2016 年 12 月 7 日，星期三

Floki Bot 攻击肆虐，Talos 和 Flashpoint 携手应对

作者：Ben Baker、Edmund Brumaghin、Mariano Graziano 和 Jonas Zaddach

执行摘要

Floki Bot 是近期各种暗网市场上出售的一种新型恶意软件变体。Floki Bot 所使用的代码库源自臭名昭著的 Zeus 木马（该木马的源码于 2011 年泄露）。但 Floki Bot 并非简单地复制 Zeus 木马“原有”的功能，而是声称拥有一些新功能，因此成为一种对犯罪分子极具吸引力的工具。由于 Talos 一直致力于全面监控威胁形势的变化，以确保紧跟威胁的演变持续保护我们的客户，因此我们对该恶意软件变体进行了深入了解，以确定 Floki Bot 的技术能力和特性。

在分析 Floki Bot 的过程中，Talos 发现其制作者对已泄露的 Zeus 源代码中的 dropper 机制进行了修改，试图使 Floki Bot 更加难以检测。Talos 还观察到，Floki Bot 引入了新的代码以便使用 Tor 网络。但此功能目前似乎没有被使用的迹象。最后，在分析过程中通过使用 FIRST 框架，Talos 很快发现 Floki Bot 重复使用了 Zeus 的代码/函数。这一发现使我们提高了样本分析的效率，为我们记录所分析的 Floki Bot 样本中存在的各种功能节省了时间。

在分析 Floki Bot 的过程中，Talos 与 Flashpoint 开展了积极的合作。这确保 Talos 和 Flashpoint 能够快速交流与正在进行的 Floki Bot 散布活动相关的情报，以及有关恶意软件中存在的技术功能的数据。此外，Talos 正在为开源社区提供脚本，以便帮助恶意软件分析人员实现部分 Floki Bot 分析过程的自动化，并简化 Floki Bot 的分析过程。

