登录

Encryption 101系列：勒索软件ShiOne案例分析

mss**** / 2018-03-12 22:14:28 / 浏览数 11936 技术文章 技术文章 顶(1) 踩(0)

原文：https://blog.malwarebytes.com/threat-analysis/2018/02/encryption-101-shione-ransomware-case-study/

0x00 简介

在本系列的第一篇文章中，我们为读者介绍了与恶意软件有关的一些加密基础概念。如果您还没有读过的话，我们强烈建议先阅读上一篇文章，因为它是理解下面介绍的案例

本文旨在通过一个实例来帮助读者深入理解勒索软件的加密过程。这里的示例软件为ShiOne，之所以选它，并不是因为它有什么过人之处或采用了什么新颖的技术，恰恰相

0x01 勒索软件常用的加密方式

在前面的文章中，我们曾说过勒索软件采用的加密方式有很多，例如：

- 加密密钥是在受害计算机上本地生成并发送到C2服务器的。
- 密钥是预先分发并嵌入勒索软件中的。

对于本文所用的例子来说，它采用的是第二种加密方式。也就是说，加密密钥是脱机生成的，并在发动攻击之前就已经嵌入到恶意软件中了。

当谈论加密密钥是离线生成并提前嵌入的时候，意味着我们讨论的是非对称密钥。接下来您将看到，在这个勒索软件中采用的是RSA和AES加密算法。

AES是一种对称加密算法，这意味着加密和解密过程使用的是同一个密钥，或者，也可以将该密钥视为加密和解密的密码。AES的密钥实际上是动态生成的，并存储在文件本

RSA是该恶意软件中的非对称加密组件——完成相应的预生成和嵌入操作。RSA的作用是隐藏用于加密所有文件的AES密钥（密码）。对于RSA的两个密钥来说，只有公钥位

0x02 加密主函数

我们分析的样本是：

408a97ac8fb82398187ffe6d4c5937d7

在初始化和枚举驱动器中的文件的代码（因为这些代码都是相当标准的，所以无需进行讲解）之后，就是调用了crtp(string path)的加密主函数。

这个函数是从主目录枚举循环中进行调用的，也就是说，每找到一个文件的时候，都会调用一次这个主函数。同时，它也是这个程序中唯一会调用加密API或随机数生成器的

对于上面的代码，有必要进行一些解释。如前所述，主加密密钥是离线生成的，并被作者嵌入到了恶意软件中。实际上，如果这是一种在本地生成主密钥的勒索软件的话，那

下面，让我们来仔细看看crtp函数：

加密过程是从调用string text = Program.CreateSalt(32);这一句开始的。

这是一个用户定义的函数，它的功能很简单，就是调用一个标准的加密API来生成一个由32个随机数组成的数组。尽管该函数功能简单，却非常重要，因为这里的随机字节数

CreateSalt函数的详细代码

```
//FUNCTION PROGRAM.CREATESALT() {
RNGCryptoServiceProvider rNGCryptoServiceProvider = new RNGCryptoServiceProvider();
byte[] array = new byte[size];
rNGCryptoServiceProvider.GetBytes(array);
return Convert.ToBase64String(array);
```

这实际上是整个恶意软件中生成的最重要的数字，因为这就是受害者需要购买的密钥，有了它，受害者就可以解密所有的文件了。

0x03 随机数生成器

在继续阅读下文之前，需要先了解RNG（随机数生成器）的一些细节。

RNGCryptoServiceProvider是符合随机数生成器安全标准的默认实现，是一种更强大的密码随机数生成器。而System.Random则是一个安全性较弱的替代品，因为其运算

根据我们的分析结果，这个样本的确使用了强大的RNG随机数生成器，所以请耐心阅读下文。

现在，我们将跳过下一部分代码，因为它只是简单地过滤勒索软件想要加密的文件类型。此外，这些代码本身也很容易理解，而且在实际的加密过程中，它们对加密本身也没

现在，我们考察下一个代码段，与前面的代码类似，它使用的也是RNG内部函数，并创建了另一个随机数组：

byte[] array = Program.GenerateRandomSalt();

