IoTCandyJar：面向物联网设备的智能交互蜜罐

TongboLuo，Zhaoyan Xu，Xing Jin，Yanhui Jia，Xin Ouyang Email：{tluo，zhxu，xijin，yjia，xoyang}@paloaltonetworks.com Palo Alto Networks Inc.

摘要

近年来，新兴的物联网（IoT）引起了人们对联网嵌入式设备安全性的担忧。迫切需要开发合适且具有成本效益的方法来发现物联网设备中的漏洞-以便在攻击者利用它们之前解决它们。在传统的IT安全中，蜜罐通常用于了解动态威胁环境，而不会暴露关键资产。在以前的BlackHat会议上，传统的蜜罐技术已经被讨论过多次。在这项工作中，我们专注于honeypot的适应性，以提高物联网的安全性，并讨论为什么我们需要有一个巨大的创新，为物联网设备构建蜜罐。

由于物联网设备的异构性，手动制作低交互蜜罐是负担不起的;另一方面，购买所有的物理物联网设备来构建高交互的蜜罐是负担不起的。这种困境迫使我们寻求一种创新的方式来为物联网设备构建蜜罐。我们建议使用机器学习技术来自动学习物联网设备的行为知识，并构建“智能交互”蜜罐。我们还利用多种机器学习技术来提高质量和数量。

1. 引言

近年来，新兴的物联网（IoT）已经导致对网络连接设备的安全性的日益关注。与传统的个人计算机不同到2020年，互联设备的数量将从50亿增长 著名的物联网设备探索网站Shodan [22]已经表明，数百万物联网设备在没有适当保护的情况下暴露在互联网中。因此，寻找物联网设备上的漏洞成为白黑之间战斗的前线

蜜罐技术是发现0-day漏洞的常用方法之一一般来说，蜜罐以真实的方式模仿交互，并鼓励未经请求的连接执行攻击。尽管蜜罐是一种被动的方法，但它仍然可以在大规模攻击的早期阶段有效地发现零日漏洞利用尝试。有许多商业蜜罐产品

在Github上有超过1000个蜜罐项目。然而，我们发现大多数用于物联网设备的蜜罐都是低交互的，具有固定的回复逻辑和有限的交互水平。

另一方面，物联网设备上的漏洞通常高度依赖于特定的设备品牌甚至固件版本。这导致攻击者倾向于在远程主机上执行多个检查，以在启动利用代码之前收集更多事实证明，现有蜜罐项目的这种有限的交互水平不足以通过检查，无法捕获真正的攻击。虽然IoT设备的恶意软件比传统的恶意软件相对简单，但如果不适当地处理响应，IoT蜜罐的有效性将受到损害。

本文提出了一种自动化、智能化的物联网平台构建方法--*智能交互*。利用互联网上公共可用的物联网设备来收集对我们的蜜罐捕获的请求的潜在响应，我们能够获得不同类型的物联网设备的行为。然而，为了通过攻击者的检查，我们还需要学习最佳对策 更高的概率成为攻击者的期望值。 我们利用多种启发式和机器学习机制来自定义扫描过程，并改进回复逻辑以扩展会话，从而有更高的机会捕获利用代码。

本文其余部分的组织结构如下：第2节简要介绍了蜜罐以及我们构建智能交互物联网蜜罐的动机。第3节解释了我们如何自定义扫描模块*IoTScanner*，以从互联网上收集原始行为知识。第4节讨论了我们对IoT响应进行聚类并生成*IoT-ID*以查明IoT设备准确性的方法。第5节讨论了如何利用机器学习技术来改进回复逻辑。评估和我们感兴趣的发现，从捕获的流量在第6节。

2. 背景和动机

在本节中，我们将讨论一种模拟物联网设备行为的新方法，以构建智能交互蜜罐。我们面临的困境是，无论是低交互还是高交互方法都不能用于为物联网设备构建蜜罐。我们的蜜罐可以同时实现高覆盖率（低交互蜜罐的优点）和行为保真度（高交互蜜罐的优点）。由于我们的蜜罐只模拟物联网设备的行为，发送的请求和代码

从攻击者到我们的蜜罐将被处理为真正的设备。因此，与高交互性的蜜罐不同，我们的蜜罐没有被破坏的风险。

2.1 传统蜜罐

在蜜罐研究领域，有两类蜜罐：*高*和*低*交互作用。低交互蜜罐只不过是一种模拟服务，给攻击者提供非常有限的交互级别，例如一种名为honeyd的流行蜜罐[18];高交互蜜罐是完全成熟的操作系统，使用真实的系统供攻击者交互。一篇很好的调查论文[4]回顾了自2005年以来的所有蜜罐研究项目

低交互和高交互蜜罐都有优点和缺点。低交互蜜罐是有限的，易于检测;高交互蜜罐通常更复杂，而且部署和维护通常需要更多的时间。此外，当部署高交互蜜罐时涉及更多风险，因为攻击者可以完全控制蜜罐并滥用它，例如，攻击互联网上的其他系统。因此，有必要引入和实现数据控制机制，防止蜜罐的滥用。这通常是使用非常危险和资源密集型的技术，如全系统仿真器或rootkit类型的软件，如第三代蜜网[2]。

致力于自动化构建高交互性的Honeypot。 [7，13，14]已经研究了蜜罐的自动构建交互系统。研究了如何以独立于协议的方式对特定请求进行响应，包括流量聚类、状态机的建立和状态的简化。然而，我们的工作与以前的研究有很大的不同 所有先前的项目都依赖于从实时流量捕获的特定协议的大型数据集。 以该数据集为基础事实，他们从流量中提取公共结构作为模板，并生成随机数据来填充模板。另一方面，由于物联网设备的各种定制协议，很难找到这样一个干净完整的流量数据集。此外，实时流量仅包含一小部分恶意流量，并且难以从数据集中识别它们。由于蜜罐只需要模拟攻击者感兴趣的行为，这就可能导致漏洞的产生。 没有必要学习所有的行为，但重要的是要学习关键的行为，这些行为很可能会从实时流量中遗漏。

物联网蜜罐的挑战。 蜜罐并不是一个新话题。许多蜜罐框架（例如honeyd [18]，GenIII honeynet [2]和nepenthes [1]）都是开源或商业化的。为什么我们要讨论在物联网设备上构建蜜罐？ 简短回答 物联网蜜罐不能建立在传统蜜罐技术上。物联网设备的异构性使得低交互物联网蜜罐的开发非常耗时;和真实设备的价格，以及作为模拟器或模拟器的访问权限，都可能成为构建的高交互物联网蜜罐。

我们必须用我们为物联网设备创建亲型蜜罐的旅程来解释它。最直接的方法是搜索开源的物联网蜜罐，我们确实找到了很多，比如IoTPot [17]，SIPHON [9]等。尽管如此，它们都是低交互的蜜罐

比如“Honeyd”[18]，它只不过是一个模拟服务，给攻击者提供了非常有限的交互级别。显然，那些低交互的蜜罐只能为我们获取有限的信息。 由于异质性 由于物联网设备的种类繁多，模仿来自不同供应商的不同类型的物联网设备的交互是具有挑战性的。尽管并非不可能，但这需要大量的技术工作，这些工作不容易重用。例如，考虑IP摄像机的情况，为了以现实的方式可视化或模拟它们的行为，人们不仅需要向攻击者广播一些视频，而且还需要忠实地对诸如倾斜摄像机的命令做出

