# 从无效的DNS流量中检测基于DGA的恶意程序

Manos Antonakakis1,3, Roberto Perdisci2,3, Yacin Nadji3, Nikolaos Vasiloglou1, Saeed Abu-Nimeh1, Wenke Lee3, David Dagon3

1Damballa公司 2佐治亚大学 3佐治亚理工学院

**摘要** 许多僵尸网络检测系统采用了已知的命令与控制(C&C)域名的黑名单去检测僵尸程序并封堵它们的流量。类似于基于签名的病毒检测，这样的僵尸网络检测的方法是静态的。因为黑名单只有在外部系统发现域名(通常是人工发现)后才会被更新。为了反击(黑名单)，僵尸网络的控制者已经开始使用域名生成算法(DGA)来动态产生大量随机域名，并且选择其中的一小部分作为实际的C&C通讯。也就是说，一个C&C域名是随机生成的并且它仅存活了短暂的时间，这样使得依赖于静态域名列表的方法变得不起作用。当然，如果我们能知道一个域名生成算法是如何工作的，我们就能提前生成这些域名并且仍然可以识别和封堵僵尸网络C&C的流量。现有的解决方案主要是基于对恶意程序的逆向，但是这并不是每次都能做到的。

在本文中我们提出了一种不需要逆向去检测随机生成算法的新技术。我们的想法是肉鸡查询了大量的DGA生成的随机域名，绝大多数查询都会返回一个域名不存在信息(NXDomain)。并且同一僵尸网络下(使用相同的DGA算法)的肉鸡会产生相似的NXDomain的流量。我们的方法是聚类和分类算法的结合。聚类算法将有着相似组成部分的域名以及相似访问特征的域名聚集在一起。分类算法用来将这些生成好的聚簇分配到已知的DGA模型中去。如果一个聚簇不能被分到已知的DGA中去，那么新的DGA变种或DGA家族就产生了。我们实现了一个系统原型，并且在北美大型ISP的真实DNS流量下测试了它。我们发现了12种DGA，其中一半是已知DGA的变种，另一半的DGA我们从未听说过。

**1** 引言

僵尸网络由一组被恶意程序感染了的主机(也称作“肉鸡”)组成。它们可以被攻击者通过命令与控制信道(C&C)远程控制。僵尸网络已成为网络犯罪分子发送垃圾邮件、盗取私人信息、网络钓鱼等等的主要平台。随着时间的推移，攻击者已经发明出基于不同网络结构的C&C信道。目前，大多数僵尸网络依赖于集中的C&C服务器，肉鸡查询一个事先定义好的C&C域名，该域名指向接收命令的服务器。这种集中式的C&C结构会受到单点失败问题的影响，因为如果指定的C&C域名垮掉的话，控制者就失去了对整个僵尸网络的控制能力。

为了克服这种局限，攻击者使用了基于P2P的C&C结构如Nugache，Storm，以及最近的Waledac，Zeus和Alureon(又名TDL4)。虽然P2P僵尸网络提供了一个更鲁棒的、不容易检测以及捣毁的C&C结构，但它很难去实现和维护。为了将集中式C&C的简单性以及P2P结构的鲁棒性结合起来，攻击者最近开发了一些通过自动生成伪随机域名来定位C&C服务器的僵尸网络。为了与主控节点联系，每个肉鸡定期地执行一个域名生成算法。即给一个随机数种子(如当前时间)，产生一组C&C域名的候选列表。接着肉鸡通过发送DNS请求去解析这些域名直到有一个域名解析到了C&C的IP地址为止。这种方法有着极强的灵活性因为即使一个或者多个C&C域名或者IP被发现和封堵了，肉鸡最终还是可以通过对下一次自动生成的域名的DNS查询得到C&C服务器的IP地址。一些代表性的基于DGA的僵尸网络有Bobax, Kraken, Sinowal(又名Torpig), Srizbi,

Conficker-A/B, Conficker-C和Murofet. 防御者可以通过尝试逆向僵尸程序，特别是它的DGA算法，来计算当前以及未来的候选C&C域名，从而达到检测、封堵甚至捣毁整个僵尸网络。但是逆向工程并不是一直可行因为攻击者可以快速地更新僵尸程序并且对程序做混淆(如加密，只有到时间才能被解密和运行)。

在本文中我们提出了一个叫Pleiades的新的检测系统。它可以不需要逆向僵尸程序，在监控的网络下检测出基于DGA的僵尸网络。Pleiades部署在本地递归服务器(RDNS)或网络边缘节点下，用来监控网络下的主机的DNS请求及应答。特别地，Pleiades对那些应答为域名不存在，也叫NXDOMAIN，即域名没有对应的IP地址及其他RR记录的请求进行分析。在本文的剩余部分，我们将这样的域名称作NXDomains。对NXDomains的关注是出于这样的事实：现在的DGA肉鸡往往查询大量的域名，这其中只有少数的域名成功解析到了C&C服务器的IP地址。因此，为了自动化地识别DGA域名，Pleiades寻找满足以下条件的相对较大的NXDomains簇：(1)有着相近的句法特征；(2)在给定时间内被多个感染主机查询。这样做的灵感是在一个大型网络中，例如我们做实验的ISP网络中，被同一DGA僵尸程序感染的主机一定很多。因此，每一个感染主机都会有一些NXDomains应答的DNS查询，而这些NXDomains中有一部分同样也被其他感染主机查询过。Pleiades可以自动化地识别(它们)并过滤掉由于错误拼写和配置导致的噪声。当Pleiades发现了一个NXDomains聚簇，它对该簇使用统计学习方法来建立DGA模型。这个模型被用来检测未来运行同一种DGA的感染主机，以及检测与NXDomains相似的活跃域名从而找出僵尸网络的C&C服务器的IP地址。

Pleiades具有能够不用辛苦地去逆向恶意软件进而就发现和建立新的DGA模型的优点。我们的系统可以在相关的恶意软件样本被捕获和分析之前检测出新的DGA僵尸网络。与以往的使用DNS流量检测恶意软件和恶意域名的方式不同，Pleiades充分利用了“被抛弃”的流量(即未成功的DNS解析)来(1)发现基于DGA的僵尸网络的增长率；(2)准确检测感染主机；(3)识别并封堵DGA肉鸡访问的活跃C&C域名。Pleiades通过监视本地网络的DNS流量来达到上述目标，不需要预先大规模地部署DNS分析工具。