这里没有给出GenerateRandomSalt()函数的细节，因为它与前面的代码没有多大区别。这个数组将用作文件加密主函数的salt，这一点将在后面进行解释。

接下来，勒索软件会读取在数组开始部分生成的第一个数字（变量text），并调用一个函数对其进行加密以进行存储：

string s = Program.Encryption(text)

如上所述，这组数字实际上将用作未来的AES密钥。很明显，变量text的值将作为明文形式的密钥使用，但是随后会对其进行加密，并被附加到文件的末尾，以便保存，具体

0x04 加密功能

以下是勒索软件攻击过程中加密代码的工作原理。勒索软件：

- 生成RSA公钥和私钥（本地或离线方式预先生成）。
- 生成一个随机数作为生成AES密钥的函数的输入（密码）。
- 使用新生成的AES密钥加密文件。
- 使用RSA公钥加密AES密钥。
- 在加密文件中追加加密以后的AES密钥。

以下是攻击者在收到赎金后解密文件的过程：

- 找到加密的文件。
- 在文件中搜索经过加密的AES密码。
- 将加密的密码读入内存，用作生成AES密钥的种子。
- 使用RSA私钥解密AES密码。
- 使用这个明文密码作为AES算法的输入对加密的文件进行解码。

Encryption(text)函数的具体代码如下所示。
[突出显示的行就是勒索软件用于提取嵌在可执行文件中的公钥的代码。]

勒索软件首先获取嵌在可执行文件自身中的公钥，然后使用RSA 2048算法对传入的随机数（变量text的值，即AES密钥密码）进行加密。

注意：这个函数与实际的文件加密毫无关系，它只是供勒索软件用来隐藏AES密钥，防止被用户发现。实际上，这个函数可以直接跳过，因为它对文件的实际加密没有一点影

重点在于，公钥已经嵌入在恶意软件中了。这意味着RSA密钥不是动态生成的，也就是说，RSA公钥和私钥对是在恶意软件服务器端生成的，因此只有恶意软件作者自己才能

0x05 进行加密

现在，勒索软件已经获得了对当前文件执行单独加密所需的一切。就本例来说，使用的加密算法是基于CBC（密码分组链接）模式的AES算法。这就意味着，为了对原始文件

```
for (i = 0; i &amp;amp;amp;lt; num; i += array3.Length)
{
int len = fileStream.Read(array2, 0, array2.Length);
array3 = Program.AES_(array2, bytes, array, len);
fileStream.Seek((long)i, SeekOrigin.Begin);
fileStream.Write(array3, 0, array3.Length);
}
```

其中，array3 = Program.AES_(array2, bytes, array, len)就是实际执行文件加密的代码。

在传递给AES_函数的参数中，array2是原始文件数据，bytes是在前面使用Program.CreateSalt(32)生成的作为“密码”的随机数。而array则是用Program.GenerateRandom

如您所见，实际加密文件的方法是AES。该算法使用salt来进一步随机化传入的“passwordBytes”（AES密钥）。这是为了符合AES的安全标准。AES
CBC模式需要使用前面提到的初始化向量，具体可以在代码中找到它。generateIV()函数只是另一个随机数生成器。后面，它会对原始文件数据逐块执行AES加密处理。每个

现在，我们来考察crypt函数的最后一步，在文件所有内容都进行完加密之后，勒索软件就会将salt值写入文件的起始位置。然后，写入RSA算法加密的AES密码，最后写入实

当解密软件运行时，它将遍历每个文件并读取salt值。然后，使用RSA私钥解密AES密钥，该私钥嵌在解密软件自身当中。最后，它会读取加密的文件数据和IV。这样，一切

0x06 总结

在这个案例研究中，我们详细演示了一个标准的文件加密程序的各种功能。在本系列的下一篇文章中，我们不仅会向读者介绍加密算法的各种弱点，同时还会分析样本勒索软

- 动动手指，沙发就是你的了！

登录 后跟帖

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板