FLOKI BOT 详情

Floki Bot 的感染过程包含多个步骤。概括来讲，此过程如下图所示：

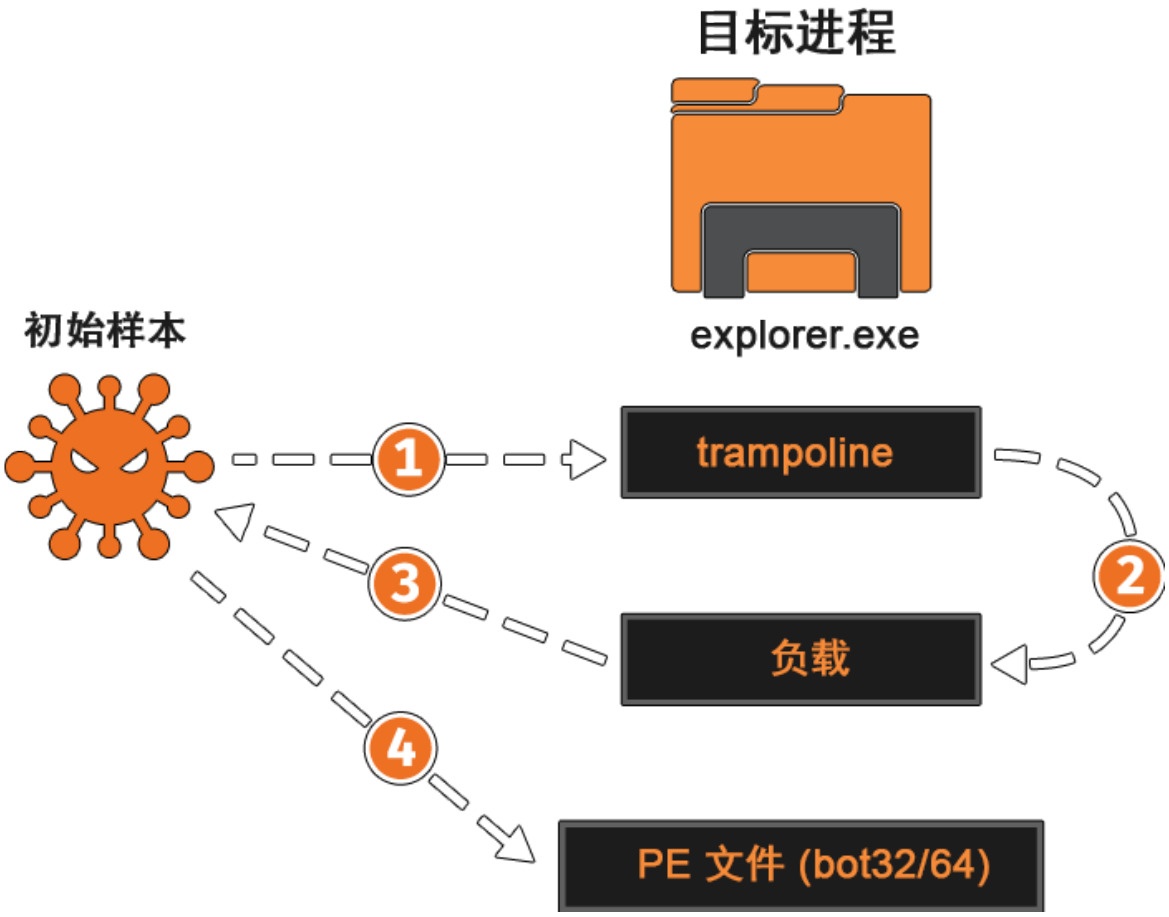

图 1：恶意软件代码注入顺序

首先，我们使用下面的二进制文件对 Floki Bot 进行了分析：

5e1967db286d886b87d1ec655559b9af694fc6e002fea3a6c7fd3c6b0b49ea6e (SHA256)

恶意软件开始执行后，便试图将恶意代码注入到 Microsoft Windows 文件管理器“explorer.exe”中。如果无法打开“explorer.exe”，则注入到“svchost.exe”中。它最初只会注入一个 trampoline（图 1 中的步骤 1）。此 trampoline 执行两个不同的调用。第一个调用是一个 100 毫秒的“Sleep()”。第二个调用将控制传递给另一个负载函数。该函数的参数是一个结构，其中包含初始样本的进程 ID、用于进一步二进制负载的解密密钥，以及初始样本地址空间中负载资源的指针和大小。奇怪的是，尽管初始样本具有标记为“bot32”和

“bot64”的资源，但我们分析的样本是硬编码的，只将“bot32”资源的地址传递到注入的负载。负责映射“bot32”、“bot64”和“密钥”资源的反向代码如图 2 所示。

```
BOOL __userpurge sub_4025C7@<eax>(_DWORD *a1@<edi>, HMODULE hModule)
{
  HRSRC v2; // eax@1
  int v3; // eax@2
  HRSRC v4; // eax@7
  int v5; // eax@8
  HRSRC v6; // eax@10
  unsigned int v7; // eax@11
  int v9; // [esp+4h] [ebp-4h]@1

  v9 = 0;
  v2 = FindResourceW(hModule, L"key", (LPCWSTR)0xA);
  if ( v2 )
    v3 = sub_40209B(hModule, (int)&v9, v2);
  else
    v3 = 0;
  if ( v3 && v9 )
  {
    sub_401831(a1 + 21, v9, 16);
    sub_401811();
  }
  v4 = FindResourceW(hModule, L"bot32", (LPCWSTR)0xA);
  if ( v4 )
    v5 = sub_40209B(hModule, (int)(a1 + 1), v4);
  else
    v5 = 0;
  a1[3] = v5;
  v6 = FindResourceW(hModule, L"bot64", (LPCWSTR)0xA);
  if ( v6 )
    v7 = sub_40209B(hModule, (int)(a1 + 2), v6);
  else
    v7 = 0;
  a1[4] = v7;
  return a1[1] && a1[3] > 0u && a1[2] && v7 > 0;
}
```

