利用沙盒（脚本虚拟机）检测加密及未知WebShell

吴康(Polymorphours)

概述

- WebShell常见检测技术及问题
- 利用脚本虚拟机来检测WebShell(以ASP为例)
- 脚本虚拟机在实战环境中的挑战

WebShell常见检测技术及问题

- WebShell的常见检测方式
  - 基于流量中payload特征检测
  - 基于文件的webshell特征检测
  - 基于日志的特征检测

WebShell常见检测技术及问题

- WebShell的常见检测方法
  - 基于WebShell字符特征
  - 基于WebShell行为特征
  - 基于统计学特征检测

WebShell常见检测技术及问题

- 基于WebShell字符特征
  - 正则表达式匹配
  - 基于解释器前端的降噪技术
- 问题
  - WebShell代码加密、变形，正则表达式难以匹配
  - 虽然匹配更加的精确，但依然无法解决代码加密的问题

WebShell常见检测技术及问题

- 基于WebShell行为特征
  - 利用HOOK技术拦截关键函数
- 问题
  - 单纯拦截函数误报率高
  - WebShell无活动时无法查杀
  - 输入内容简单变形无法查杀

WebShell常见检测技术及问题

- 基于统计学特征检测
  - 文本文件信息熵
  - 文件的使用频度
  - 重合指数IC
  - 压缩比
  - 最大字符串长度等
- 问题
  - 纯文本信息熵易受干扰
  - 对于混合进正常文件的代码较难检测
  - 缺少行为特征向量

利用脚本虚拟机检测WebShell

- 脚本虚拟机的优势
  - 不依赖文本特征检测
  - 大部分自加密的脚本可被检测
  - 可检测未活动的WebShell

利用脚本虚拟机检测WebShell

- 脚本虚拟机的构成
  - 脚本语言的解释器
  - 脚本语言的执行环境

利用脚本虚拟机检测WebShell

- 构造脚本解释器
  - 解释器结构
    - 词法分析器
    - 语法分析器
    - AST执行器

利用脚本虚拟机检测WebShell

- 构造脚本解释器（以ASP为例）
  - ASP的脚本引擎
    - VBScript
    - JScript
    - PerlScript

利用脚本虚拟机检测WebShell

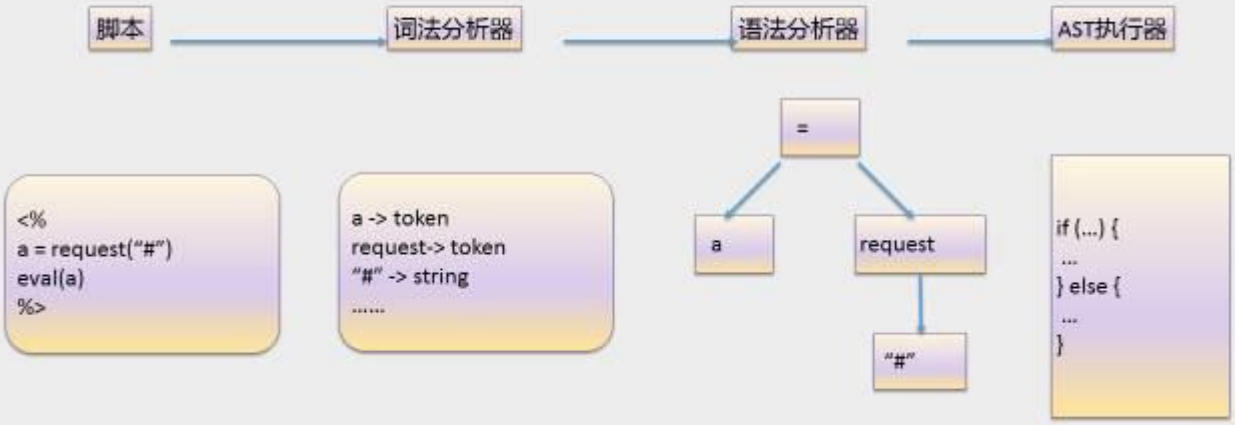
- VBScript解释器构造
  - 词法分析器
    - 词法分析前的准备工作
      - 提取脚本编码方式
      - 解密vbscript.encode编码
      - 合并文件，解决include的问题
    - 进行词法分析
      - 过滤HTML部分，提取真实的脚本
      - 给出脚本各个单词的解释