此外，虽然基于网络流分析或深度报文检测的僵尸网络检测系统可以在本地网络内检测到感染主机，但是它们不能针对大型ISP环境下的大规模流量进行很好的扩展。另一方面，Pleiades采用“轻量级”的DNS监控方法，可以通过关注ISP网络下的所有DNS流量的一小部分来检测DGA恶意程序。这使得Pleiades可以很好地适应特别大的ISP网络，我们在ISP网络下评估了原型系统。

本文做出了以下贡献：

* 我们提出了Pleiades，第一个基于DGA的僵尸网络识别系统，它可以在大型ISP网络下有效地分析解析失败的域名流量从而自动识别僵尸网络。
* 我们给出了Pleiades的原型实现，并评估了它在一个包括了4种不同的已知DGA的僵尸网络产生的NXDomains以及拼写和配置错误导致的NXDomains的环境中识别DGA的准确度。我们的实验结果说明了Pleiades可以准确地检测出DGA僵尸网络。
* 我们花了15个月在大型的ISP网络下部署和评估了Pleiades的原型系统。通过实验我们发现了12种新的基于DGA的僵尸网络并且列举了感染主机。新发现的DGA有一半我们从未听过。

本文的其余部分安排如下。第2节讨论相关工作，第3节介绍Pleiades的整体概况。第4节介绍DGA的发现过程。第5节介绍DGA的分类和C&C检测过程。我们在第6节阐述了使用的数据集的性质以及得出参考标准的方法。第7章展示了实验的结果，第8章讨论了我们系统的局限，第9章全文总结。

**2** 相关工作

动态域名生成算法已经被恶意软件用于逃避检测和减轻系统的复杂度上，如Bobax, Kraken, Torpig, Srizbi和Conficker。为了发掘潜在的域名生成算法(DGA)，研究者通常需要逆向二进制代码。这样的任务很耗时间，并且需要熟练的逆向工程技术。

臭名昭著的Conficker蠕虫就是使用域名生成算法的最活跃的恶意程序之一。蠕虫的C变种每天产生50000个域名。但是Conficker-C每隔24小时只请求其中的500个。相比于C，蠕虫之前的变种A和B，每隔两三小时就会对循环访问域名列表。在Conficker-C中，生成的域名长度介于4~10个字符间，分布有110个TLD。

Stone-Gross等人是第一个报告域名速变(domain-fluxing)的。在过去，恶意软件使用IP速变(fast-fluxing)，即将一个域名指向多个IP地址从而避免C&C服务器很快被捣毁。相比之下，在域名速变中恶意程序使用域名生成算法产生一些域名，接着尝试连接其中的一部分。论文的作者也分析了Torpig的DGA并且发现bot程序使用了Twitter的API。具体来讲，它每天使用了Twitter搜索中最流行的第五个词的第二个字符作为随机数种子产生新的域名。Srizbi是另一个僵尸程序，它通过使用独特的幻数来实现DGA算法。研究者从多个bot的拷贝中发现了一些独特的幻数。这个幻数会和当前时间做异或运算然后生成一组不同的域名。在域名生成算法中，只有这些字符“q w e r t y u i o p a s d f”被使用。

Yadav等人提出了通过寻找随机生成的域名来识别僵尸网络的技术，并且在改进的方法中又包括了NXDomains和时间相关。他们通过自动检测Conficker僵尸网络来测试他们的方法，第一篇论文数据来自南亚的Tier-1的ISP，在第二篇论文中新增了一个大学的DNS流量。

Villamarin-Salomon和Brustoloni对比了两种识别僵尸网络C&C的方法。在第一种方法中，他们识别出那些有着高查询率或与时间相关的域名。他们用切比雪夫不等式和马氏距离来识别异常域名。在第二种方法中，他们分析了反复出现的“动态”的应答为NXDOMAIN的DNS请求。他们的实验结果表明，第一种方法是无效的，因为一些合法的服务在DNS中设置了短的生存时间(TTL)。然而他们的第二种方法取得了更好的检测结果并且识别出了可疑的C&C域名。

Pleiades与以上说过的方法不同，主要在这几个方面：(1)我们5种不同的僵尸家族进行了建模，包括Conficker, Murofet, Sinowal和Bobax；(2)我们使用了两种聚类方法对这些僵尸家族进行聚类。第一种利用了域名字符的分布以及2-grams。第二种依赖于肉鸡和域名的历史访问关系；(3)我们还建立了一个分类器用来预测从两种聚类方法中得到的恶意域名。

不同于以往的工作，我们的方法不需要主动探测来维持一份最新的合法域名列表。我们的方法不依赖于外部信誉数据库(如DNSBL)；相反，它只需要访问本地DNS查询流量从而识别出新的DGA的NXDomains聚簇。我们的方法不仅可以识别出新的DGA，它还可以对这些DGA进行建模从而将产生相似NXDomains的肉鸡归为一类。此外，在DGA的被鉴定域名列表中，我们的方法能精准定位到C&C域名。最后，我们指出我们的方法参考了前人试图检测和识别恶意域名的研究并做了补充。

**3** 系统整体流程

在本节中，我们会给这个基于DGA的僵尸网络检测系统的一个高层次的概述。如图1所示，该系统主要包括两个模块：DGA发现模块与DGA分类和C&C域名识别模块。我们讨论这两个模块的作用和它们包含的组件，以及它们是如何相互协作来“积极学习”并更新DGA僵尸网络检测系统的。我们会在第4节和第5节详细地描述了这些组件的具体功能。

