CSE 40567 / 60567: Computer Security

Network Security 6
Homework #8 Released
Due: 5/2/17 at 11:59PM Eastern Time

See Assignments Page on the course website for details
Course Instructor Feedback (CIF)
Deadline: 11:59PM, 5/7/17
Rapid IMAP Connections

alert tcp $EXTERNAL_NET any -> $HOME_NET 143 (msg:"ET SCAN Rapid IMAP Connections - Possible Brute Force Attack"; flags: S,12; threshold: type both, track by_src, count 30, seconds 60; reference:url,doc.emergingthreats.net/2002994; classtype:misc-activity; sid:2002994; rev:6;)

Baidu Toolbar

alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS
(msg:"ET MALWARE Baidu.com Spyware Bar Pulling Data";
flow:to_server,established; content:"/cpro/ui/ui"; nocase;
http_uri; content:"baidu.com"; nocase; http_header;
content:!'Referer|3a| "'; nocase; http_header;
reference:url, doc.emergingthreats.net/bin/view/Main/
2003578; classtype:trojan-activity; sid:2003578; rev:9;)
Baidu Toolbar

Origin of this toolbar is somewhat unclear

https://www.fixyourbrowser.com/removal-instructions/remove-baidu-toolbar/
Fake updates for Windows

alert tcp $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any (msg:"ET CURRENT_EVENTS Fake MS Security Update (Jar)"; flow:established,from_server; file_data; content:"Microsoft Security Update"; content:"applet_ssv_validated"; fast_pattern:only; flowbits:set,et.exploitkitlanding; classtype:trojan-activity; sid:2017549; rev:1;)

SQL Injection

SQL Injection in Cookie

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
(msg:"ET WEB_SERVER Possible DELETE FROM SQL Injection In Cookie"; flow:to_server,established; content:"DELETE "; nocase; http_cookie; content:"FROM"; nocase; http_cookie; pcre:"/\x0a\x0dCookie\x3a\[^
\]DELETE.+FROM/i"; reference:url,www.w3schools.com/Sql/sql_delete.asp;
reference:url,en.wikipedia.org/wiki/SQL_injection;
reference:url,doc.emergingthreats.net/2009772;
classstype:web-application-attack; sid:2009772; rev:7;)

alert tcp any any -> $HOME_NET !$HTTP_PORTS (msg:"ET CURRENT EVENTS Malformed HeartBeat Request"; flow:established,to_server; content:"|18 03|"; depth:2; byte_test:1,<,4,2; content:"|01|"; offset: 5; depth:1; byte_extract:2,3,record_len; byte_test: 2,>,2,3; byte_test:2,> ,record_len,6; threshold:type limit,track by_src,count 1,seconds 120; flowbits:set,ET.MalformedTLSHB; reference:cve, 2014-0160; reference:url, blog.inliniac.net/2014/04/08/detecting-openssl-heartbleed-with-suricata/; reference:url, heartbleed.com/; reference:url, blog.fox-it.com/2014/04/08/openssl-heartbleed-bug-live-blog/; classtype:bad-unknown; sid:2018372; rev:2;)


Heartbleed
Tor Detection

alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:"ET POLICY Onion2Web Tor Proxy Cookie"; flow:established,to_server; content:"onion2web_confirmed="; fast_pattern:only; content:"onion2web_confirmed="; http_cookie; reference:md5,a46e609662eb94a726fcb4471b7057d4; reference:md5,2b62c6b6bce4b7e4f437e4fc46d3; reference:url,github.com/starius/onion2web; classtype:policy-violation; sid:2020324; rev:1;)
alert udp $HOME_NET any -> any 53 (msg:"ET TROJAN Large DNS Query possible covert channel"; content:"|01 00 00 01 00 00 00 00 00 00 00|"; fast_pattern; depth:10; offset:2; dsize:>300; content:!!"youtube|03|com|00|"; content:!!"sophosxl|03|net|00|"; content:!!"|0a|hashserver|02|cs|0a|trendmicro|03|com|00|"; content:!!"spamhaus|03|org|00|"; classtype:bad-unknown; sid:2013075; rev:8;)

(Not Very) Covert Channel
Denial of Service Detection

alert tcp $EXTERNAL_NET 10000: -> $HOME_NET 0:1023 (msg:"ET DOS Potential Tsunami SYN Flood Denial Of Service Attempt"; flags:S; dsize:>900; threshold: type both, count 20, seconds 120, track by_src; reference:url,security.radware.com/uploadedFiles/Resources_and_Content/Threat/TsunamiSYNFloodAttack.pdf; classtype:attempted-dos; sid:2019404; rev:2;)

