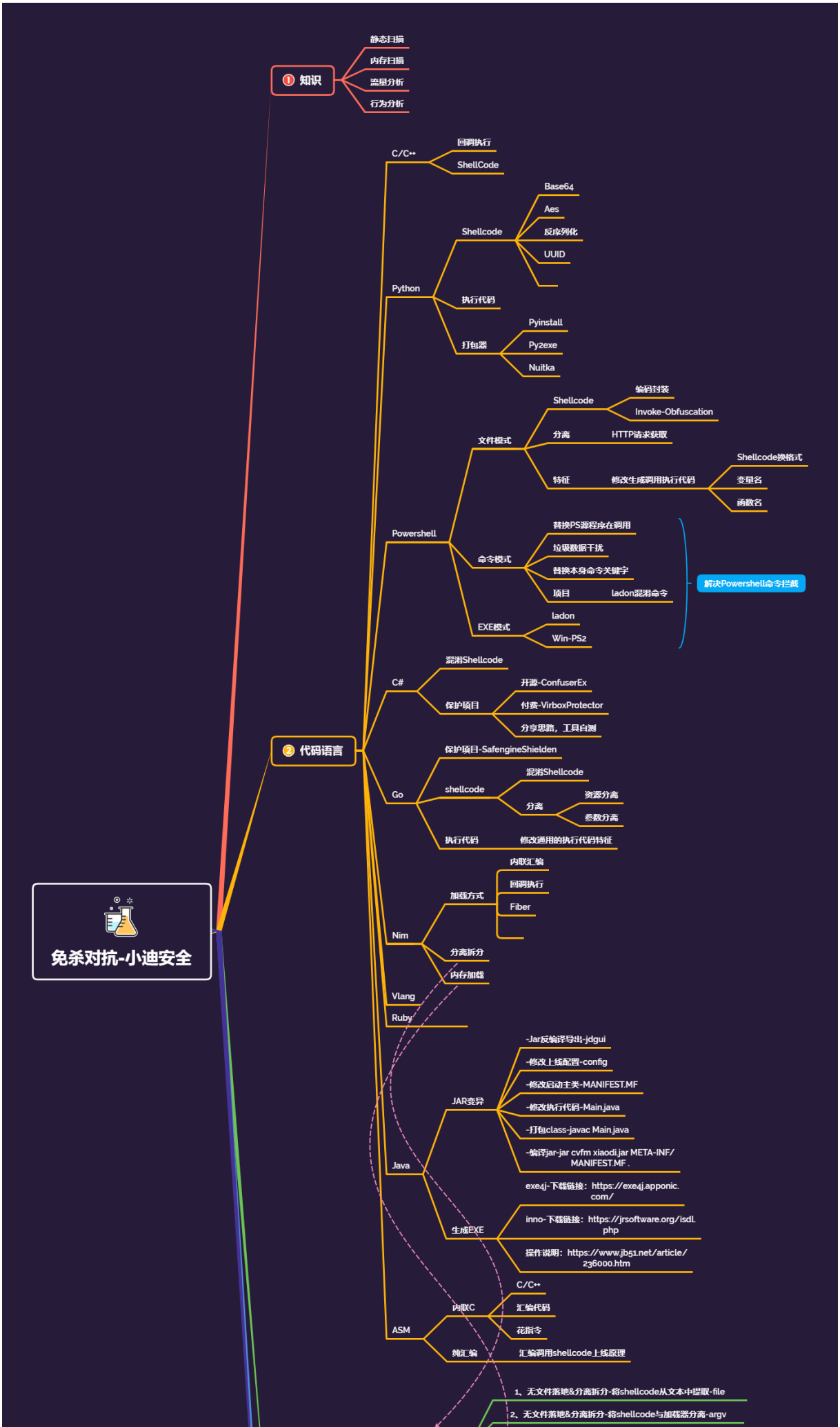
免杀对抗-EDR&Syscall-hook&DLL 反射注入&白加黑&隐写分离&加载器