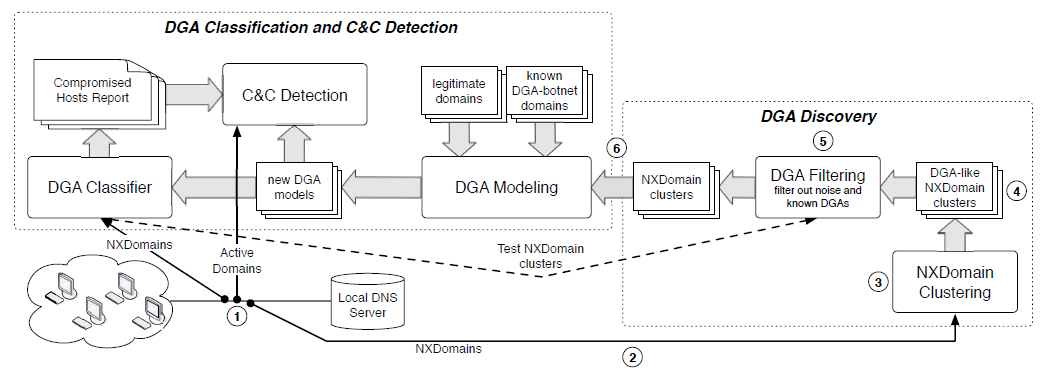


图1 Pleiades的整体流程

**3.1** DGA发现模块

DGA发现模块分析了未成功的DNS解析流量，如图1所示，它部署在DNS服务器的下游。在一定时间段内网络产生的所有NXDomains都会被收集在内。接着，收集到的NXDomains会根据以下两个相似准则分别被聚类：(1)域名字符串所具有的相似的统计特征(例如相似的长度、随机性和有关字符的频率分布等)；(2)域名被一组相同或相近的IP访问。对NXDomain聚类的主要目的是将那些使用同一种域名生成算法的主机和域名聚类出来。

理所当然地，由于这个聚类过程是无监督的，所以聚类出来的某些NXDomain簇里可能包含那些偶然错误的域名(例如由于拼写或者配置错误导致的NXDomains)。因此，我们需要对这些数据进行过滤。我们使用有监督的DGA分类器去修整这些聚类簇，修正的对象包括我们已经发现并建模的DGA产生的域名和那些与合法的域名相近的域名。DGA发现模块最终的输出是NXDomains的聚簇集，其中的每一个集合都代表了已知或者未知DGA产生的域名。

**3.2** DGA分类与C&C域名检测

每当发现一个新的DGA，我们使用监督学习方法为这样的域名建立一个模型，该模型用来描述新DGA产生的域名“长什么样”。特别地，我们建立了两种不同的统计模型：(1)一个为肉鸡产生的一组NXDomains赋予DGA标签(如DGA-Conficker.C)的多类统计分类器；(2)基于隐马尔科夫模型判断查询过的可能由一个DGA产生的活跃的单一C&C域名的类型。

DGA建模部分将不同的域名集合作为输入，如标记为合法的域名(non-DGA), DGA-Bobax, DGA-Torpig/Sinowal, DGA-Conficker.C, New-DGA-v1, New-DGA-v2等等，并使用这些数据作为训练集训练DGA多类分类器和基于隐马尔科夫模型的C&C域名检测模块。

DGA分类模块的工作过程如下。与DGA发现模块相似，我们监测DNS递归服务器中由每个主机产生的NXDomains流量。

给定一个主机产生的NXDomains的子集，我们提取与其字符串有关的一系列统计特征。接着将这些特征作为特征向量传入分类器函数，分类器会输出这些NXDomains是否由一个已知的DGA产生。如果一个主机产生的NXDomains被贴上了相关的DGA标签，那么这个主机也极有可能被相关的僵尸网络所控制。

一旦我们获得了那些感染主机的列表，我们就可以进行更深一步的检测。之前所有步骤都是围绕无效域名NXDomains展开的，现在我们把注意力集中到这些感染主机访问的活跃域名上。我们的目标是确定哪些由DGA算法生成的域名最后解析成了IP地址，换句话说，我们的目标是识别僵尸网络的C&C服务器。

为了实现这一目标，我们将感染主机访问过的所有可疑活跃域名都收集起来。接着我们对这些域名输入之前训练好的隐马尔科夫模型中，由它来决定单一活跃域名是由已知DGA产生的或是未知DGA产生的。

我们使用隐马尔科夫模型而不是分类器做判断的原因是需要对单一域名进行检测。DGA分类器不适合单一域名的检测，因为它根据一个感染主机产生的一组无效域名进而判断这组无效域名是哪种DGA产生的。另外一些在DGA分类过程中用到的特征也不能通过单一域名来提取，我们将在4.1.1节和5.2节详细讨论这一问题。

**4** DGA发现

DGA发现模块分析了监控网络的主机产生的NXDomains序列，并且使用了完全无监督的方式将由一种DGA自动产生的NXDomains聚簇在一起。为了实现这一目标，我们采取了如下步骤(见图1)。首先(第1步)我们收集在一段时间*E*内每个主机产生的NXDomains序列。而后(第2步)我们将所有NXDomains序列分割成不同的子集，并将每个子集转化为统计特征向量(见4.1.1节)。我们接着对这些子集使用X-means聚类算法，把有着相似字符特征的域名聚成一大类。

而另一方面(第3步)，我们基于一种完全不同的方法对NXDomains进行聚类，该方法考虑到两个NXDomains可能被多个主机集合查询。首先，我们构建一个二部图，左右两边的顶点分别代表不同的主机和不同的NXDomains。如果主机查询过NXDomain，那么就将主机顶点连接到NXDomain顶点上。这样我们可以看出不同的NXDomain被不同的主机集合重复查询。自然地，如果两个NXDomains被多个相同的主机查询，这就说明这些主机上运行着相同的DGA。我们可以利用NXDomains的这样的相似特征将他们聚在一起。

对NXDomains相似关系的两个完全不同的解读会在聚类关联阶段得到协调(第4步)。这一步通过将第2步和第3步的聚类结果想关联从而提高最终聚类结果的质量，并减少域名聚类中由于偶然相似而产生的噪音，例如来自不同网络用户产生的相似的错别字。