High false positive potential
Out-of-date Software Detection

alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:"ET POLICY Vulnerable Java Version 1.8.x Detected"; flow:established,to_server; content:"Java/1.8.0_"; http_header; content:!'73'; within:2; http_header; content:!'74'; within:2; http_header; flowbits:set,ET.http.javaclient.vulnerable; threshold: type limit, count 2, seconds 300, track by_src; reference:url,javatester.org/version.html; classtype:bad-unknown; sid:2019401; rev:9;)

General limitations of intrusion detection

1. In the general case, detecting intrusions reduces to a computationally hard problem
   - (Cohen 1994) proved that virus detection is as hard as the halting problem
   - We will never achieve a perfect IDS solution

2. Some attacks generate errors, and others do not
   - The latter case is problematic

3. Reactive IDS complicates policies regarding network traffic
   - Opens the door to DoS
   - May break normal use with unanticipated actions
General limitations of intrusion detection

4. High-cost of false alarms
   ‣ Check your snort logs — the system cries wolf more often than you’d expect.

5. Policy problem: redlining
   ‣ Certain netblocks are routinely flagged (China, Russia)
   ‣ This can lead to unintended discrimination
Specific problems detecting network attacks

TCP 60 45680 > 50624 [ACK] Seq=1347 Ack=214330
Win=64975 Len=0 [Malformed Packet]

• The Internet is a very noisy environment
  ‣ Many (most?) malformed packets are the result of software bugs
  ‣ Drives up the false positive rate of the IDS

• There are very few real attacks
  ‣ 10 real attacks per million with a false alarm rate of 0.1%. What is the ratio of false to real alarms?

100:1
Specific problems detecting network attacks

$ wc -l emerging-all.rules.txt
73884 emerging-all.rules.txt

• Many attacks are specific to particular versions of software
  ‣ IDS needs a large and constantly changing library of attack signatures

• Encrypted traffic is a stumbling block
  ‣ IPSec protects both headers and payloads
Evading IDS

Example: polymorphic shellcode

Static shellcode is trivial to detect via a signature:

```
\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b
\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd
\x80\xe8\xdc\xff\xff\xff/bin/sh``

Solution for attacker: encrypt the shellcode

<table>
<thead>
<tr>
<th>FAKENOP</th>
<th>DecipherRoutine</th>
<th>Encrypted Shellcode</th>
<th>Bytes to Cram</th>
<th>Return Address</th>
</tr>
</thead>
</table>

Polymorphic shellcode routine

**FAKENOP:**

Generate two-byte instructions, the second byte of which is a one-byte instruction or the first byte of a two-byte instruction

\x15\x11\xF8\xFA\x81\xF9\x27\x2F\x90\x9E

Start at first byte

ADC $0x11F8FA81
STC
DAA
DAS
NOP
SAHF

Start at second byte

ADC %eax,%edx
CMP %ecx,$0x272F909E

Or
Polymorphic shellcode routine

**DecipherRoutine:**

This can’t be the same between attacks. 2 strategies:

1. **Use the same routine, but change the instructions**

2. **Generate different routines for decipher**

   - Generate routines which cipher with several instructions XOR, ADD, ROR
   - Use random registers
   - Four byte encryption
Polymorphic shellcode routine

Ideal strategy: make encrypted shellcode resemble “normal” traffic

Distribution of encrypted shellcode bytes should be consistent with this distribution

Estimate by sniffing network
Polymorphic shellcode routine

What about the NOPs and distributional sampling?

• Sample these instructions so that the distribution is also consistent with normal traffic

• Problem: the set of instructions is smaller than the set of all the hex codes in the network traffic
Semantics-aware IDS

Strategy: don’t look at the code, instead, consider its behavior

Internet -> Traffic Classifier

Binary Detection and Extraction

Disassembler

Intermediate Rep. Generator

Semantic Analyzer
Machine Learning

People are good at reading logs, computers are not

- Can we teach machines to read logs like humans?

Machine Learning: learning and making predictions from data without explicit programming
Anomaly Detection IDS

**Assumption:** Attacks exhibit characteristics that are different than those of normal traffic

Promises to find novel attacks without anticipating specifics

Machine learning works well in other domains (e.g., computer vision, natural language processing). Why not IDS?

**Rarely deployed in 2016**

Anomaly Detection IDS: Data Space

- Session Duration
- Session Volume
Anomaly Detection IDS: Training

Create profile of normal activity
Anomaly Detection IDS: Testing

Session Duration

Session Volume

Anomaly
The difficulty of anomaly detection

Significant differences from other problem domains exist:

• How do we find the opposite of normal?
• What is the cause of the anomaly?
• How can we make sure it works?
• What is the feature space?
• Can the attacker fool the machine learning?
Feature space

Feature X

Label A

Label B

Label C

Feature Y
Labeled Anomaly space

Feature X

Label A

Trigger Alert

Label: Unknown

Feature Y

Label B

Label C

202
Course Roadmap

Basics
(weeks 1 & 2)

3 Core Areas
(weeks 3 - 6)

The Web
(weeks 15 & 16)

(weeks 6 - 10)  (weeks 11 - 15)
Anatomy of a Website Hack

New file appears in wordpress file hierarchy:

Obfuscated Code

Accessing it via a browser yields backdoor interface:

Image Credits: http://bartblaze.blogspot.com/2015/03/c99shell-not-dead.html
What is this code?

$shver = "2.1 madnet edition ADVANCED";

$timelimit = 0;
$host_allow = array("*");
@login_txt = "Admin area";
$accessdeniedmess = "die like the rest";
$gzipencode = TRUE;
$c99sh_sourcesurl = ""; //Sources-server
$filestealth = TRUE;
$donated_html = "";
$donated_act = array("");
$curdir = "./";
$tmpdir = "";
$tmpdir_log = "./";
Path to privilege escalation