利用脚本虚拟机检测WebShell

- VBScript解释器构造
  - 语法分析器
    - bison工具构造(LALR算法)
    - VBScript文法二义性问题
      - 不可忽略的换行符
        - if ... then ... end if和if ... then ...
      - 必须配合符号表
        - a = b(c(d(x)))

利用脚本虚拟机检测WebShell

- 解释器图例



利用脚本虚拟机检测WebShell

- 构造脚本执行环境
  - ASP内置函数
    - 大约96个
  - ASP内置对象
    - request response session 等等
  - COM组件

利用脚本虚拟机检测WebShell

- 利用虚拟机污点跟踪方法
  - 标记数据输入入口
    - request session等等
  - 标记危险函数
    - eval类函数 fso wscript.shell等组件

利用脚本虚拟机检测WebShell

- 利用虚拟机污点跟踪方法
  - 为数据入口变量设置污染标签
  - AST执行器执行
    - 数据赋值，复制时标签一同复制
  - 污染数据进入危险函数
    - 判定检测出WebShell

利用脚本虚拟机检测WebShell

- 利用虚拟机污点跟踪方法
  - 虚拟机污点跟踪过程示例

```
<%
function MorfiCoder(Code)
MorfiCode = Replace(Replace(StrReverse(Code), "/*/", """"), "\*\", vbCrlf)
end function
execute MorfiCoder("")/*/z/*/(tseuqer lave")
%>
```

利用脚本虚拟机检测WebShell

- 利用虚拟机污点跟踪方法
  - 虚拟机污点跟踪过程示例

```
<%
function MorfiCoder(Code)
MorfiCode = Replace(Replace(StrReverse(Code), "/*/", """"), "\*\", vbCrlf)
end function
execute MorfiCoder(")/*/z
%>
```

```
PolymorphoursdeMacBook-Air:ASPAnalyzer Polymorphours$ ./aspvm -r vbscript -s -f
6.asp
install vbs parser
use Intelligence function
declare function: morficoder
ser function return: eval request("z")
```

利用脚本虚拟机检测WebShell

- 利用虚拟机污点跟踪方法
  - 虚拟机污点跟踪过程示例

```
<%
' 此时代码等同于调用
execute eval request("z")
%>
```

  - 进而调用eval request("z")
    - request("z")返回污染标记
    - 带有污染标记的临时变量进入eval，判定为webshell

脚本虚拟机在实战中的挑战

- 脚本虚拟机遭遇的挑战
  - 无法覆盖有问题的分支
  - 解密密钥存在于流量中的问题

```
<%
   a = request("x")
   b = request("y") : c = (CInt(a) + 1024) mod 99
   if c = 5 then eval b
%>
```

脚本虚拟机在实战中的挑战

- 脚本虚拟机遭遇的挑战
  - 解决代码覆盖率的魔咒
    - 利用动态符号执行技术
    - 利用SMT求解器生成所需输入向量

脚本虚拟机在实战中的挑战

- 脚本虚拟机遭遇的挑战
  - 动态符号执行
    - 获得路径的精确约束
      - 声明符号a
      - 经过CInt函数转换为整数
      - 经过表达式c = (CInt(a) + 1024) mod 99获得符号
      - c = (a + 1024) mod 99
      - 最后求的约束为(a + 1024) mod 99 = 5

脚本虚拟机在实战中的挑战

- 脚本虚拟机遭遇的挑战
  - 利用SMT求解器求解
    - Z3脚本

```
(declare-const a Int)
(assert (= (mod (+ a 1024) 99) 5))
(check-sat)
(get-value (a))
(exit)
```

脚本虚拟机在实战中的挑战

- 脚本虚拟机遭遇的挑战
  - 遍历所有路径
    - 每次运行至分支时将生成下一次输入的向量
  - 利用深度优先算法完成所有路径

脚本虚拟机在实战中的挑战

- 脚本虚拟机遭遇的新挑战
  - 更多的挑战
    - 遭遇死循环的尴尬
    - 适时结束污点跟踪
    - 使用强加密算法来加密代码

脚本虚拟机在实战中的挑战

- 脚本虚拟机遭遇的新挑战
  - 与其它技术结合的展望
    - 解释器前端降噪后的信息熵
    - 结合统计学与机器学习分类算法

椒图天择实验室

吴康（Polymorphours）

polymorphours@gmail.com