最终的聚类结果是不同组的NXDomains，每组包含的域名很大程度上是由同一个DGA产生的。对于得到的每一个NXDomains聚簇，我们考虑的问题是它们是属于一个已知的DGA，还是新发现的DGA。为了解决这个问题(第5步)，我们使用5.2节描述的DGA分类器，它经过特殊的训练可以辨认出由已知DGA生成的NXDomains集合。那些匹配到已知DGA模型的NXDomains将会被丢弃。另一方面，如果一个NXDomains聚簇与之前的DGA都不匹配，我们认为该NXDomains簇是由一个未知的、新的DGA产生的。这些NXDomains会被送到DGA建模模块(第6步)，在这里会更新(重新训练)DGA分类器组件。

**4.1** NXDomain 聚类

我们现在详细讨论NXDomain聚类模块。首先，我们介绍Pleiades将NXDomains的集合转换为特征向量过程中使用的统计特征，接着讨论这些特征向量是如何将相似的NXDomains聚簇在一起的。

4.1.1 统计特征

为了方便展示统计特征是如何计算的，我们首先介绍一些我们将这一节中使用的符号。

**定义和符号** 一个域名d由一组被点“.”隔开的标签组成，如www.example.com. 最右边的标签被称作顶级域名(TLD或TLD(d)),如com. 以点隔开的最右边的两个标签被称作二级域名(2LD或2LD(d))，如example.com. 三级域名(3LD或3LD(d))包括最右边的三个标签，如www.example.com，以此类推。

我们经常会提到将一个序列分割成一个长度为的子序列(子集),其中,.下标k代表在m个NXDomains集合中第k个长度为的子序列。子序列中的每个域名都可以按照如下方法转换成一个特征向量。

**n-gram特征** 给定一个有着个NXDomains的子序列，我们计算域名字符串的n-grams频率分布，其中n=1,2,3,4.例如，当n=2时，我们计算每个2-gram的频率。在这一点上，我们可以计算2-gram频率分布的中位数、平均值和标准差，从而就有了三个特征。我们为n=1,2,3,4分别做此操作，总共产生了12个统计特征。通过计算中位数、平均值和标准差，我们尝试发现n-grams频率分布的形状。

**基于熵的特征** 这一组特征计算了不同级的域名的字符分布的熵。例如，我们分别计算中域名的2LDs和3LDs的字符熵。为了更好地理解这些特征是如何计算的，考虑一个有个域名的集合。我们首先提取中每个域名的2LD，接着我们计算它的2LD的字符熵。接着我们计算集合的平均值和偏差。我们对全部域名的3LD也进行相同的操作。我们总共计算了6个特征，它们用来描述域名不同级数的随机性。这样做的直觉是大多数的DGA产生了看起来很随机的域名，并且我们想利用DGA的这一特征。

**域名结构特征** 这一组特征用来归纳中的NXDomains的结构信息，如它们的长度、独立TLD的个数以及域名级数的个数等。总共我们计算了14个特征。具体来说，给定，我们计算域名长度和域名级数的平均数、中位数、标准差和方差(各4个特征)。此外，我们计算出现在这些 NXDomains 的不同字符的个数(1个特征)，不同的顶级域名的数量，以.com为TLD的域名数量与其他TLD域名数量的比率(2个特征)。剩下的特征是不同TLD的出现频率的平均值、中位数和标准差(3个特征)。

4.1.2 使用统计特征进行聚类

若要找到相似NXDomains集群，我们进行如下操作。给定监控网络下主机访问过得所有NXDomains的集合NX，我们将NX分为长度为的子集若干，如4.1.1节提到的那样。假设m是NX中不重复的NXDomains的个数，我们将集合NX分割成个子集，其中.

对于得到的每一个子集，我们计算上述33个的统计特性。在我们将每个转换成对应的特征向量后，我们使用X-means聚类算法。X-means会将聚成X个簇，X是由X-means本身的自动优化过程计算出来的。此时，给定一个大小为个子集的聚簇，我们简单地将C里的求并作为一个NXDomain聚类。

4.1.3 使用二部图进行聚类

感染了同一种DGA的恶意软件的主机有着很大的可能生成互相重叠的NXDomains集合。另一方面，其他“非DGA”的NXDomains则不会被多个主机查询。例如，在一段时间内多个用户同时犯同样的拼写错误是不太可能的。这促使我们考虑那些被一些相同的主机查询的NXDomains的相似性，而通过利用这些相似性我们可以将可能由同种DGA产生的NXDomains聚类起来。

为了达到这一目的，我们构建一个稀疏矩阵M，其中行代表一段时间内查询过两个以上NXDomains的主机，列代表NXDomains。为了降低矩阵的维度，我们抛弃了只查询一个NXDomain的主机，因为鉴于它们产生的NXDomains太少，这些主机上不太可能运行着DGA。如果主机没有查询过NXDomain，令矩阵元素. 相反，如果查询过，令, 其中是权重。

所有与相关的非0项都被赋予权重, 其中是查询过的NXDomains的个数。很明显，M可以看成是一个二部图，其中主机顶点与NXDomains顶点被一个权重为的边相连。采用这种独特的计算权重的方法的原因是我们认为如果一个主机查询过的不重复的NXDomains的个数越多(也就是说越大)，这个主机就越不可能“代表”它查询过的那些NXDomains. 这是一种类似于文本挖掘领域中使用的逆文本频率指数(IDF)的方法。

当计算出M后，我们使用基于谱聚类的图划分策略(归纳见算法1)。首先，我们计算M的个特征向量(在我们实验中)，接着我们将每个NXDomain(M的每一列)映射到维向量上。这样做能很大程度上将NXDomains的向量维度从所有主机(M的每一行)减小至。接着我们对得到的维NXDomains向量使用X-means算法，从而基于它们的“主机关系”进行聚类。即被相近主机共同查询过的NXDomains会被聚簇在一起。

输入：稀疏矩阵,行代表*l*个主机，列代表*k*个NXDomains

[1]：归一化M：

[2]：依据M计算相似矩阵S=

[3]：通过特征值分解依据S计算个特征向量

令为从S特征值分解得来的包含k个维度为的向量的矩阵(向量代表第i个NXDomain压缩成维后的向量)

[4]：使用X-means算法聚类向量(也就是NXDomains)

输出：NXDomains簇

算法1 NXDomains谱聚类

4.1.4 关联聚类

我们现在有了基于两个不同的对“相似性”的解读得到的NXDomains聚簇。它们每一个都不是完美的，产生的聚类依旧存在噪音。关联两个聚类的结果可以帮助过滤噪音并且更有可能得到由同一DGA产生的NXDomains聚簇。聚类关联过程如下所示。

令为使用统计特征聚类的NXDomains簇(4.1.2节)，为从二部图方法中得到的NXDomains簇(4.1.3节)。我们计算所有可能的组合对的交集，其中，。对于关联后的聚类，我们会抛弃那些元素个数小于预定义的聚簇(即||<)，其余的聚簇会被传递到DGA过滤模块(4.2节)。不满足以上两个聚类方法的聚簇在下一阶段将不会被使用。依照以往的经验，在初步的实验研究中我们设定.