由于低交互蜜罐不能满足我们的需求，我们转向另一个方向，建立高交互蜜罐，这也是不可能的 有两种方法可以构建一个高交互的蜜罐：物理地或虚拟地。在传统的定义中，物理蜜罐是网络上具有自己IP地址的真实机器。 在物联网的背景下， 这意味着我们需要购买不同品牌和不同类型的真正的物联网设备，并将它们连接到互联网上。由于有限的空间和财务限制，该解决方案是不实际的，更不用说引入和实现数据控制机制以防止滥用蜜罐的风险。另一方面，虚拟蜜罐是一种模拟易受攻击的系统或网络的软件。但事实是，与操作系统（例如，Android、Windows），大多数IoT设备没有任何可用的模拟器。

2.2 智能交互蜜罐

为了克服这些挑战，我们提出了一个通用的框架，为物联网设备构建*智能交互*蜜罐我们将解释如何以及为什么我们需要智能交互蜜罐。

什么是智能交互？智能交互的目标对客户端的正确响应应该能够扩展与潜在攻击者的会话，欺骗他们通过检查并发送利用请求。 为了实现这一目标 它要求我们的系统自动收集有效的响应作为候选，通过与攻击者交互，学习过程帮助蜜罐优化每个请求的正确行为。

智能交互是如何工作的？图1显示了我们系统的概述。有4个主要组件单独运行，但在执行过程中共享数据。*IoT-Oracentral*

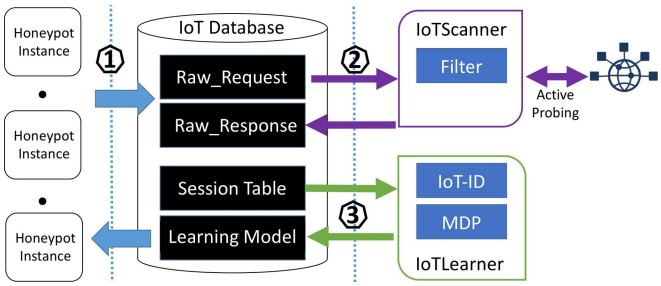

图1IoTCandyJar高级架构。

数据库，存储了我们获得的关于I oTdvi ce s的所有信息。我们在Amazon AWS和Digital Ocean上部署的honeypot实例中包含honeypot模块。它们的目的是接收攻击者的流量并与攻击者进行交互，引诱他们进行真正的攻击。它们将定期与IoT-Oracle同步，以将新接收的原始请求推送到表*原始请求*，并检索*IoT知识*表以获得IoT设备的最新知识。

该模块*IoTScanner*利用捕获的攻击该col-
选择的响应将被存储在*原始响应*表中以供进一步分析。该模块*IoTLearner*利用机器学习算法根据攻击者的反馈和给定的响应训练模型。经过几轮的学习迭代，我们的蜜罐可以优化一个模型来应对攻击者。

在第一时刻，我们的系统表现得就像一个低交互的蜜罐，因为我们的系统从关于物联网设备及其行为的零知识开始。我们已经评估过，在很短的时间内，蜜罐可以覆盖很多物联网设备。

模拟就够了吗？对于高交互蜜罐，他们通常在虚拟机中部署真实系统或仿真器来对攻击者（例如，攻击者）做出反应。响应请求或执行上传/注入的脚本）。我们的hon-eypot纯粹基于学习到的知识生成响应，但不运行它。由于物联网设备的攻击面，大多数攻击都是使用HTTP请求和其他物联网相关协议发起的，并最终尝试在没有身份验证的情况下注入命令或获取登录凭据。注入的命令非常简单和简洁，通常由*wget*命令组成，用于将shell（busybox）代码从恶意服务器放置到设备，作为- sign权限并执行它。有关攻击和注入命令的详细信息，请参见第3.4节。 我们的目标 捕获注入的脚本并从中提取恶意外壳代码。因此，与攻击者的交互并不复杂，只需回复他们的响应。因此，模拟物联网设备的行为足以构建有效的蜜罐。

为什么是智能交互？正如我们在前面几段中所讨论的，当前对物联网设备的攻击往往简单明了，并不困难 用蜜罐抓到他们然而，对于大多数攻击，攻击者通常会对目标设备进行一些检查，以了解设备是否易受攻击。使用Cisco Firepower上的上一个示例（CVE-2016-6433），在发送漏洞利用请求之前，攻击者可能会检查设备是否为Cisco Firepower以及版本是否为6.0.1这
可以通过发送请求到*ip：443/image/favicon.png来完成？v=6.0.1-1213*，检查响应状态是否为200。还有
攻击者可能进一步尝试用某个凭证登录。如果这些步骤中的任何一个失败，攻击者将停止攻击，我们的蜜罐可能无法捕获真正的漏洞。

3. IOT-Scanner：主动探测物联网行为

构建智能交互蜜罐的第一步是收集来自所有类型物联网设备的响应福图

当然，从互联网上，我们可以找到所有可访问的物联网设备。因此，我们设计并实现了一个模块，*IoT-Scanner*，以主动探测互联网上的物联网设备，并收集它们对我们从蜜罐捕获的各种请求的响应扫描结果将被存储在中央数据库中，作为我们的

重要的是，我们希望我们的探测是礼貌的，并防止不必要的流量到互联网。我们探测的细节可以在我们以前的工作中找到[24]。为了使扫描过程有效且不非法，我们采用了多种*过滤方法*来缩小远程主机的范围，使流量更多地与物联网相关，并消除有害请求

3.1 IP过滤。

与43亿IPv4地址空间相比，大物联网设备的数量仍然很小。收集物联网设备的IP地址子集可以提高扫描结果的质量，还可以加快扫描过程。据我们所知，我们无法为物联网设备找到可靠且完整的IP地址集。因此，我们从开始构建自己的IoT-IP数据库。

设备类型	供应商	计数
IP摄像机	海康威视	8,785
	Avtech	4,391
	大华	4,002
	NetWave	3,713
	东芝姆	1,302
	摩维斯	202
	不详	892
路由器	TP-Link	4,560
	Linksys	3,604
	网件	2,461
	天空	2,186
	水牛科技	235
	ZyXEL	1,232
打印机	HP	3,200
	爱普生	2603
	佳能	1,989
	哥哥	1,230
智能路由器	Linksys	1,581
	不详	330
防火墙	华为	783
	福蒂内	623
	思科	525
	SonicWall	553
	3com	197
	杜松	30
VoIP网关	D-Link	6,369
	Innovaphone	3,598
	AddPac	1,671
	彩色	959
	埃奇沃特	100
ONT	阿尔卡特朗讯	1,263

表1IoT设备的IP集合。

我们从在线平台获取原始IP信息，例如Censys [6]，ZoomEye [25]和Shodan [22]，或者我们自己部署的端口扫描工具（例如：MASSCAN [15]）。我们使用端口扫描工具来收集有关给定IP地址的基本信息，如远程计算机中打开的端口以及该打开端口的横幅信息

