: stnt', fp, } \sigma, k, t) \rightarrow (S(l), fp, \sigma, k, t) \\
\text{(new-instance(r, t) :: stnt, fp, } \sigma, k, t) \rightarrow (\text{stnt, fp, } \sigma \cup [(r, fp) \rightarrow o], k, t) \\
\text{ where } o = \text{new}(\varsigma) \\
\text{(if-eq(r, r', t) :: stnt, fp, } \sigma, k, t) \rightarrow (S(t), fp, \sigma, k, t) \\
\text{ if } \exists v_1 \in \sigma(r, \tilde{fp}), \exists v_2 \in \sigma(r', \tilde{fp}), v_1 = v_2 \\
\text{(if-eq(r, r', t) :: stnt, fp, } \sigma, k, t) \rightarrow (\text{stnt, fp, } \sigma, k, t) \\
\text{ if } \exists v_1 \in \sigma(r, \tilde{fp}), \exists v_2 \in \sigma(r', \tilde{fp}), v_1 \neq v_2 \\
\text{(iget(r_d, r_s, field) :: stnt, fp, } \sigma, k, t) \rightarrow (\text{stnt, fp, } \sigma \cup [(r_d, \tilde{fp}) \rightarrow \sigma(a)], k, t) \\
\text{ where } \sigma(r_s, fp) \supset o \text{ and } o.\text{field} = a \\
\text{(iput(r_v, r_s, field) :: stnt, fp, } \sigma, k, t) \rightarrow (\text{stnt, fp, } \sigma \cup [a \rightarrow \sigma(r_v, \tilde{fp})], k, t) \\
\text{ where } \sigma(r_s, fp) \supset o \text{ and } o.\text{field} = a \\
\text{(invoke-direct(r_0, ..., r_n, id) :: stnt, fp, } \sigma, k, t) \rightarrow (\mathcal{M}(id), f_{p'}, \sigma'', \text{fnk(stnt, fp, } \tilde{a}_n), t') \\
\text{ where } \sigma'' = \sigma' \cup [(0, fp) \rightarrow \sigma(r_0, \tilde{fp}), \ldots, (n, \tilde{fp}) \rightarrow \sigma(r_n, \tilde{fp}]] \\
\sigma' = \sigma \cup [\tilde{a}_n \rightarrow k] \\
\tilde{fp'} = \hat{alloc}(\varsigma) \\
\tilde{a}_n = \hat{alloc}(\varsigma) \\
\tilde{t'} = \hat{tick}(l) \\
\text{(invoke-virtual(r_0, ..., r_n, id) :: stnt, fp, } \sigma, k, t) \rightarrow (\mathcal{V}(id, v), f_{p'}, \sigma'', \text{fnk(stnt, fp, } \sigma, k), t') \text{ if } v \in \sigma(r_0, \tilde{fp}) \\
\text{ where } \sigma'' = \sigma' \cup [(0, fp') \rightarrow \sigma(r_0, \tilde{fp}), \ldots, (n, \tilde{fp}') \rightarrow \sigma(r_n, \tilde{fp})] \\
\sigma' = \sigma \cup [\tilde{a}_n \rightarrow k] \\
\tilde{fp'} = \hat{alloc}(\varsigma) \\
\tilde{a}_n = \hat{alloc}(\varsigma) \\
\tilde{t'} = \hat{tick}(l) \\
\text{(unop(r_d, r_s) :: stnt, fp, } \sigma, k) \rightarrow (\text{stnt, fp, } \sigma \cup [(r_d, fp) \rightarrow v], k) \\
\text{ where } v \in \delta(\text{unop}, \sigma(r_s, fp)) \\
\text{(binop(r_d, r_1, r_2) :: stnt, fp, } \sigma, k) \rightarrow (\text{stnt, fp, } \sigma \cup [(r_d, fp) \rightarrow v], k) \\
\text{ where } v \in \delta(\text{binop}, \sigma(r_1, \tilde{fp}), \sigma(r_2, \tilde{fp}))
\end{align*}
Theorem
A fatal exception 0E has occurred at 0028:C0011E36 in UVD UMM(01) + 00010E36. The current application will be terminated.

- Press any key to terminate the current application.
- Press CTRL+ALT+DEL again to restart your computer. You will lose any unsaved information in all applications.

Press any key to continue _
Passwords.
Patches.
Popularity.
Protection.
Preparation.
Prayer.
Thanks!
matt.might.net
matt@might.net
@mattmight