**4.2** DGA过滤

DGA过滤模块将聚类模块输出的NXDomain作为输入。此过滤步骤将新发现的NXDomains簇与已经发现并且建模的DGA产生的域名进行比较。如果聚类交集中的NXDomains被认为是由已知DGA生成的，我们就抛弃该。这样做的原因是DGA发现阶段的目的就是去寻找那些(很大可能上)由一个新的、从未见过的DGA产生的NXDomains簇。与此同时，过滤步骤还会判断一个NXDomains簇是否过于嘈杂，即它是DGA和非DGA域名的混合簇。

为此，我们利用DGA分类器(第5节详细描述)。宏观上，我们可以将DGA分类器看成是一个函数，它接收一个NXDomains集合，输出一个元组集合，其中是DGA的种类(例如DGA-Conficker.C)，是分类器认为是的可能的评分，*c*是DGA分类器可以辨别的不同种类的数量。

当DGA过滤模块收到一个新的NXDomains聚类交集时，它将该聚簇分成大小为的NXDomains子集若干，并将每个子集传入DGA分类器中。假设被分成了*n*个不同的子集。从DGA分类器中，我们可以得到n个元组集合，，…，。

首先我们考虑每个元组集合，其中k=1，…，n。标签被赋予了最大得分。如果相关的的种类太多，我们就认为聚簇过于嘈杂。具体来说，中最多的那个标签与集合个数n的比例时我们就认为过于嘈杂了。那些没有超过纯度阈值的聚簇将会被抛弃掉。此外，最多标签为合法的NXDomains簇也会被抛弃。

对于每一个过滤后的聚簇，我们进行一次额外的“纯度”检查。令聚簇的最多的标签为。在集合中，我们取出所有的得分。即我们选择那些将分类器认为域名子集为的可能性的得分，计算它们的平均数)、方差(注意在区间[0,1]内)。我们抛弃那些大于预设阈值的聚簇，因为我们认为聚簇中的域名与最多的类别并不“充分地相似”。

这里，如果)<，其中=0.98，我们认为该NXDomain簇与最多的标签类不够相似，反而我们将它标记为“new DGA”并将它送到DGA建模模块。另一方面，如果)≥，我们确认该簇是最多的标签的DGA(如DGA-Conficker.C)产生的，并且不会再去考虑它。

上述阈值选择的原因将会在7.2节介绍。

**5** DGA分类及C&C检测

一旦DGA发现模块报出新的DGA，我们使用一个监督学习的方法，通过分析它生成的NXDomains集合来找出受到有关基于DGA的恶意程序感染的主机。为了找出感染的主机，我们收集主机生成的NXDomains集合，接着我们向DGA分类器“询问”是否看起来像由已知DGA产生的。如果是，就被认为是感染主机并且被标记为(可疑)DGA-bot。

此外，我们目标是构建一个可以分析活跃域名的分类器，感染主机访问的活跃域名集合表示为，我们提取看起来像上的DGA算法生成的C&C域名集合，这里。最终，我们可能人工检查集合，从而确认C&C域名和相关IP的身份。反过来，C&C IP列表可以添加到IP黑名单中，进而阻止C&C通讯和恶意程序感染的影响。接下来我们会详细介绍DGA分类模块和C&C检测模块。

**5.1** DGA建模

如4.2节提到的，那些DGA过滤模块输出的并且没有匹配到任何已知DGA模型的NXDomains簇将会(自动地)赋予一个New-DGA-vX标记，其中X是一个唯一标识符。此时我们建立两个不同的静态模型代表New-DGA-vX：(1)一个可以为产生的NXDomains集合赋予特定DGA标记的多类统计分类器；(2)一个可以计算查询过的单一活跃域名是C&C域名的可能性的隐马尔科夫模型(HMM)，通过它我们可以得到C&C域名的可疑列表。

DGA建模模块将以下信息作为输入：(1)从alexa.com前10000个域名中提取的常用合法域名列表；(2)沙箱环境下由已知DGA-bot产生的NXDomains簇(见第6节)；(3)从DGA发现模块中收集到的NXDomains簇。令*NX*为这样的新发现的NXDomains簇。因为在一些情况下，*NX*只包含相对较少的域名，所以我们尝试将*NX*扩大到更大的帮助我们建立更好的新DGA的统计模型。为了达到这一目的，我们将产生NX中所有NXDomains的主机从稀疏关联矩阵M中找出来，并且我们收集它们在一段时间内产生的所有NXDomains。例如，主机产生了一些在*NX*中的域名，我们收集产生的其他NXDomains 。接着我们将该集合加入到训练集中(标记为合适的新DGA)，即=。读者在这里可能注意到集合不仅包括了由于DGA产生的NXDomains，它也包括主机“偶然”产生的NXDomains。因此，这会在训练集中引入一些噪音。但是这样偶然产生的NXDomains和DGA产生的比显得微不足道。因此，我们利用统计机器学习算法的泛化能力来消除潜在的噪声源的影响。这种方法在实践中效果很好，我们将在第7节说明。