图 2：“bot32”、“bot64”和“密钥”资源的映射

从以下截图中可以看出，图 3 显示的是负责为注入准备 shellcode 的代码。此操作在初始二进制文件中执行。图 4 显示的是注入“explorer.exe”进程的结果。我们可以清楚地观察到反汇编是基于以前的 shellcode 并且包含上述两个调用。具体来说，0xA001F 处的调用是调用负载，即图 1 中的步骤 2。

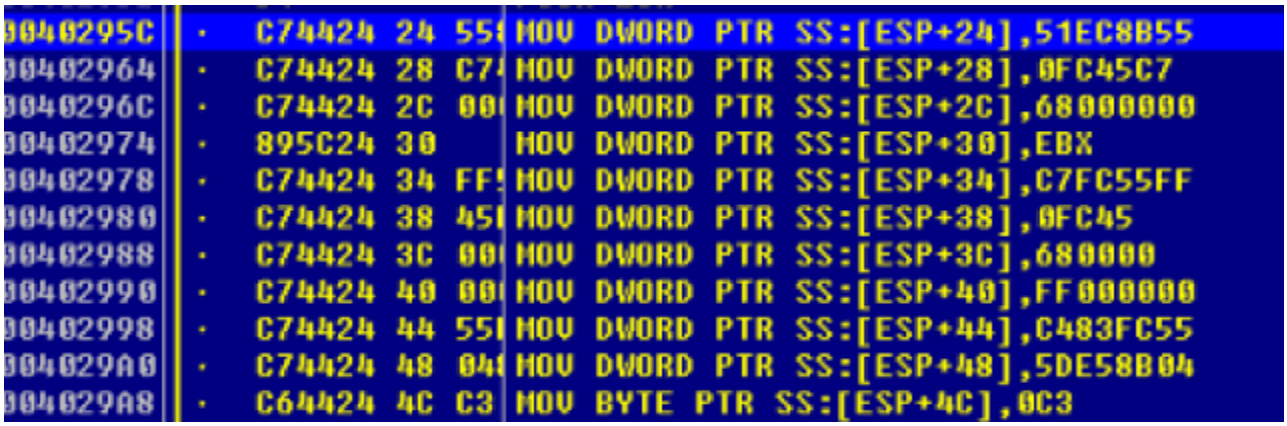

图 3：Shellcode 准备

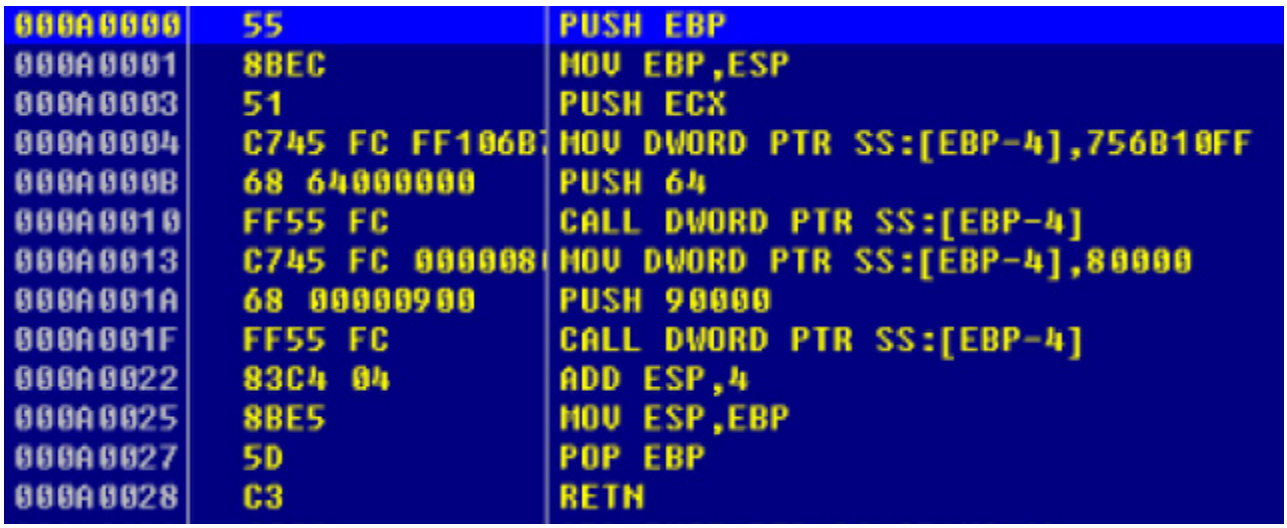

图 4：注入 Shellcode 的反汇编

下一个逻辑步骤是再次在“explorer.exe”地址空间进行注入。这一次，在 trampoline 之后执行的负载通过使用 CRC 查找解析所需的 API，然后映射初始二进制文件中的“bot32”资源部分。

该资源使用 RC4 进行了加密，并且可以使用“密钥”资源中的 16 字节密钥数据进行解密，该密钥数据作为参数传递到注入代码。此外，该资源还通过 LZNT1 算法进行了压缩，并通过调用 RtlDecompressBuffer 进行解压缩。Talos 已创建（即将发布）一个名为“PayloadDump”的脚本，用于解压缩这些 bot 负载。此 bot 是最后一个组件，并且包含银行木马功能。许多 AV 引擎将其标记为典型 Zeus bot。系统会加载该 bot 并将其注入“explorer.exe”。这些步骤在图 1 中标记为 3 和 4。

在每个阶段，恶意软件都使用散列处理来模糊动态库解析中使用的模块和函数名称。有趣的是，初始样本和 bot (bot32) 可执行文件使用相同的 CRC32 实现，并使用静态密钥对结果进行 XOR 运算（在本例中为 0x5E58），而负载使用相同的 CRC32 实现，但使用不同的 XOR 密钥（在本例中为 0x3086）。系统在计算之前将模块的名称转换为小写形式（Windows 文件名通常不区分大小写）。