港口。这些在线平台也收集了类似的信息，但它们提供了查询工具，可以更方便地在数据库中搜索 (2)给予 一个关键字，什么是IP地址服务包含该关键字的内容？

目前，广泛采用的方式是使用不同类型的横幅信息来确定给定IP地址后面的机器是否是IoT设备。 对于示例，所形成的Telnetb aner被用于识别设备类型。自1990年《计算机滥用法》发布以来，强烈建议计算机在允许用户登录之前显示横幅登录横幅是在违规者未经授权访问之前通知他们的最佳方式。

因此，我们在这些平台上持续进行两次搜索：（1）搜索端口号（2）搜索关键字，即品牌名称，用于IoT设备。我们定期在我们的数据库中存储和更新这些信息。首先，我们通过现有的开放平台定期搜索信息在使用这些信息之前，我们进行另一轮同步扫描，以验证端口确实打开。如果是这样，我们将把这些IP作为更高的优先级，并探测它们 收集可能的答案。到目前为止，我们已经收集了超过40，000个物联网设备的IP地址，如表1所示。除了来自开放平台的馈送之外，如果我们的目标端口对于物联网设备是唯一的，我们还进行互联网范围的探测。

3.2 端口过滤。

在65535个端口中，IoT设备可能仅侦听其中的一小部分以进行交互。其中最流行的是TR-069服务的端口7547，*TR-069*服务是一种基于SOAP的协议，用于远程管理终端用户设备。它通常由诸如调制解调器、网关、路由器、VoIP电话和机顶盒之类的IoT设备使用。另一个例子是用于通用即插即用（UPnP）协议的端口1900，67%的路由器打开该端口以便于设备和程序发现路由器及其相应的配置。对于通过嵌入式Web服务器提供远程配置的物联网设备，它们通常会暴露80、8080、81等端口我们还监视和扫描物联网设备大量使用的协议所使用的端口，包括用于XMPP的端口5222，用于CoAP的端口5683和用于MQTT的端口1883/8883。表2突出显示了我们从分析和先前调查中确定的端口列表[16]，我们将优先扫描这些端口上的流量。

设备类型	开放端口
IP摄像机	81（35%）、554（20%）、82（10%）、37777（10%）、49152、443、83、84、143、88
路由器	1900（67%）、21（16%）、80（1%）、8080、1080、9000、8888、8000、49152、81、8081、8443、9090,8088,88,82,11,9999,22,23,7547
打印机	80（42%）、631（20%）、21（13%）、443（7%）、23,8080,137,445,25,10000
防火墙	8080、80、443、81、4433、8888、4443、8443
ONT	8080、8023、4567
杂项	5222（XMPP），5683（CoAP），1883/8883（MQTT），

表2IoT设备使用的端口。

3.3 种子请求筛选。

扫描器的关键输入之一是蜜罐捕获的请求。我们的目标是了解IoT设备如何对它们中的每一个做出反应，以便模拟这些行为。在我们的数据库中，我们在过去几个月中总共记录了大约1800万扫描所有这些请求是不可行和高效的，我们清理了原始请求数据库，以消除明显与物联网无关的流量。例如，近一半（53%）的捕获请求不包含有效负载。其他非物联网流量包括*BitTorrent*协议（7%），MS-RDP协议（5%）和SIP协议（4%）。注意HTTP流量中的两种特殊流量非常重要：HTTP代理流量（6%）由代理代理重定向（通常看起来像“GET http://full-HTTP/1.1”）和仅用于根路径的扫描流量（1%）。对于UDP流量，我们还发现其中大部分（6%）是一些shell代码，例如使用busybox的命令。通过应用我们上面解释的启发式方法，我们成功地将用于扫描的原始请求的总数从1800万减少到1000万（如图2所示）。

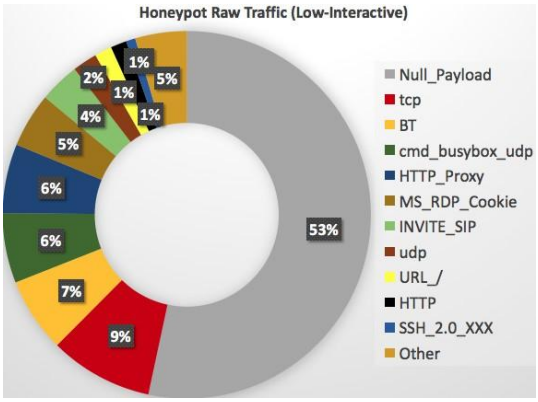

图2来自低交互的原始请求。

此外，我们根据端口对流量进行分组，进一步减少每个组中重复和相似的请求。图3显示了每个端口上的请求数。最流行的端口攻击者倾向于扫描端口80，超过90%的流量是有意义的HTTP请求。由于最近的僵尸网络*Mirai*，在过去的几个月里，端口7547上的扫描突然增加

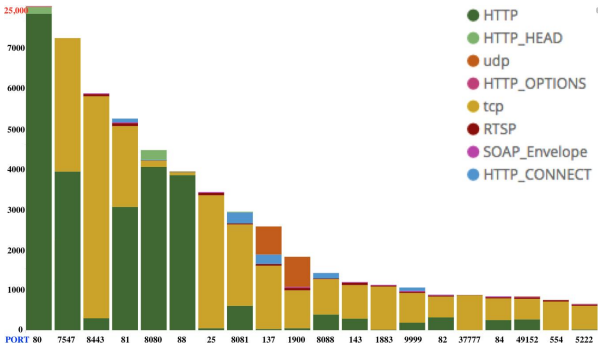

图3按端口划分的流量类型。

3.4 利用流量过滤

当我们使用捕获的流量作为内容来扫描互联网时，重要的是过滤掉危险的，例如包含漏洞代码的请求。在我们的系统中，我们利用多个启发式和现有的检测工具（例如：snort规则，我们的防火墙）来检测流量中的漏洞利用代码。一旦检测到漏洞利用请求，我们将标记它，

远程命令执行（RCE）。命令注入是IoT设备上最流行的攻击之一。攻击者通常在请求中嵌入恶意外壳代码，并将其发送到易受攻击的设备。由于实施不佳，易受攻击的IoT设备将在未经授权的情况下执行注入的命令。通常，注入的代码可以用处理请求的易受攻击程序的特权来执行（例如Web服务器）。

HTTP*POST*请求中的主体是嵌入命令的最常见位置。其他协议可以用于连接通信和数据传输。例如，MIBO等承诺使用位于服务的实现中的记录的SOAP利用的其他设备，该服务允许ISP使用TR-069协议（端口7547）来配置和修改特定调制解调器的设置。其中一个设置错误地允许执行Busybox命令（如wget）来下载恶意软件。例如，*NewNTPServer1*字段中的嵌入式外壳代码将丢弃恶意代码并执行它。

```
POST/UD/act?1HTTP/1.1
主机：x.x.x.x：7547
SOAPAction：urn：dslforrum-org：service：
Time：1

<?xmlversion="1.0"?>
<SOAP-ENV：Body>NewNTPServer1>
http：//host/1;chmod 777 1;./1
</NewNTPServer 1>/SOAP-ENV：正文>/SOAP-
ENV：信封>
```

其他协议也可用于嵌入恶意外壳代码。例如，多种类型的D-Link路由器容易受到UPnP远程代码执行攻击，允许在SSDP广播分组中嵌入外壳代码。M-SEARCH包的内容变成外壳配置文件。

```
M-SEARCH * HTTP/1.1主机：
239.255.255.250：1900
ST：uuid：';telnetd -p 9094; ls'
Man："ssdp：discover"
MX：2
```