**5.2** DGA分类器

DGA分类器基于多类的决策树算法(ADT)。ADT利用了Boosting的高分类精度，它产生紧凑的分类规则，可以更容易被解释。

为了检测出感染了DGA恶意程序的主机，我们监视受控网络下每个主机产生的NXDomains并定期将这些信息传入DGA分类器。给定主机产生的NXDomains集合，我们将分割成长度为的子集，每个子集我们提取在4.1.1节描述的统计特征。如果其中的一个子集被DGA分类器标记为是由给定的DGA产生的，我们就将标记感染主机并且将它的IP地址和DGA标签记入恶意程序检测报告中。

**5.3** C&C检测

C&C检测模块基于隐马尔科夫模型(HMM)。每一个DGA我们使用一个不同的HMM。给定一个DGA D生成的集合，我们分别考虑中的每一个域名*d*，将这些域名输入一个HMM中进行训练。HMM将这些域名试作字符序列，训练的结果是一个隐马尔科夫模型，对于单一域名*s*，它会输出*s*由D生成的相似性。

我们使用从左至右的隐马尔科夫模型，它们在实际中被用来降低模型的复杂度，有效地缓解下拟合相关问题。HMM的输出符号是域名中允许出现的字符(也就是字母、数字、“\_”、“-”、和“.”)。我们设置隐藏状态的个数等同于训练集中域名的平均长度。

在操作中，C&C检测模块接收感染主机查询的活跃域名。令为这样的主机，D为上运行的DGA算法。C&C检测模块会将每个查询过的域名s送入中，它会计算一个相似性得分*f*(*s*)。如果*f*(*s*)>，s就被标记为DGA D的C&C域名候选域名。

阈值可以在训练阶段学习到。首先我们使用集合训练HMM。接着我们使用从Alexa获取到的一个非DGA的合法域名集合*L*。对于域名，我们计算相似度*f*(*l*)并且设定得到最大假阳性率(如 最大FPs=1%)的阈值。

**6** 数据收集

在这一节我们介绍了连续15个月来(实验阶段)观察到的NXDomain流量大小的一个概述，时间始于2010年11月1日，结束于2012年1月15日。之后，我们讨论了如何收集用于训练的域名并测试了我们的DGA分类器(见第5节)。

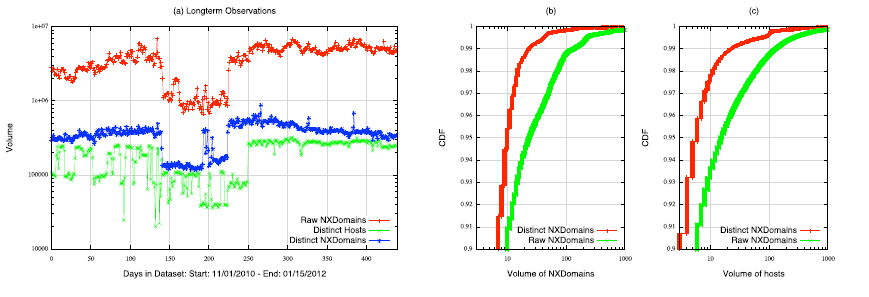


图2 在一组ISP下的递归DNS服务器观察439天的NXDomains流量

**6.1** NXDomain流量

我们通过监控一个大的北美ISP的一组递归DNS服务器的DNS进出报文得到的15个月的DNS流量评估了Pleiades。这些服务器都位于美国，并且为200万用户提供服务。我们的监视节点在DNS服务器下，因此可以看见每个客户端生成的NXDomains。

图2(a)展示了每天(1)DNS流量中NXDomains的个数，(2)认为在一天内查询过至少一个NXDomains的不同主机的个数，(3)不同(去重)NXDomains的个数(我们也过滤掉那些没有有效TLD的域名)。2011年3月24日至6月17日的NXDomains和主机突然的下降(大约减少了30%)的原因是ISP网络配置改变。

平均每天我们观察到了500万(条)NXDomains，187600个至少查询过一次的不重复的主机以及360700个不重复的NXDomains。因此，用于谱聚类(见4.1.3节)的关联矩阵M的平均大小是187600360700。但是值得注意的是M是稀疏的并且可以高效地存储在内存中。事实上，大多数(大约90%)的主机每天查询10个以下的NXDomains，因此M的大多数行只有一些少数的非0元素。图2(b)展示了监控网络下一个主机访问NXDomains量的累积分布函数(CDF)。另一方面，图2(c)展示了查询一个NXDomain的主机个数的CDF(这直接关系到M的稀疏性(依据它的列))。

**6.2** 参考标准

为了生成训练的参考标准以及评估DGA分类器(第5节)，我们使用一个简单的方法。为了收集已知DGA恶意程序产生的NXDomains，我们使用两种不同的方法。首先，因为DGA的很多变种是已知的(逆向工程提取得来)，我们使用这些算法产生这些僵尸网络域名的集合。为了得到Bobax和Sinowal这些DGA算法未知的域名，我们在一个虚拟机里运行这两个恶意软件样本并且只允许DNS流量通过，拒绝其他流量。总共我们收集了Conficker生成的30000个域名，Murofet的26078个域名，Bobax的1283个域名以及Sinowal的1783个域名。

最后，我们使用alexa.com前10000个带和不带www前缀的常用域名。因此，我们在训练和测试DGA分类器时有了20000个反例(非DGA)。

**7** 分析

本节我们给出系统的实验结果。开始我们展示Pleiades对已知DGA(如Conficker，Sinowal，Bobax和Murofet)的建模准确度。接着我们详细介绍了在对NXDomains的15个月的监控中发现的DGA。最后我们总结了检测到的12个DGA中的有趣的发现。它们中的一半使用了已知恶意程序家族的DGA算法，另一半，据我们所知，没有已知的恶意程序关联。

**7.1** DGA分类器的检测结果

本节我们给出DGA分类器的精度。我们使用Bobax, Sinowal, Conficker-A, Conficker-B,Conficker-C和Murofet产生的NXDomains来引导分类器。我们在两种模式中测试分类器。第一种模式使用Conficker的一个“超类”，它由Conficker-A, Conficker-B和Conficker-C的相同数目的样本组成。另一种模式则将Conficker的每个变种作为不同的类别看待。正如我们在5.2节提到的，DGA分类器基于多类决策树算法。我们通过收集一天中蜜罐的NXDomains(如果是Sinowal和Bobax)和一天中Conficker-A, Conficker-B, Conficker-C和Murofet产生的NXDomains来构建向量。最后，我们使用Alexa前10000的带和不带www域名作为良性域名。