目前，Virustotal 上的 30 多个 AV 引擎可以立即识别出“bot32”资源，其中大多数检测将其识别为 Zbot，而只有 10 个 AV 引擎可以将“bot64”资源检测为恶意资源。在我们的分析过程中，我们从 explorer.exe 进程的物理内存转储（请参阅“内存分析”部分）以及初始二进制文件的资源部分中提取了样本。乍一看，该样本似乎是一个普通的 Zeus bot。但它的主要不同之处在于支持 Tor 网络，当恶意软件配置中指定的 C2 域以“.onion”结尾时（“.onion”是 Tor 相关域的伪 TLD），Tor 网络便会被激活。在这种情况下，该样本将配置一个标准 Tor 代理服务器来监听 localhost:9050，如以下截图所示：

```
if ( q_StrStr(v6, ".onion") )
  v3 = sub_40B4CB((int)".onion", v6);
else

 v16 = 0;
 v2 = sub_40B32D(9050, "127.0.0.1");
 if ( v2 != -1 )
 {
```

图 5：Floki Bot Tor 功能

此功能似乎尚在开发中，无法在 Talos 所分析的样本中激活。

FLOKI BOT 的 DROPPER/加载程序

Floki Bot 使用的加载程序未加密，也未使用任何反调试技术。该加载程序会隐藏用于将恶意负载注入到其他进程中的系统调用。此处详细记录了 Floki Bot 加载程序使用的注入技术，因此我们将不对此进程的工作原理进行深入探讨。

网络分析

Floki Bot 通过 HTTPS 连接与 C2 通信。有趣的是，该恶意软件的制作者宣称其具备防深度数据包检测功能。为实现此功能，制作者将网络数据包中的字节封装在通过 HTTPS 发送的 BinStorage 结构中。BinStorage 结构中的每个字节都用前一个字节进行 XOR 运算，然后使用 RC4 进行额外加密。其实，此功能早就存在于已泄漏的 Zeus 源码中，并不是 Floki Bot 的新功能。在断开 HTTPS 连接并解密数据包负载后，我们注意到该恶意软件发回了有关受