混淆和解码是逃避检测的常用方法。例如，我们已经捕获到url/*shell?%的流量75%6E%61%6D%65%20%2D%61*，这是从原始url /shell中解码出来*的?uname+-a*用于命令注入。我们还集成了常用的混淆机制的检测。

信息披露。由于缺乏访问控制和认证，大量IoT设备无意中泄露了关于它们的配置的信息和系统的其他敏感信息。通常，这些信息可以用来发动更强大的攻击。虽然它对目标远程设备无害，但我们也会对此类请求进行清理。

例如，D-Link个人Wi-Fi热点、*DWR-932*、暴露CGI脚本/*cgi-bin/dget.cgi*来处理大多数用户端和服务器端请求。它会回复未经授权的用户的请求，因此攻击者可以通过在URL中填充*DEVICE web passwd*作为*cmd*参数的值，以明文形式查看管理或WiFi密码路径遍历是收集泄露信息的另一种攻击类型，例如传递像*../这样的URL etc/shadow*。

数据回火。物联网设备上的许多漏洞允许攻击者篡改设备上的数据。例如，由操作系统*AirOS 6.x*允许未经身份验证的用户通过*airOS*Web服务器的HTTP将任意文件上传和替换到*airMAX*设备，因为"php2"（可能是因为补丁）不验证POST请求的"文件名"值。攻击者可以通过覆盖/*etc/passwd*或/*tmp/system. cfg*等文件来攻击设备。与检测路径遍历攻击类似，我们对请求中敏感文件的路径进行了清理

3.5 扫描结果

由于预算的原因，我们只将物联网扫描仪部署在 我们实验室里有三台机器。它们从一个共享的*redis*队列中获取种子请求，新捕获的请求也将被插入到这个队列每一秒我们都在发送
*300*个不同的请求，并将超时设置为3秒。为了加快扫描过程，我们倾向于重用已建立的会话到以前扫描的因此，我们同时向同一台主机发送10个请求。 我们还有3台机器定期检查现有标记的物联网设备的开放端口的变化，并通过互联网扫描以找到更多可用的主机。

对于现有的种子请求，我们成功地完成了大约一周的扫描，并在数据库中收集了200万个响应，尽管许多开放的端口未能回复任何响应或重置连接。由于大多数扫描数据是HTTP流量，因此我们从中提取状态代码（表3）以快速分析它们。对于所有端口，响应代码403（禁止），404（未找到）和401（未授权）是IoT设备的前3个状态代码。

Rsp Prt	8000	80	8080	88	7547
403	651,646	120,659	12,953	26,660	0
404	88,034	175,497	30,746	10,789	3,832
401	31,468	36,388	36,863	3,870	373
200	3,483	3,742	1,289	300	1267
501	481	1,898	6,337	3	6,080
307	40	0	0	0	0
未知的	52	1,693	10	2	2720
其他	1,320	8,193	1,938	6	5140

表3HTTP协议的扫描结果。

4. IOT-ID：针点物联网设备

目前，我们确定给定IP地址背后的机器是否要检查响应中是否存在一些预定义的例如，如果我们发现给定IP地址中的一个端口返回一个包含模式

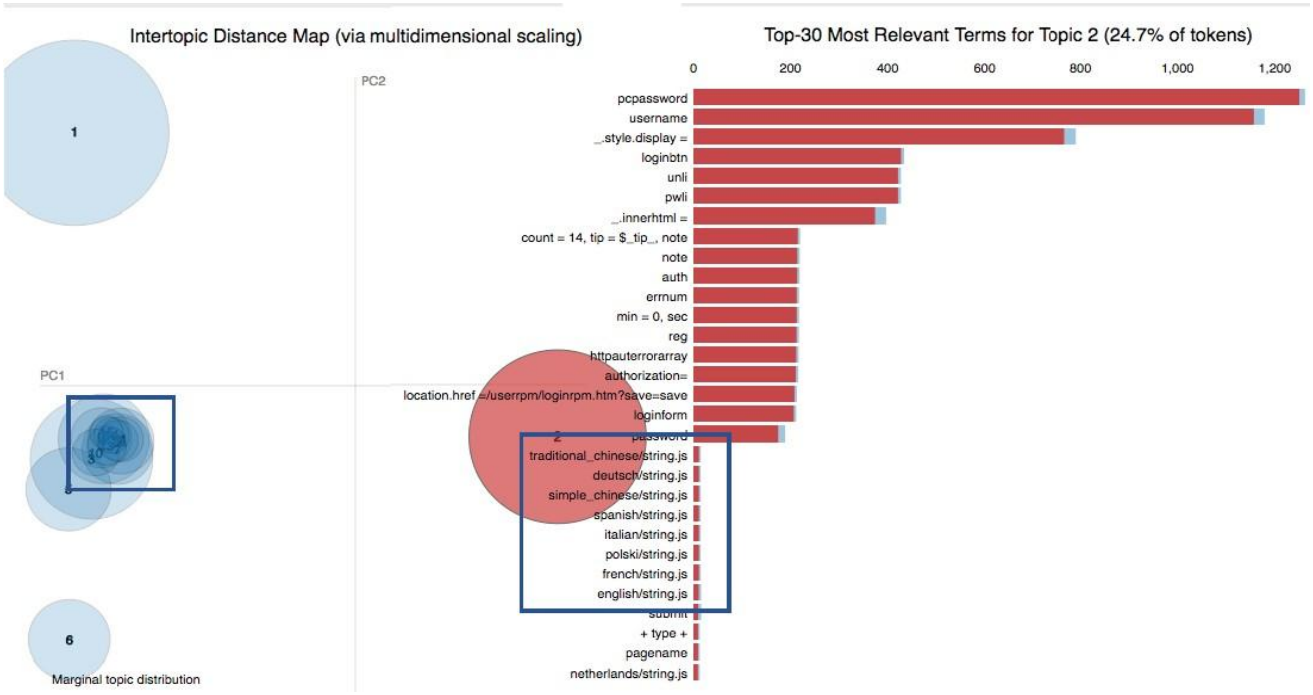

图4示例性LDA模型

路由器及其版本。然而，许多因素可能导致关于IoT设备的模糊甚至错误的信息。例如，我们观察到一些IP地址在不同端口中返回多个IoT设备的横幅内容。在80端口返回摄像机的登录入口，在99端口返回交换机的管理页面。另一个问题是，我们发现成千上万的物联网设备经常更改其IP地址。因此，当我们从中获取横幅内容时，我们将IP地址标记为IoT设备，当我们扫描它以收集对各种请求的响应时，它将更改为非IoT设备。

因此，我们提出了*IoT-ID*的概念，使我们能够区分不同的物联网设备，并获得它们的我们的想法是利用机器学习算法对扫描结果进行聚类，并从中提取模式作为某种类型物联网设备的签名

基于LDA的解决方案。 为了找到签名，我们仔细研究了我们收集的物联网流量。我们对流量的见解是，类似设备的流量应该包含类似的模式。基于此，我们把我们的问题作为一个自然语言处理问题。我们的目标是找到一些词的组合，这在NLP中被称为*主题*