从每个类的域名中，我们随机选取了大小为的3000个集合。注意，我们使用的值有2、5、10和30。这是建立不同的训练集的过程，目的就是通过实际证明哪个值可以给出DGA模型之间的最佳的分割。

我们生成额外的测试数据集。这种情况下我们使用的域名来自不同日期的训练集中的每个类别。我们这样做，可以得到训练和测试数据集之间的最短域名的重叠。我们使用两种方法来评估训练集：**训练集上**的10折交叉验证和使用**不同日子**收集到的域名的测试集。这两种方法给了我们非常相似的结果。我们的系统在10折交叉验证的情况下表现最差，因此我们选择它作为最坏的情况。

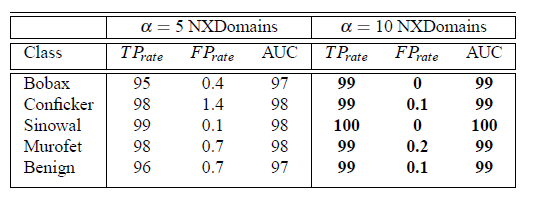


表1 不同下使用折交叉验证的检测结果(单位 %)

在表1中我们可以看到等于5和10时的检测结果。由于空间限制，我们没有给出其他值的结果。我们在数据集中观察到了类别间的混淆，特别是Conficker-A和Conficker-B之间。为了解决这一问题，我们创建了一个通用的Conficker类，它由每个Conficker变种的相同个数的向量构成。这种对Conficker变种的合并使得DGA分类器的正确率提高到了99.72%(判断正确：7986，错误：22)。使用这5种DGA作为数据集，平均和分别为99.7%和0.1%。我们可以在表1中看到，当时表现得不错，但是较高。

**7.2** NXDomains聚类结果

本节我们将讨论DGA发现模块的结果。特别地，我们详细说明在研究期间的阈值的选择过程，聚类的鉴别过程以及DGA发现模块产生的错误告警。

7.2.1 相关阈值

为了设定在4.2节中出现的阈值和，我们使用2010年11月的前5天标记了的213个聚簇，这里有DGA的正例和反例。在这个实验中，我们包括除了那些过滤掉的与已知DGA“相似”的的所有的聚簇。图3中，我们可以看出y轴代表五天内每个聚簇产生的显性（非良性）类的百分比。在x轴我们可以看出每个聚簇内显性类的方差。结果表明，正例和反例间有一个明确的界限，通过它我们可以设定阈值，。在整个的实验过程中，这些阈值给我们提供了非常好的结果。我们会在7.2.3节中讨论，DGA发现模块在15个月中只错误报出了5个良性聚簇。所有误报聚簇的方差都接近0.001。

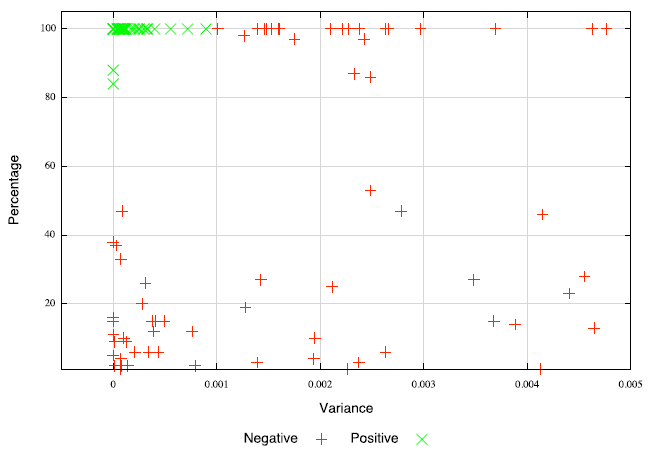


图3 2010年11月的前5天的和

7.2.2 新DGA

Pleiades从2010年11月的第一天就开始对NXDomains流量进行聚类。我们使用已知的DGA作为正例和一组Alexa域名作为反例来引导DGA建模过程。在表2中，我们展示了在实验过程中发现的独特的聚簇。“Malware Family”列将变种关联到已知的恶意程序家族上(如果有的话)。我们通过检查NXDomains与从恶意程序库中提取到的NXDomains的重叠关系来发现恶意程序家族。另外我们也在威胁情报公司的帮助下手动地检查了聚簇。“First Seen”列表示我们从流量中第一次发现该DGA变种的时间。最后，“Population on Discovery”列表示发现那天的变种的感染数目。我们可以看到，我们可以检测出每个DGA变种有着平均32个感染主机，这些主机横跨全州的ISP网络。

正如我们在表2中看到的，Pleiades报出了6个已知DGA家族的变种。其他6个NXDomains变种也被Pleiades报了出来，但是据我们所知，没有已知的恶意程序与他们相关联。图4展示了每一个变种中的10个域名样本。

在我们15个月的观察期内，我们平均每天在ISP网络下发现742个Conficker的感染主机。Murofet第二，每天平均有92个，Boonana DGA第三，每天平均84个。增长最快的DGA是Zeus v3，它平均每天有50个感染主机，但是实验最后四天Zeus v3DGA平均每天有134台感染主机。值得注意的是New-DGA-v1平均每天有19个主机，是新发现的DGA中个数最多的。