感染计算机的信息（如计算机名称和屏幕分辨率）。Floki Bot 声称它“与 Zeus 不同，不会被深度数据包检测功能检测出来”。但它对已泄漏的 Zeus 源码的唯一重要更改是支持 Tor，而目前尚未发现任何现有恶意软件样本使用 Tor 支持。因为该恶意软件不对其通信使用证书锁定，所以 Talos 能在使用 mitmproxy 拦截 Floki Bot 网络数据包后对其进行解密。

内存分析

在我们的分析中，我们还在使用 Floki Bot 感染虚拟机后进行了基于内存的调查分析。这是为了使用逆向方法，从最终结果开始我们的分析，然后尝试重建感染过程的不同步骤。首先，我们用 win32dd 提取了物理内存转储，并用 Volatility（著名的开源内存调查分析框架）对其进行分析。首先，我们使用了“pslist”Volatility 插件。此插件通过运行连接所有 _EPROCESS 对象的双链表列出所有进程。输出中未显示任何异常。然后我们使用“netscan”插件显示“explorer.exe”进程的网络活动，结果发现了源于文件管理的网络流量，这是绝对不正常的，因此需要进行进一步调查。为此，我们在“explorer.exe”进程上运行了“malfind”，结果发现了一些有趣的跟踪信息和注入到该进程的 PE 文件。我们转储了这些项目，它们与逆向过程的部分结果相匹配。我们可以观察 trampoline、负载和 PE 文件。关于 Floki Bot 采用的持续攻击机制，我们通过“filescan”识别了一些项目，并且观察到还有二进制文件（具有随机名称）复制到启动文件夹中。

与 FLASHPOINT 协作

在调查 Floki Bot 恶意软件期间，我们与 Flashpoint 携手合作，共同收集情报信息并共享技术详细信息，其中涉及恶意软件样本、目前使用 Floki Bot 的攻击活动，以及买卖 Floki Bot 的暗网市场。Flashpoint 一直在跟踪多个 Floki Bot 发起者和攻击活动。Flashpoint 还发布了一篇博文，介绍与当前在全球活跃的 Floki Bot 攻击活动相关的情报。如要阅读该 Flashpoint 博文，请点击此处。

使用 FIRST 分析 FLOKI BOT

在分析过程中，Talos 利用功能识别和恢复签名工具 (FIRST)，以及相关的 IDA Python 插件来收集和记录所分析的 Floki Bot 样本中的功能。FIRST 是 Talos 最近发布的一款开源框架，恶意软件分析人员与研究人员可以通过此框架开展协作，并共享与恶意软件样本中存在的恶意功能相关的分析数据。

使用 FIRST 可以快速高效地分析恶意软件，因为它可以最大限度地减少对已分析和已记录的恶意代码进行分析所需的时间。FIRST 目前包含 17 万多个功能的信息，其中包括：Zlib 和 OpenSSL 等常用库、已泄露的恶意软件源码，以及社区分析的恶意 Windows 和 Linux 文件。在分析具有数千个库函数的静态链接的可执行文件时，此框架特别有用。虽然 Bindiff 等工具

非常有用，但它们只能用于比较少量的文件，而且您必须找到并获取这些文件，才能进行比较。FIRST 插件可以自动查找社区提交的数千个文件中每个文件的函数相似性。

IDA Pro 尝试使用 FLIRT 签名来识别常用库函数，但未能识别此 Floki Bot 样本中的任何函数。IDA Pro 对未识别函数的默认响应是根据起始地址对该函数进行命名。在本例中，我们获取了 516 个通用名称类似于“sub_402F34”的函数。