使得该组合可以唯一地标记相同品牌和固件的公共标识、公共表达。我们建议使用生成统计模型，潜在狄利克雷分配（LDA）[11]，它允许通过未观察到的（*主题*）来解释观察集。具体来说，我们将每个响应视为一个文档，并将回复响应的IoT设备类型作为其主题。我们通过预定义的分隔符将每个文档拆分为一系列*单词*。然后计算每个词在语料库中的统计分布，并将其组织成n个类别。每个类别被进一步公式化为主题，并且每个主题被表示为生成统计分布。

我们的实现基于一个开源实现[12]。我们所生成的模型中的一个由图4中的py_s_o_o_该模型包含来自6个不同路由器供应商的HTTP流量，我们为他们总结了15个不同的主题。如图所示，LDA模型可以成功地集群的话，这是在每个库和固件唯一。图的右侧显示了一个库示例。该库提供多种语言支持，LDA可以将这些特定于语言的单词组合在一起作为一个主题。同时，我们可以发现一些主题共享一些共同的词，这意味着它们的流量符合常见的HTTP语法。

我们的LDA模型的输出是一些主题，即 词与其置信概率之间的一系列映射关系。基于输出，我们可以有效地聚类收集到的相似流量，并提取主题到单词的映射作为其IoT-ID。

5. IOTLEARNER：学习物联网行为的智能引擎。

在IoTScanner的帮助下，我们的蜜罐能够根据接收到的请求向客户端回复有效的响应，而不是响应固定的请求。在本节中，我们将讨论如何利用马尔可夫决策过程模型来优化响应选择，以最大可能捕获攻击。

5.1 IoTLearner概述。

对于每个单独的请求，*IoTScanner*模块可以从远程主机收集至少数百甚至数千个响应。所有这些都是*有效的*反应，但只有少数是*正确*的。这是因为，对于给定的请求，各种IoT设备可以重新配置。

根据他们自己的逻辑来处理它并产生响应。最简单的例子是访问其Web服务的根路径的请求：一些设备可以回复登录门户页面，其他设备可以将其重定向到另一资源，而其余设备可以用错误页面来响应。因此，所有扫描结果都是作为对客户端的响应的潜在候选者，但挑战在于找到攻击者所期望的扫描结果

我们的方法。 我们的方法背后的想法是首先从候选池中随机选择响应，并记录客户端的下一步行动。 我们假设，如果我们碰巧选择了正确的一个，攻击者会认为我们的蜜罐是易受攻击的目标物联网设备，并且他们会继续发送恶意有效载荷（例如：注入命令）。因此，我们需要将每个事务存储在*会话表*中，并利用机器学习技术从数据集中提取正确的行为

系统架构。*IoTLeaner*模块的架构如图5所示，用于从数据库获取原始响应每一个进入蜜罐的请求都会被转发到这个模块，而选择的响应会被返回给客户端。该模块的核心部分是*选择引擎*，它规范化的请求，并从扫描结果中提取潜在的响应列表。在随机选择模式下，它只是从候选列表中随机选择一个并返回它。在MDP选择模式中，它首先从规范化请求中定位图中的状态，然后由模型选择最佳状态。

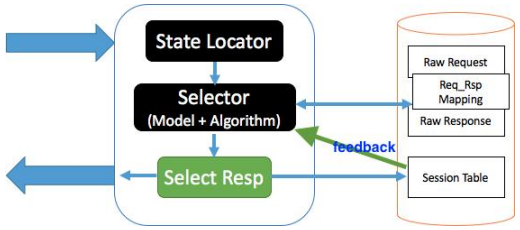

图5IoTLearner概述。

由选择引擎做出的每个决定将创建新事务以扩展当前会话。所有会话信息都将存储在*会话表*中。表中的每一行都像tuple一样表示一个事务
<req1、*rsp1*、*conn_info>*和*conn_info*是作为源IP、源端口、目的地IP、目的地端口和协议的连接信息。例如，如果引擎选择id为*rsp1*的响应来回复id为*req1*的传入请求，则该行将被插入到会话表中

学习最佳对策的挑战。 有许多因素使学习过程变得困难。第一个是，并非所有与我们的蜜罐对话的客户端都是攻击者。这就导致了一个问题，即并非所有的会话都是恶意的。只有那些能达到剥削对我们来说才是重要的。

5.2 模型制定。

我们将讨论如何将响应选择问题公式我们假设客户端是继续会话还是执行at-
粘性仅由先前请求的响应确定。这是一个合理的假设，基于我们对现有恶意软件样本的最佳了解。因此，我们可以使用一个简单的数学模型来近似会话活动的统计结构，该模型被称为*1阶马尔可夫特性*。

马尔可夫决策过程。马尔可夫决策过程[19]，也称为随机控制问题，是标准（非隐藏）马尔可夫模型的扩展。MDP是一种用于在结果不确定时进行顺序决策的模型可以在该特定状态下执行动作的收集，这些动作用于将系统移动到新状态。在每个*决策时期*，将基于所选择的动作通过转变概率函数来确定下一状态。它可以被视为一个*马尔可夫链*，其中状态转移是唯一确定的过渡功能，灰和采取的行动，在前一步。行动的后果（即，奖励），并且政策的效果并不总是立即知道的。因此，当当前状态空间的收益不确定时，需要有一些机制来控制和调整政策。该机制统称为*强化学习*。

问题表述。在标准强化学习模型中，智能体与其环境交互这种交互采取代理感测环境的形式，并且基于输入选择要在环境中执行的动作。每个强化学习模型通过与动态环境的试错交互来学习从情况到动作的映射。该模型由多个变量组成，包括决策时期（t）、状态（x，s）、转移概率（T）、回报（r）、行动（a）、价值函数（V）、折扣（γ）和估计误差（e）。

强化学习任务的基本规则是Bell-man方程[3]，表示为：

$$V^{*}(x_t) = r(x_t) + \gamma V^{*}(x_{t+1})$$

它可以解释为最优策略的状态x_t的值是从状态x_t开始执行最优动作直到研究最终状态时的加强之和。折扣因子γ用于指数地减少将来接收的增强物的权重从定义上讲，强化学习问题本质上是求解一个动态规划问题。因此，RL的标准解决方案是使用值迭代，它将值V*表示*为查找表。 那么算法就可以
通过在状态空间中执行扫描来找到最优值函数V^{*}，通过以下方式更新每个状态的值：
更新策略，直到状态值没有变化（状态值已收敛）。一般更新策略可以表示为：

$$\Delta\omega_t = max_a\left(r(x_t, a) + \gamma V(x_{t+1})\right) - V(x_t)$$

然而，要在我们的问题中应用RL，有一个限制a-
的作用。我们的问题本质上是一个*非确定性的*马尔可夫决策过程，这意味着在每个状态下，存在 - 转移概率函数T，以确定下一状态。 换句话说，我们的学习策略是探索、用以前没有使用过的响应*进行回复*和*利用用具有已知高回报的响应进行回复之间的概率权衡*。应用一般

赋值迭代不可能在没有附加知识或某些决策修改的情况下计算必要的积分。因此，我们应用Q学习[21]来解决必须在一组积分上取最大值

Q学习不是找到从状态到状态值的映射，而是找到从状态/动作对到值（称为Q值）的映射[10]。Q学习利用Q函数，而不是具有相关联的值函数。在每个状态中，存在与每个动作相关联的Q值。Q值的定义是在执行相关联的动作并且随后遵循给定策略时接收的增强物的总和。同样地，最优Q值的定义是当执行相关联的动作并且随后遵循最优策略时接收的增强的总和。