```php
$cmdaliases = array(
    array("-----------------------------------------------------------", "ls -la"),
    array("find all suid files", "find / -type f -perm -04000 -ls"),
    array("find suid files in current dir", "find . -type f -perm -04000 -ls"),
    array("find all sgid files", "find / -type f -perm -02000 -ls"),
    array("find sgid files in current dir", "find . -type f -perm -02000 -ls"),
    array("find config.inc.php files", "find / -type f -name config.inc.php"),
    array("find config* files", "find / -type f -name "config*""),
    array("find config* files in current dir", "find . -type f -name "config*""),
    array("find all writable folders and files", "find / -perm -2 -ls"),
    array("find all writable folders and files in current dir", "find . -perm -2 -ls"),
    array("find all service.pwd files", "find / -type f -name service.pwd"),
    array("find service.pwd files in current dir", "find . -type f -name service.pwd"),
    array("find all .htpasswd files", "find / -type f -name .htpasswd"),
    array("find .htpasswd files in current dir", "find . -type f -name .htpasswd"),
    array("find all .bash_history files", "find / -type f -name .bash_history"),
    array("find .bash_history files in current dir", "find . -type f -name .bash_history"),
    array("find all .fetchmailrc files", "find / -type f -name .fetchmailrc"),
    array("find .fetchmailrc files in current dir", "find . -type f -name .fetchmailrc"),
    array("list file attributes on a Linux second extended file system", "lsattr -va"),
    array("show opened ports", "netstat -an | grep -i listen")
);
```

http://pastebin.com/L02T4kFp
How does this happen?

- Misconfigurations
- Brute Force Attacks
- SQL Injection
- Cross-Site Scripting (XSS)
- Cross-Site Request Forgery (CSRF)
Vulnerabilities in Web Software: SQL Injection
Injection attacks

Open Web Application Security Project (OWASP) Top vulnerability:

• Untrusted data is sent to an interpreter as part of a command or query.

• Attacker’s data can trick the interpreter into executing unintended commands or accessing data without proper authorization

Common embodiment: SQL Injection

https://www.owasp.org/index.php/Top_10_2013-Top_10
CardSystems Solutions

• Now defunct credit card processing company
• Experienced SQL injection attack in June 2005
• Over 40 million accounts compromised
• Over 200,000 credit card numbers stolen
• Data was not adequately protected

Consequence: company terminated operations and was sold to Pay by Touch (also defunct)

https://www.schneier.com/blog/archives/2005/06/cardsystems_exp.html
Persistent threat

Wordpress : Security Vulnerabilities (SQL Injection)

<table>
<thead>
<tr>
<th>#</th>
<th>CVE ID</th>
<th>CWE ID</th>
<th># of Exploits</th>
<th>Vulnerability Type(s)</th>
<th>Publish Date</th>
<th>Update Date</th>
<th>Score</th>
<th>Gained Access Level</th>
<th>Access</th>
<th>Complexity</th>
<th>Authentication</th>
<th>Conf.</th>
<th>Integ.</th>
<th>Avail.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CVE-2015-2213</td>
<td>89</td>
<td></td>
<td>Exec Code Sql</td>
<td>2015-11-09</td>
<td>2015-11-09</td>
<td>7.5</td>
<td>None</td>
<td>Remote</td>
<td>Low</td>
<td>Not required</td>
<td>Partial</td>
<td>Partial</td>
<td>Partial</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CVE-2015-3530</td>
<td>89</td>
<td></td>
<td>Exec Code Sql</td>