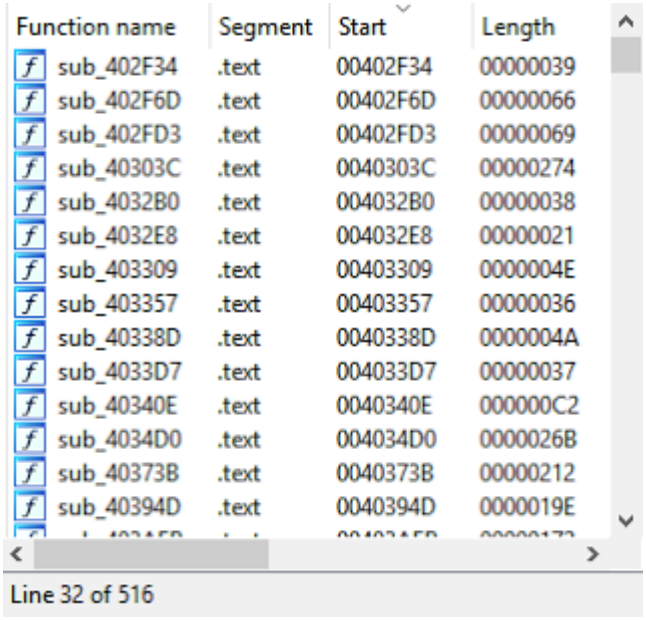

图 6：运行 FIRST 之前的 IDA Pro 函数列表

我们查询了 FIRST，并在几秒钟内获取了 128 个标记有有意义的名称、原型和注释的函数。现在，我们可以立即查看这些函数执行什么操作以及它们采用什么参数，即使这些参数是自定义结构。

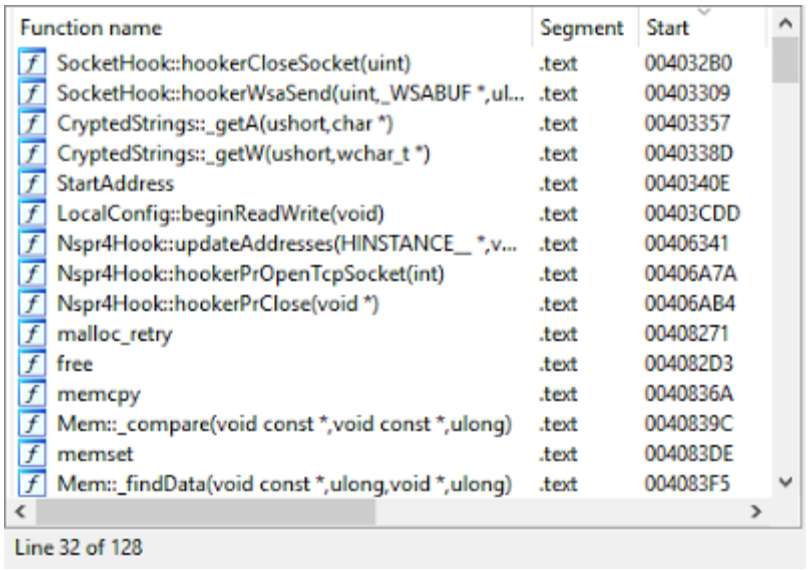

图 7：运行 FIRST 之后的 IDA Pro 函数列表

如果不先分析函数的子函数，许多函数很难分类。分析人员通常使用深度优先方法来标记具有明显行为的函数，然后在对嵌套函数有更好的了解后再回溯到父函数。

```
00413249 sub_413249 proc near
00413249
00413249 var_538= byte ptr -538h
00413249 var_4E1= byte ptr -4E1h
00413249 var_104= byte ptr -104h
00413249 arg_0= dword ptr  8
00413249
00413249 push    ebp
0041324A mov     ebp, esp
0041324C sub     esp, 538h
00413252 lea     eax, [ebp+var_538]
00413258 call    sub_41321C
0041325D push    102h
00413262 lea     eax, [ebp+var_4E1]
00413268 push    eax
00413269 lea     eax, [ebp+var_104]
0041326F push    eax
00413270 call    sub_40836A
00413275 mov     eax, 1E6h
0041327A push    eax
0041327B push    offset unk_41E2C4
00413280 push    [ebp+arg_0]
00413283 call    sub_40836A
00413288 push    eax
00413289 push    [ebp+arg_0]
0041328C lea     eax, [ebp+var_104]
00413292 call    sub_409899
00413297 leave
00413298 retn    4
00413298 sub_413249 endp
```