因此，在我们使用Q学习的问题中，Bellman方程的等价形式化为：

$$Q(x_t, a_t) = r(x_t, a_t) + \gamma max_{a_{t+1}} Q(x_{t+1}, a_{t+1})$$

直接Q学习的更新规则被形式化为，a是学习率：

$$\Delta w_t = a[(r(x_t, a_t) + \gamma max_{a_{t+1}} Q(x_{t+1}, a_{t+1}, w_t)) - Q(x, a, w_{t-1})] \frac{\partial Q(x_t, a_t, w_t)}{\partial w}$$

奖励功能。 奖励函数r：(x_t, a_t) *转* r 将某个值r分配给处于状态和动作对(x_t, a_t)中。奖励的目标是定义每对的偏好，并最大化最终的奖励（最优策略）。在我们的上下文中，即时奖励$r(x_t, a_t)$反映了我们在交互过程中所取得的进展，当我们选择响应a_t来请求x_t并且我们移动到下一个状态x_{t+1}时。由于进度可以是负的或正的，因此奖励函数也可以是负的或正的。定义奖励的启发式方法是，如果响应a是攻击者所期望的目标设备类型，并且攻击者通过在下一个请求中发送利用代码来发起攻击，则奖励必须是正的且巨大的。相反，如果响应不是预期的响应（例如，反映了不易受攻击的设备版本），则攻击者可以停止攻击并结束会话。它导致了死胡同状态，并导致负报酬。换句话说，我们奖励那些可能导致我们最终攻击数据包的响应并惩罚那些导致无结果的人

我们的设计之一是将奖励分配为等于最终会话长度的值，因为我们相信攻击者发送的请求越长，包含恶意有效载荷的机会就越高。标准会话是2，这意味着在我们发送响应后，至少有一个来自同一IP的其他请求在同一端口。如果没有观察到进一步的转变，我们为该响应分配负的回报。其他替代的奖励分配可以基于我们是否接收到一些已知的漏洞利用分组。

5.3 MDP模型构建

我们解释了如何启动模型所需的参数，以根据现有结果执行计算

状态和动作。 在没有任何协议语义概念的情况下构建马尔可夫模型的状态可能导致缺乏通用性和稀疏状态空间。 因此，它无法处理任何尚未看到的东西。 解决方案是通过将相似的状态分组为单个状态来简化和概括状态。在我们的例子中，我们希望将类似的请求和类似的响应分类到同一个组。由于物联网设备使用的通信协议数量巨大，我们必须以协议无关的方式进行。以前的研究[7，13，14]已经研究了如何在不理解协议的情况下简化状态，在HTTP，SDP，NSS，NTS等上。详细的实现可以在这些论文中找到。一般来说，它们都依赖于比对算法来识别多个字符串的相似部分作为协议的结构。

状态转换概率。 状态转移概率可以由转移函数$T(s, a, s')$来描述，其中a是在当前状态s期间可执行的移动动作，并且s'是某个新状态。更正式地，函数转换函数$T(s, a, s')$可以由以下公式描述：

$$P(S_t = s' | S_{t-1} = s, a_t = a) = T(s, a, s')$$

其中a是在当前状态s期间可执行的动作移动，并且s'是某个新状态。

在我们的上下文中，转移函数$T(s, a, s')$指的是如果我们将响应a回复给客户端，则接收请求s'作为同一会话其当前请求s的应答。 来衡量 对于(s, a, s')的每个组合，我们通过从候选集合中随机返回响应并将会话信息保存到*会话表来*部署朴素算法。Af-

在运行一段时间之后，我们能够收集大量会话，并且我们解析它们中的每一个以计数每个组合(s, a, s')的出现，其被表示为$C(s, a, s')$。

转换函数$T(s, a, s')$的值被定义为如下：

$$T(s, a, s') = C(s, a, s') / \sum_{x \in S} C(s, a, x)$$

响应选择的在线Q学习算法。 基于Q-学习模型，我们的学习过程从在t_0决策时期接收请求开始。给定请求，我们应用我们的匹配算法来选择一组*相同*的响应。我们采用g-贪婪[23]策略进行动作选择。特别是，我们为每个可用的响应分配均匀的概率作为初始transsac-灰函数。使用此策略，我们可以选择具有g概率的随机动作，也可以选择具有$1-g$在给定状态下给出最大奖励的概率。然后我们开始Q学习迭代并更新Q学习表。当我们学习一个状态和动作对$r(x_t, a)$的强化时，我们首先反向传播并更新Q查找表。因此，我们可以通过删除以负奖励结束的响应并更新g值来进行调整。迭代结束，直到模型收敛

在实践中，我们的模型是在线运行和实时更新因此，它可能不会收敛并达到全局最优。然而，我们认为该模型仍然是有价值的，因为它不仅允许我们丢弃这些不期望的响应，而且还使会话尽可能地进行

5.4 物联网蜜罐的MDP。

我们将使用真实世界的例子来解释如何构建MDP并计算每个响应的概率。为了演示的目的，我们简化了图中的状态和动作的数量，并将模型表示为状态空间图。在图中，每个矩形（黑色）表示单个状态，每个圆圈（橙色）表示在特定状态下可以采取的动作。图中的箭头指的是在采取某个动作之后从一个状态到下一个状态的转变在我们的上下文中，每个状态都是一个唯一的请求抽象，请求的每个操作都是要回复的唯一响应候选

从HNAP协议的会话表构建MDP。HNAP协议非常简单，特别是对于攻击场景。例如，在攻击我们的蜜罐之前，攻击者通常会向URL*/HNAP 1/*发送一个请求，以 获取一个SOAP响应（一个XML格式的文档，包含结果数据），其中包含主机及其支持的SOAP操作的信息。有些设备不支持HNAP协议，例如*SonicWall*防火墙返回*404 Not Found*页面，*TRENDnet*路由器返回*401 Unauthorized*页面。其他人可能会回复有效的SOAP响应，但在响应中包含不同的信息，例如*LinkSys*路由器响应中的型号名称为*WRT 110*，*D-Link*路由器响应中的型号名称为*DIR-615*。由于我们没有任何关于HNAP协议和预期行为的先验知识，我们可能会尝试将它们中的每一个发送给攻击者 并记录会话信息。

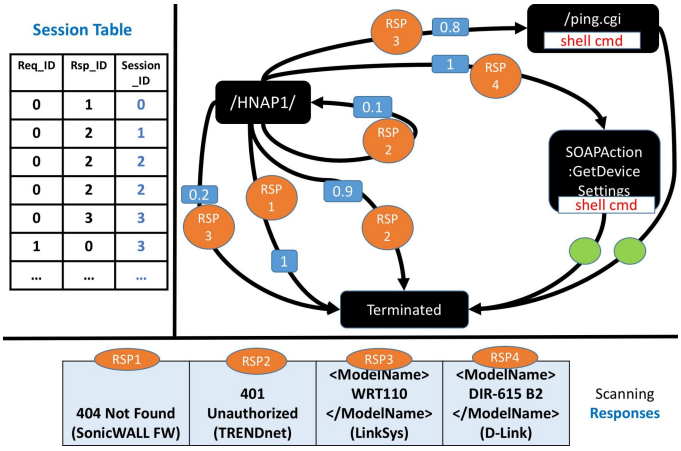

图6从会话表构建MDP状态图。

CGI脚本MDP图。由我们捕获的完整会话数据构建的图更加复杂。我们选择请求到*CGI*脚本的URL，并生成它们的图形。 *CGI*脚本被许多类型的 物联网设备，包括摄像头、路由器等大量的预先检查和漏洞已执行的*CGI*脚本．如图7所示，诸如*getstatus.cgi*、*检查user.cgi*并*获取摄像头参数*。攻击者经常扫描。由于它们在没有任何特权的情况下可访问，因此攻击者倾向于从它们收集设备信息
在几个请求之后，会话转到特权*CGI*
脚本和脆弱的人。