<td>2012-10-09</td>
<td>2012-10-10</td>
<td>6.0</td>
<td>None</td>
<td>Remote</td>
<td>Medium</td>
<td>Single system</td>
<td>Partial</td>
<td>Partial</td>
<td>Partial</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>CVE-2015-5216</td>
<td>89</td>
<td></td>
<td>Exec Code Sql</td>
<td>2012-10-25</td>
<td>2012-10-26</td>
<td>7.5</td>
<td>None</td>
<td>Remote</td>
<td>Low</td>
<td>Not required</td>
<td>Partial</td>
<td>Partial</td>
<td>Partial</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>CVE-2011-4899</td>
<td></td>
<td></td>
<td>Exec Code Sql XSS</td>
<td>2012-01-30</td>
<td>2012-01-31</td>
<td>7.5</td>
<td>None</td>
<td>Remote</td>
<td>Low</td>
<td>Not required</td>
<td>Partial</td>
<td>Partial</td>
<td>Partial</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>CVE-2011-4669</td>
<td>89</td>
<td></td>
<td>Exec Code Sql</td>
<td>2011-12-02</td>
<td>2012-03-08</td>
<td>7.5</td>
<td>None</td>
<td>Remote</td>
<td>Low</td>
<td>Not required</td>
<td>Partial</td>
<td>Partial</td>
<td>Partial</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>CVE-2011-3130</td>
<td>89</td>
<td></td>
<td>Sql</td>
<td>2011-08-10</td>
<td>2012-06-28</td>
<td>7.5</td>
<td>User</td>
<td>Remote</td>
<td>Low</td>
<td>Not required</td>
<td>Partial</td>
<td>Partial</td>
<td>Partial</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SQL injection vulnerability in the wp_untrash_post_comments function in wp-includes/post.php in WordPress before 4.2.4 allows remote attackers to execute arbitrary SQL commands via a comment that is mishandled after retrieval from the trash.

SQL injection vulnerability in the Pay With Tweet plugin before 1.2 for WordPress allows remote authenticated users with certain permissions to execute arbitrary SQL commands via the id parameter in a paywithtwitter shortcode.

SQL injection vulnerability in ajax.php in SCORM Cloud For WordPress plugin before 1.0.7 for WordPress allows remote attackers to execute arbitrary SQL commands via the active parameter. NOTE: some of these details are obtained from third party information.

** DISPUTED ** wp-admin/setup-config.php in the installation component in WordPress 3.3.1 and earlier does not ensure that the specified MySQL database service is appropriate, which allows remote attackers to configure an arbitrary database via the dhost and dbname parameters, and subsequently conduct static code injection and cross-site scripting (XSS) attacks via (1) an HTTP request or (2) a MySQL query. NOTE: the vendor disputes the significance of this issue; however, remote code execution makes the issue important in many realistic environments.

SQL injection vulnerability in wp-users.php in WordPress Users plugin 1.3 and possibly earlier for WordPress allows remote attackers to execute arbitrary SQL commands via the uid parameter to index.php.

wp-includes/taxonomy.php in WordPress 3.1 before 3.1.3 and 3.2 before Beta 2 has unknown impact and attack vectors related to "Taxonomy query hardening," possibly involving SQL injection.

http://www.cvedetails.com/vulnerability-list/vendor_id-2337/opsqli-1/Wordpress.html
Use SQL to change the meaning of a DB command

Consider this php code:

```
$recipient = $_POST['recipient'];
$sql = "SELECT PersonID FROM Person WHERE Username='$recipient';
$rs = $db->executeQuery($sql);
```

What can an attacker do if they have control over $recipient?