Moving OS fingerprint adaptively in SDN network

Writer	Yulong Wang, Jun Guo, Jiuchao Zhang and Zhanming Guan
Year of Publication	2017
Source	2017 3rd IEEE International Conference on Computer and Communications
Link	https://ieeexplore.ieee.org/document/8322586
Notes Link	
Status	Finished
Type	MOFA SDN

总结与摘录

1. 研究成果

- 开发需求
  - OS 指纹混淆
  - 终端透明性：用户不可感知
  - 可接受的网络通信延迟
- 创新点
  - OS 指纹混淆度与攻击威胁度正相关
  - 无需在终端主机上安装软件，无需在网络交换设备之间放置专用设备
  - 不显著影响网络服务
  - 不显著影响网络通信质量
  - 保留了对 OS 检查的管理权（？不是很懂在讲什么）

2. 研究背景

- 现有的 OS 混淆思路
  - 设置 IDS rules 阻拦 ICMP echo 响应包　　缺点：影响正常通信（ping）
  - 部署在终端的 TCP / IP 指纹混淆软件　　缺点：需要软件的安装与升级
  - 蜜罐　　缺点：无法阻止【 OS 指纹识别】本身
  - 基于 SDN 的 OS 混淆方法（ 检测→ 混淆）　　缺点：
    1. 传统的【检测 → 混淆】思路会造成通信延迟
    2. 恶意流量检测不一定准确
    3. 未陈列可能的 OS 混淆空间
    - FPH: specific flow tables 引流恶意通信报文，OS 指纹混淆设备
    - Kampanakis: onePK（思科），混淆 OS 指纹 + 增加通信负载
- 动态 / 静态 OS 指纹检测优缺点对比

	优点	缺点
动态	检测时间可控	可能会由于特殊探针而被发现
静态	不容易被发现	对攻击者流量收集与分析的能力有较高要求

3. MOFA

- 实验环境
  - 实验设备
    - a computer (two 1.70GHz Intel i7 CPU and 4 GB RAM) with Mininet, OpenFlow 1.3 and Ryu controller installed
    - another computer (2.90 GHz Intel Pentium and 4 GB RAM) acting as a Web and File server
  - 网络环境
    - all links between switches in Mininet are set to 100 Mbps bandwidth and 1ms delay
    - all links between a switch and a host are set to 20 Mbps bandwidth and 0.2 ms delay

  All links
  between a switch and a host are set to 20 Mbps bandwidth
  and 0.2 ms delay
- MOFA 架构
  - 子系统 1#
    - 基于 sFlow 的流量模拟系统（sFlow agent 安装在 SDN switch 上）
    - 当流在预定义的时间限制内处于非活动状态 / 处理缓存变满 / RST 和 Fin 数据包到达时，流结束
  - 威胁概率模型（N_o是啥？？）

$$P_{rst}^{i} = N_{rst}^{i} / N^{i} \quad (1)$$

$$P_{sna}^{i} = N_{sna}^{i} / N^{i} \quad (2)$$

$$P_{iu}^{i} = N_{iu}^{i} / N^{i} \quad (3)$$

$$P_{o}^{i} = N_{o}^{i} / N^{i} \quad (4)$$

$$E^{i} = -\sum_{j \in \{rst,sna,iu,o\}} P_{j}^{i} \log_2 P_{j}^{i} \quad (5)$$

$$P_{a}^{i} = \begin{cases} 1 & E^{i} > 2 * E^{i-1} \\ \max\left(0, \dfrac{E^{i} - E^{i-1}}{E^{i-1}}\right) & otherwise \end{cases} \quad (6)$$

in which, N^i, N^i_{rst}, N^i_{sna}, N^i_{iu} and N^i_o are the counts of all packets, RST packets, SYN without ACK packets and ICMP port unreachable packets in the *i*th time window, respectively. E^i denotes the entropy of the packets in the *i*th time window. P^i_k is the fingerprinting possibility.

特点

- **包熵可根据需求扩展**（很有趣，在验证准确性与可行性的情况下可借鉴）
  - 根据流量变化判断恶意检测
  - 子系统 2#
    - 基于 OpenFlow 的独立控制平面
    - 根据当前的 OS 指纹检测威胁和网络性能来控制 SDN switch 间的数据包传输
  - MTD 指纹引擎
    - 可以是一台单独的机器，也可以是部署在 SDN 控制器上的应用程序
    - 向子系统 1# 查询网络情况并选择防御策略
    - 向子系统 2# 发送更改 switch 配置的命令
- 移动 OS 指纹混淆
  - 特点
    - 扰动随机程度与恶意检测可能性正相关
    - 修改 OVS 代码，向 flow table 中添加随机属性
    - 根据网络环境移动不同范围的指纹攻击面

- 工程实现

Algorithm 1 moving fingerprint attack surface

Require: Suspicious Host Set H with fingerprint probabilities Pr, fingerprint attack levels P_l<P_m<P_h<1, TCP seq moving factors M_t, M_b

```
1.Set C = Mt
2.For each incoming packet Pi
3.  If (Pi.TCP_FLAG is SYN-ACK) then
4.    Pi.TTL=(rand(1,120)+Pi.TTL)% 255;
5.  End if
6.  If (Pr_H(Pi)≥Pl) then
7.    If (Pi.TCP_FLAG is SYN-ACK) then
8.       Reorder the options in Option Field;
9.       Pi.win_size = Pi.win_size – rand(0, 1000);
10.      Pi.ID = rand(0, 2^16 – 1);
11.   End if
12.   if (Pr_H(Pi)≥Pm) then
13.      If (Pi.TCP_FLAG is RST or RST-ACK) then
14.         Pi.DF= Pi.DF xor rand(0,7);
15.         Pi.TTL=(rand(1,120)+Pi.TTL)% 255;
16.         Pi.win_size = Pi.win_size – rand(0, 1000);
17.      End if
18.      If(Pi.protocol is ICMP and Pi.type=0|3|14|16|18|30) then
19.         Pi.DF= Pi.DF xor rand(0,7);
20.         Pi.TTL=(rand(1,120)+Pi.TTL)% 255;
21.         If(Pi.type=3) then
22.            Pi.payload = "";
23.         End if
24.      End if
25.      If(Pr_H(Pi)≥Ph) then
26.         If(Pi.TCP_FLAG is SYN-ACK  and C>0 ) then
27.            Pi.ISN = random value;
28.         End if
29.         C =( C < -Mb?) Mt:(C – Mt*(1-Pr_H(Pi)));
30.      End if
31.   End if
32.  End if
33.End for
```

- 评价指标选择
  - 网络通信
    - 网络服务响应时间
    - 文件批量下载时间
  - 安全性
    - active：Nmap
    - passive：p0f

3. 我的评价

- 关于语言：看上去像是翻译软件直译出来的，读起来很别扭
- 关于稿件：MOFA - C. Threat and Performance Sensing 处有一明显的漏写错误
- 关于阈值设置：没有实验数据支持
- 关于思路：有值得借鉴的地方