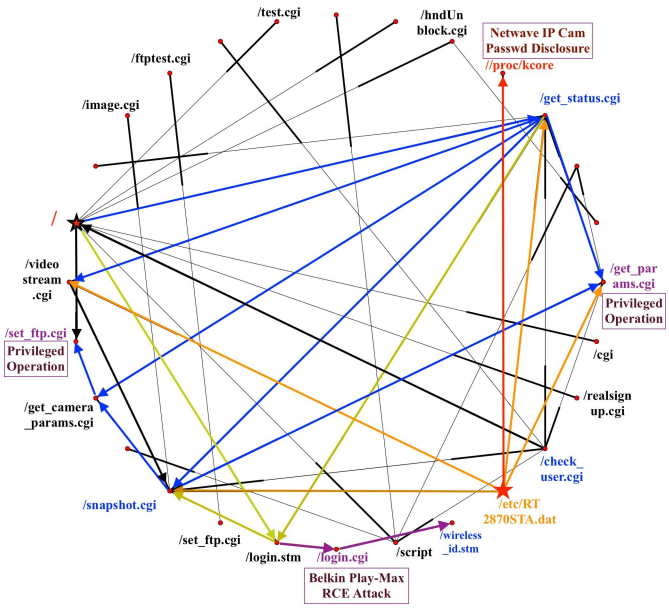

图7CGI脚本MDP图。

6. 评估

在本节中，我们将评估智能交互蜜罐的有效性我们已经将其部署在*数字海洋*的5台虚拟机中。由于预算的原因，我们只选择最小的一个，配置为：512MB内存，1个CPU，20G SSD硬盘，标准网络带宽。在准备阶段，我们利用低交互蜜罐捕获的*100万个*请求作为种子。
扫描互联网，并收集了数百万的反应，使用扫描仪，我们上面解释。

6.1 会话改进

每个会话的长度是一个关键的eas- 一个可衡量的指标，以证明我们的学习过程的有效性：我们可以引诱攻击者发送的请求越多，我们捕获exploitation的机会就越高。对于低交互蜜罐，大多数会话在2个事务内结束，因此平均值低于2。

在我们的设置中，我们从扫描结果中选择30k个唯一的响应，并将副本下载到每个蜜罐实例。也下载响应与对应的种子请求之间的映射。我们在前两周将honeypot配置为随机回复模式，以收集客户的反应。通过随机选择机制，我们成功地从客户端接收了更多的请求。然而，如图8所示，大部分会话仍然非常短。这个结果是合理的，因为我们可以从扫描器获得数百甚至数千个针对单个请求的唯一

每个蜜罐实例包含我们选择的30万个响应及其到特定请求的映射。2个hon-eypot实例使用随机选择算法，其将从使用当前接收的请求收集的池中随机选择响应;其余的都采用了MDP算法，如前所述

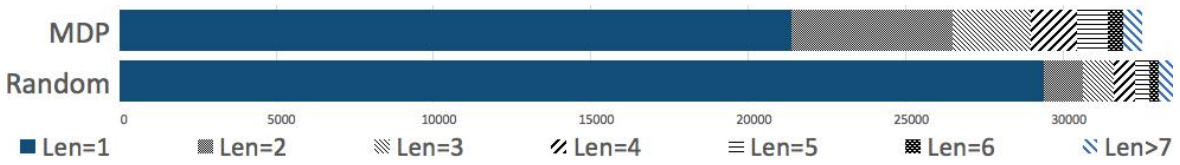

图8会话长度分布。

节。在一个月的运行中，我们已经收集了*100万个*有效请求.

6.2 捕获的攻击前检查和利用。

利用智能交互蜜罐，我们有上限

捕获了来自攻击者的更多精心编制的恶意请求我们-
通过分析会话信息，我们还识别了*50*种针对不同类型IoT设备（例如，IPCam、路由器、投影仪）。在这一部分我们重点介绍了几个案例

6.2.1 HTTP协议

HTTP协议被IoT设备广泛用于管理。 设备信息可以直接或间接地从所有类型的HTTP响应中泄露。我们根据响应的状态代码讨论我们的

200好。这是成功HTTP请求的最标准响应。Netgear路由器的几个版本在响应*/currentsetting. htm*上的请求时很容易泄漏型号版本、固件和其他信息。如果攻击者不能从响应内容中解析或获得有效令牌，他们就不会执行攻击动作。对于*Netwave*IP摄像机，我们观察到在URL*get status.cgi*、*/etc/RT2870STA.dat*和*login.stm上有*相当多的请求。访问这些页面不需要身份验证，这些页面会显示固件版本（用户界面和系统）、时间戳、序列号、p2p端口号或wifi SSID。

401未授权HTTP状态代码*401 Unauthorized*表示由于验证方法不正确或根本未验证而无法加载资源。 401响应几乎不包含任何其他有意义的信息。然而，它仍然可以嵌入帮助客户端执行下一动作的信息（例如：重新认证、重定向）。

例如，响应报头中的*WWW-Authenticate*字段用于描述身份验证模式[8]。通常，该响应通常从网络服务器返回到IoT设备，而不是从网络应用[1]返回。具有相同品牌的所有类型的IoT设备共享相同的机制。因此，攻击者可以利用该字段的值来确定当前的IoT设备是否易受攻击，特别是对于*Netgear*调制解调器或路由器。我们的蜜罐观察到对*NETGEAR R7000*和*NETGEAR R6400*模式的检查，以在此特定路由器版本上发起远程代码注入攻击（CVE-2016-6277）。

有时401-未授权响应的内容也可能泄露敏感信息。例如，netgear无线路由器（*N150 WNR 1000 v3*）在基本登录尝试失败时包含凭据（to- ken），如以下片段所示：

[1]403 Forbidden响应与Web应用程序的逻辑相关

```
HTTP/1.0 401未授权
WWW-应用程序： Basicrealm="NETGEARWNR1000v 3"
<htm l>< h ead>< t itle>401Un aut hor ize d</t itle></he ad>. ..
<form method="post"action="unauth.cgi? 2019 - 04 - 28 11: 00: 00
</form>/body>/html>
```

我们发现攻击者需要成功地找到pat-
在精心制作恶意有效负载以进一步利用路由器之前，从它中提取令牌。

404 Not Found当请求的资源找不到时，服务器将返回404-not-found响应。404页面也可以被攻击者用来识别IoT设备。例如，我们已经观察到对ZyXEL的调制解调器Eir D1000的攻击攻击者向URL/*globe*发送一个合法请求，并期望看到一个404-not-found页面，其中包含模式*home wan.htm*这个特殊的404页面告诉攻击者，主机支持基于SOAP的协议TR-069，他们可以通过将命令嵌入到请求的“NewNTPServer”字段中来向主机注入命令