免杀对抗-小迪安全
① 知识
静态扫描
内存扫描
流量分析
行为分析
② 代码语言
C/C++
回调执行
ShellCode
Python
Shellcode
Base64
Aes
反序列化
UUID
执行代码
打包器
Pyinstall
Py2exe
Nuitka
Powershell
文件模式
Shellcode
编码封装
Invoke-Obfuscation
分离
HTTP请求获取
特征
修改生成调用执行代码
Shellcode换格式
变量名
函数名
命令模式
替换PS源程序在调用
垃圾数据干扰
替换本身命令关键字
项目
ladon混淆命令
EXE模式
ladon
Win-PS2
解决Powershell命令拦截
C#
混淆Shellcode
保护项目
开源-ConfuserEx
付费-VirboxProtector
分享思路，工具自测
Go
保护项目-SafengineShielden
shellcode
混淆Shellcode
分离
资源分离
参数分离
执行代码
修改通用的执行代码特征
Nim
加载方式
内联汇编
回调执行
Fiber
分离拆分
内存加载
Vlang
Ruby
Java
JAR变异
-Jar反编译导出-jdgui
-修改上线配置-config
-修改启动主类-MANIFEST.MF
-修改执行代码-Main.java
-打包class-javac Main.java
-编译jar-jar cvfm xiaodi.jar META-INF/MANIFEST.MF .
生成EXE
exe4j-下载链接：https://exe4j.apponic.com/
inno-下载链接：https://jrsoftware.org/isdl.php
操作说明：https://www.jb51.net/article/236000.htm
ASM
内联C
C/C++
汇编代码
花指令
纯汇编
汇编调用shellcode上线原理
1、无文件落地&分离拆分-将shellcode从文本中提取-file
2、无文件落地&分离拆分-将shellcode与加载器分离-argv

#知识点：

1、DLL 劫持-自写&导入

2、DLL 劫持-重写&分离

3、syscall-底层&项目

#章节点：

编译代码面-ShellCode-混淆

编译代码面-编辑执行器-编写

编译代码面-分离加载器-编写

程序文件面-特征码定位-修改

程序文件面-加壳花指令-资源

代码加载面-Dll 反射劫持-加载

权限逻辑面-杀毒进程干扰-结束

工具数据面-通讯内存流量-动态

对抗目标：

X60 Defender 某绒 管家 VT 等

编程语言：

C/C++ Python C# Go Powershell Ruby Java ASM NIM Vlang 等。

涉及技术：

ShellCode 混淆，无文件落地，分离拆分，白名单，DLL 加载，Syscall，加壳加花，资源修改，特征修改，二次开发 CS，内存休眠，进程注入，反沙盒，反调试，CDN 解析等

演示案例：

- C&Py-DLL 劫持-语言-调用加载
- C&C++-DLL 劫持-白加黑-导入加载
- C&C++-DLL 劫持-白加黑-导出编译

- C&C++-DLL 劫持-白加黑-图片分离
- C&C++-Syscall 底层-加载器生成-项目

#C&Py-DLL 劫持-语言-调用加载

-dll.c 生成 dll

-dll.py 调用 dll

#C&C++-DLL 劫持-白加黑-导入加载

StudyPE 导入 DLL 生成

#C&C++-DLL 劫持-白加黑-导出编译

导出源码：Dependencies

编译执行：dllmain.cpp

#C&C++-DLL 劫持-白加黑-图片分离

图片分离：https://github.com/Mr-Un1k0d3r/DKMC

gen

set shellcode xxxxxx

run

编译执行：dllmain-image.cpp

#C&C++-Syscall 底层-加载器生成-项目

Windows 下有两种处理器访问模式：用户模式（user mode）和内核模式（kernel mode）。用户模式下运行应用程序时，Windows 会为该程序创建一个新进程，提供一个私有虚拟地址空间和一个私有句柄表，因为私有，一个应用程序无法修改另一个应用程序的私有虚拟地址空间的数据；内核模式下，所有运行的代码都共享一个虚拟地址空间，因此内核中驱动程序可能还会因为写入错误的地址空间导致其他驱动程序甚至系统出现错误。内核中包含了大部分操作系统的内部数据结构，所以用户模式下的应用程序在访问这些数据结构或调用内部 Windows 例程以执行特权操作的时候，必须先从用户模式切换到内核模式，这里就涉及到系统调用。

x86 windows 使用 sysenter 实现系统调用。

x64 windows 使用 syscall 实现系统调用。

https://github.com/icyguider/Shhhloader

https://j00ru.vexillium.org/syscalls/nt/64/

https://github.com/7BitsTeam/EDR-Bypass-demo

https://cloud.tencent.com/developer/article/1944012

涉及资源：

补充：涉及录像课件资源软件包资料等下载地址