图 8：IDA Pro 显示对未使用 FIRST 的未知函数的调用

在本示例中，FIRST 识别了所有函数，并使用它们的参数名称和类型对它们进行了标记。这样，这些函数便具有相应的备注，表明它们来自泄漏的 Zeus 源代码，这为我们到何处查找更多有关未识别函数的信息提供了重要线索。有些 FIRST 未能识别的函数与 Zeus 源中的函数相似，但经过了更改（源码或编译器选项已被修改）。

```
00413249 void __fastcall Core::getPeSettings(struct PESETTINGS *) proc near
00413249
00413249 var_538= byte ptr -538h
00413249 from_buffer= byte ptr -4E1h
00413249 to_buffer= byte ptr -104h
00413249 arg_0= dword ptr  8
00413249
00413249 push    ebp
0041324A mov     ebp, esp
0041324C sub     esp, 538h
00413252 lea     eax, [ebp+var_538]
00413258 call    Core::getBaseConfig(BASECONFIG *) ; Leaked Source - Zeus Client
0041325D push    102h            ; size
00413262 lea     eax, [ebp+from_buffer]
00413268 push    eax             ; from_buffer
00413269 lea     eax, [ebp+to_buffer]
0041326F push    eax             ; to_buffer
00413270 call    Mem::_copy(void *,void const *,ulong)
00413275 mov     eax, 1E6h
0041327A push    eax             ; size
0041327B push    offset unk_41E2C4 ; from_buffer
00413280 push    [ebp+arg_0]     ; to_buffer
00413283 call    Mem::_copy(void *,void const *,ulong)
00413288 push    eax
00413289 push    [ebp+arg_0]
0041328C lea     eax, [ebp+to_buffer]
00413292 call    Crypt::_rc4(void *,ulong,Crypt::RC4KEY *) ; Leaked Source - Zeus Client
00413297 leave
00413298 retn    4
00413298 void __fastcall Core::getPeSettings(struct PESETTINGS *) endp
```

图 9：FIRST 标记的相同函数

如果将 Floki Bot 可执行文件与 Zeus 进行比较，您会看到 BASECONFIG 结构的大小不同，并且全局变量的偏移量也发生了变化。尽管这些参数已经过修改，但 FIRST 的一个引擎还是识别出这些函数。得益于 FIRST，我们才能够快速找到负责此函数的泄漏源码数据块。

```
void Core::getPeSettings(PESETTINGS *ps)
{
  BASECONFIG baseConfig;
  getBaseConfig(&baseConfig);

  Crypt::RC4KEY rc4k;
  Mem::_copy(&rc4k, &baseConfig.baseKey, sizeof(Crypt::RC4KEY));
  Mem::_copy(ps, &coreData.peSettings, sizeof(PESETTINGS));
  Crypt::_rc4(ps, sizeof(PESETTINGS), &rc4k);
}
```

图 10：泄露的函数源码

Talos 在分析 Floki Bot 样本时创建的所有分析数据和函数文档都已通过公共 Talos FIRST 服务器（测试版）公布。有关 FIRST 框架及其用途的更多信息，请点击此处。

工具版本

在分析过程中，Talos 还创建了脚本来帮助自动执行部分 Floki Bot 分析，该脚本现已向开源社区发布。分析人员可以使用这些脚本转储 Floki Bot 样本使用的配置参数，以及 Floki Bot 负载本身。

PayloadDump - 从初始 Floki Bot 样本中提取 PE32 格式的最终负载。

ConfigDump - 支持提取样本使用的 Floki Bot 配置参数。

可以从此处的 Github 下载这些脚本。

总结

Floki Bot 再次证实了成功的恶意软件套件源代码在线泄漏后会产生什么恶果。正如我们自 Zeus 源代码泄露以来多次看到的那样，基于此代码库的新恶意软件变体不断出现。Floki Bot 的独特之处在于此恶意软件的制作者努力扩展了 Zeus 中存在的功能，并实施了新的功能，使 Floki Bot 对犯罪分子极具吸引力。