6.2.2 在自定义方案中

除了HTTP协议外，还对物联网协议甚至自定义协议进行了初步检查。家庭网络管理协议（HNAP）是示例之一。这种基于SOAP的协议首先由Cisco在2007年发明，用于网络设备管理，其允许网络设备（例如，路由器、NAS设备、网络摄像机等）在网络设备上进行通信。安静地管理。由于其长期的错误实现，已经发现了许多攻击，例如利用*GetDeviceInfo*动作绕过身份验证和将shell命令注入*SOAPAction*字段以发起RCE攻击。正如我们在第5节中所讨论的，攻击者通常会检查对URL/*HNAP 1/*的响应，以获取设备支持的服务列表。另一个例子是*Netcore*和*Netis*路由器使用的自定义协议，它打开UDP端口53413作为远程配置的后门。攻击者发送一个8字节值为' x00 '的有效载荷，

6.2.3 使用回显命令。

远程命令执行是物联网设备上常见的攻击之一例如，最近在路由器*NETGEAR DGN 2200*上发现了其所有固件版本（v1-v4）的漏洞，该漏洞不需要ad- min访问来在路由器上执行shell命令。易受攻击的脚本是*ping.cgi*，它是为用户向路由器提交诊断信息而设计的。但是，由于实现缺陷，如果攻击者向此url发送POST http请求，并将命令作为有效负载中的参数*ping IPAddr*的值，则他们能够执行

没有人允许的命令。返回的页面包含注入命令的结果

我们已经捕获了试图利用此漏洞的恶意请求，但命令非常简单，例如echo命令。我们观察到，大多数恶意会话都是在带有echo命令的请求处终止的。这是因为攻击者通常在echo命令后生成一个随机字符串，如果执行该命令，响应内容应该包含完全相同的字符串。然而，由于随机字符串在每个请求中改变，所以扫描结果中的字符串与当前接收到的请求匹配的可能性很高。

```
POST/ping.cgi HTTP/1.1r
efe rer: htt p: //x. x.x.
x/D IAG_di ag. htm

IPAddr1=1 IPAddr2=2 IPAddr3=3 IPAddr4=4
ping=Ping
ping_IPAddr=12.12.12.12;echo"zP8ZDXwQCC";
```

如上面的请求示例所示，攻击者在发送真正的漏洞利用外壳代码之前检查响应中是否有随机字符串zP8ZDXwQCC我们通过将echo命令中的字符串插入到响应页面的正确位置来

7. 结束语

由于物联网的特殊性，利用传统的方法为物联网设备构建蜜罐具有挑战性。然而，对IoT设备的攻击在发起攻击之前对设备信息执行初步检查。如果没有与攻击者的适当交互机制，捕获完整的ex-ploit有效负载是非常困难的。我们提出了一种自动和智能的方法来收集潜在的响应，使用扫描仪和利用机器学习技术，以学习正确的行为，在与攻击者的互动。我们的评估表明，我们的系统可以改善与攻击者的会话，并捕获更多的攻击。

8. 参考文献

[1] Paul Baecher，Markus Koetter，Thorsten Holz，Maximillian Dornseif，and Felix Freiling.猪笼草平台：收集恶意软件的有效方法。*入侵检测的最新进展*，第165-184页。Springer，2006年。

[2] Edward Balas和Camilo Viecco面向蜜网的第三代数据捕获体系结构。2005*年，信息安全研讨会。05年的。第六届IEEE SMC年会论文集*，第21-28页。IEEE，2005年。

[3] 贝尔曼方程 https: //en. 我的天啊。 或无线/局域网质量。

[4] Matthew L Bringer、Christopher A Chelmecki和Hiroshi Fujinoki。调查：蜜罐研究的最新进展和未来趋势。*International Journal of Computer Network and Information Security*，4（10）：63，2012.

[5] 以下 公司简介bu si n ess in si d er. com/物联网 –生态系统 –物联网 –预测 –和 –商业 –机会 –2016 –2.

[6] censys.io网站。 https://censys.io/。

[7] Weidong Cui，Vern Paxson，Nicholas Weaver，and Randy H Katz.应用程序对话框的协议无关自适应重放。国家*统计局*，2006年。

[8] John Franks、Phillip Hallam-Baker、Jeffrey Hostetler、Scott Lawrence、Paul Leach、Ari Luotonen和Lawrence Stewart。Http身份验证：基本和摘要访问身份验证。技术报告，1999年。

[9] Juan David Guarnizo、Amit Tambe、Suman Sankar Bhunia、Mart´ın Ochoa、Nils Ole Tippenhauer、Asaf Shabtai和Yuval Elovici。虹吸：迈向可扩展的高交互物理蜜罐。*第三届ACM网络物理系统安全研讨会论文集*，第57-68页。ACM，2017.

[10] Mance E Harmon和Stepanie S Harmon。强化学习：一个教程。*技术报告*，1997年。

[11] L ate ntdir ichletaloc at ion。 https: //en. 我的天啊。 Org/wiki/Latent_Dirichlet_allotion.

[12] 开放源代码的lda实现。https: //radimrehurek. com/gensim/models/ldamodel. HTML.

[13] Corrado Leita，Marc Dacier和Frederic Massicotte。利用基于scriptgen的蜜罐自动处理协议依赖关系和对0天攻击的反应。在*入侵检测*。Springer，2006年。

[14] Corrado Leita，Ken Mermoud，and Marc Dacier. Scriptgen：honeyd的自动化脚本生成工具。在*计算机安全应用会议，第21届年会*。IEEE，2005年。

[15] masscan github https: //github. com/robertdavidgraham/masscan.

[16] 绿盟科技。我国物联网资产暴露分析。http://blog.nsfocus.net/wp-content/uploads/2017/05/Analysis-on-Exposed-IoT-Assets-in-China-0521.pdf，2017.

[17] Yin Minn Pa Pa，Shogo Suzuki，Katsunari Yoshioka，Tsutomu Matsumoto，Takahiro Kasama，and Christian Rossow. Iotpot：分析物联网妥协的兴起。*EMU*，9：1，2015.

[18] Niels Provos等人一个虚拟蜜罐框架。输入*USENIX安全研讨会*，2004年。

[19] 马丁·L·普特曼。*马尔可夫决策过程：离散随机动态*规划 John Wiley& Sons，2014年。

[20] 派戴维斯 https: //pyldavis. 请记住这一点。io/en/latest/.

[21] Q学习 https: //en. 我的天啊。 或 g/wiki/Q-learng。

[22] 是的。现在是最好的时候了。 https: //shodan. 1/0。

[23] 米歇尔·托基奇适应性？基于值差的强化学习中的贪婪探索。在*人工智能的进展*，2010年。

[24] Zhaoyan Xu，Antonio Nappa，Robert Baykov，Guangliang Yang，Juan Caballero，and Guofei Gu. Autoprobe：使用动态二进制分析实现自动主动恶意服务器探测。*第21届ACM计算机和通信安全会议（CCS 14）*，2014年11月

[25] zoomey e. 或我们都知道。 http: //zoomeye. 或 g/。