由于目前多个暗网市场上 Floki Bot 交易活跃，因此随着网络犯罪分子继续试图利用它攻击系统来获利，人们可能会继续看到 Floki Bot 兴风作浪。由于 Zeus 源代码的泄漏在整个威胁环境中继续具有连锁反应，Talos 将继续监控此威胁以及攻击者大肆利用的其他威胁，以确保客户在出现新威胁或现有威胁随着时间的推移发生变化时继续得到保护。

防护

思科客户可通过其他方式检测并阻止此威胁，包括：

产品	保护
AMP	✔
CWS	✔
邮件安全	不适用
网络安全	✔
WSA	✔

高级恶意软件防护 (AMP) 解决方案可以有效防止执行威胁发起者使用的恶意软件。

CWS 或 WSA 的网络扫描功能可以阻止访问恶意网站，并检测这些攻击中所用的恶意软件。

IPS 和 NGFW 的网络安全保护功能拥有最新的签名库，可以检测威胁发起者的恶意网络活动

危害表现 (IOC)

1.1.a 恶意软件二进制文件：

08e132f3889ee73357b6bb38e752a749f40dd7e9fb168c6f66be3575dbbbc63d (SHA256)
5028124ce748b23e709f1540a7c58310f8481e179aff7986d5cfd693c9af94da (SHA256)
0aa1f07a2ebcdd42896d3d8fdb5e9a9fef0f4f894d2501b9cbbe4cbad673ec03 (SHA256)
5e1967db286d886b87d1ec655559b9af694fc6e002fea3a6c7fd3c6b0b49ea6e (SHA256)
d1d851326a00c1c14fc8ae77480a2150c398e4ef058c316ea32b191fd0e603c0 (SHA256)
e0b599f73d0c46a5130396f81daf5ba9f31639589035b49686bf3ef5f164f009 (SHA256)
e43ee2ab62f9dbeb6c3c43c91778308b450f5192c0abb0242bfddb8a65ab883a (SHA256)
2b832ef36978f7852be42e6585e761c3e288cfbb53aef595c7289a3aef0d3c95 (SHA256)
4bdd8bbdab3021d1d8cc23c388db83f1673bdab44288fccae932660eb11aec2a (SHA256)
3c2c753dbb62920cc00e37a7cab64fe0e16952ff731d39db26573819eb715b67 (SHA256)
7bd22e3147122eb4438f02356e8927f36866efa0cc07cc604f1bff03d76222a6 (SHA256)
9d9c0ada6891309c2e43f6bad7ffe55c724bb79a0983ea6a51bc1d5dc7dccf83 (SHA256)
e205a0f5688810599b1af8f65e8fd111e0e8fa2dc61fe979df76a0e4401c2784 (SHA256)
ac5ae89af8d2ffdda465a4038f0f24fcbcb650140741c2b48adadc252a140e54 (SHA256)

命令和控制 URL：

https[:]//193.201.225[.]30/sweetdream/gxve8xj4a7t8t8sug8s57.php
https[:]//shhtunnel[.]at/class/gate.php
https[:]//extensivee[.]bid/000L7bo11Nq36ou9cfjfb0rDZ17E7ULo_4agents/gate.php
https[:]//5.154.190[.]248/gate.php
https[:]//vtraffic[.]su/gate.php
https[:]//springlovee[.]at/adm/config.bin
https[:]//feed.networksupdates[.]com/feed/webfeed.xml
https[:]//wowsupplier[.]ga/cpflkabwbebeu/gtlejbsbu.php
https[:]//adultgirlmail[.]com/mail/gate.php
https[:]//uspal[.]cf/3faf5c96-9c2b-11e6-95d4-00163c75bf83/gate.php

发布者：EDMUND BRUMAGHIN；发布时间：11:02

标签：银行木马、FLOKI BOT、恶意软件、逆向工程