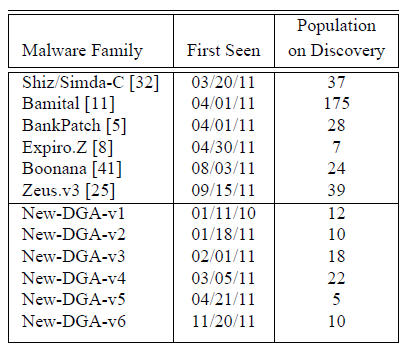


表2 Pleiades发现的DGA

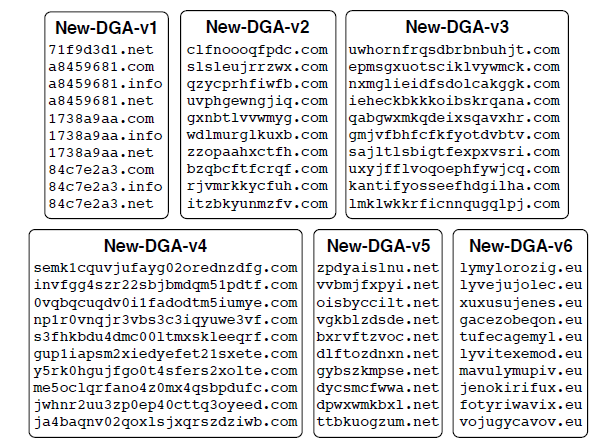


图4 那些与已知恶意程序家族无关的DGA聚簇的10个NXDomains样本

7.2.3 对新DGA的误报

在我们的评测期间我们遇到了5起DGA聚簇的误报。在所有的情况下，我们将这些类别在DGA建模器中看作是良性域名的变种。现在我们详细讨论每种情况。

第一个NXDomains簇的误报来自于Chrome产生的随机域名。当Chrome浏览器启动时，它会查询3个随机的域名。这些域名起到一个DNS检查的作用，浏览器可以判断NXDomain的重写模块是否已启用。Pleiades将“Chrome DGA”判断为了Bobax的变种。我们为该“DGA”训练了一个类并且把它标记为良性。NXDomain重写测试的情况出现在一个品牌的无线接入点中。Connectify提供了无线热点的功能并且它其中的一个配置选项就是允许用户劫持ISP的默认NXDomain重写服务。该设备生成固定数目的NXDomains用来测试重写。

另外两个误报是由.it和.edu的TLD产生的。这些域名包括了对一些常用单词(例如repubblica, gazzetta, computer)做的微小的变化。匹配到这些聚簇的域名在监控中只出现了两天并且再也没出现过。这两个聚簇如此短命的原因可能是域名曾经是垃圾邮件活动的一部分，而在垃圾邮件活跃之前就已经被当局制止。

第5个误报来自于美国政府下的域名并且包含字符串wpdhsmp。对此最好的猜测就是这些是内部域名，偶然间泄漏到我们ISP的递归DNS服务器上。该簇的域名只出现了一天。这一类NXDomains也被标记为良性变种。值得注意的是除了Chrome簇，所有误报的DGA聚簇都很短命。如果运营商愿意等几天直到Pleiades报出新的DGA簇，这些误报就不会出现。

**7.3** C&C检测

为了评估C&C检测的有效性，我们做了如下操作。我们考虑可以归因于特定恶意程序的6个新发现的DGA，如表3所示。令为在DGA发现(第4节)和DGA建模(5.1节)模块中第i个DGA产生的NXDomains集合。对于每个DGA，我们提取一个NXDomains子集用来训练模型。接着我们使用剩下的=-来计算的真正(TP)率，使用从alexa.com得到的与热门网站相关的602969个独特的域名的集合A来计算假正(FP)率。为了得到A，我们首先考虑一年内稳居alexa.com排行榜前10万的域名。这样我们得到了大约6万个稳定的热门域名集合T，这些域名我们认为是合法的。接着，我们监测几小时内大型网络下成功解析了的DNS记录，我们将满足T中2LD的所有域名也添加进A中去。

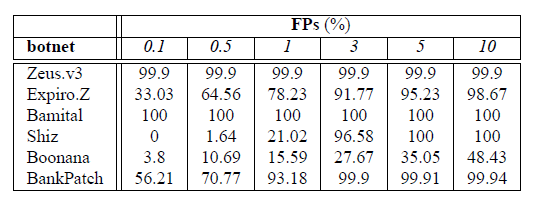


表3 C&C检测的真正率(1000个测试序列)

我们以不同个数的训练样本进行实验。具体来说，我们令c=100，200,500，1000,2000，5000和10000。接着对于不同的检测阈值，我们在TPs和FPs间寻找最优的方案。由于空间原因，我们在表3中只给出了c=1000的实验结果。一般情况下，随着训练实例的增加结果也会增加。我们设定检测阈值以便使得假正率等于0.1%，0.5%，1%，3%，5%和10%。我们可以看到，FP=1%时我们可以在6个DGA中的3个中得到一个很高的(>93%)真正率，6个中的5个得到相对较好(>78%)的真正率。FP=3%时6个中的5个的真正率都很高(>91%)。

如第3节提到的，C&C检测模块减少了主机h查询的成功解析了的域名的个数，它将受DGA感染的主机访问的活跃域名缩小到由DGA产生的C&C域名中去。表3的结果说明了如果我们依据HMM给出的相似性得分来对h查询过的域名进行一个排名的话，在多数情况下我们只需要检查h访问过的活跃域名的来检测C&C。

**7.4** 案例研究

7.4.1 Zeus.v3

在2011年9月，Pleiades检测到了一个新的DGA，几个星期后我们将它视为Zeus.v3变种。从感染主机中收集到的域名主要分布于6个不同的TLD中：.biz,.com,.info ,.net ,.org and和.ru. 不算顶级域名，该DGA产生的域名长度介于33~45之间。通过分析其中的一个恶意样本我们发现它的C&C基础设施是基于P2P的。如果恶意程序与P2P C&C网络连接失败，它就启动应急方案，此时一个DGA组件就开始尝试恢复断开了的C&C通讯。恶意程序将查询那些伪随机的域名，直到发现活跃的C&C域名为止。

到目前为止，我们发现了12个这样的域名。随着时间的推移，这12个C&C域名指向5个不同的C&C IP，它们分布在4个不同的网络下，3个在美国(AS6245, AS16626 and AS3595)，一个在英国(AS24931)。有趣的是我们发现英国那个C&C IP只活跃了非常短的几分钟，从东部标准时间的2012年1月25日的12:14:04到12:22:37。该C&C从美国(AS16626)转移到英国(AS24931)，然后迅速转移回了美国(AS3595)。

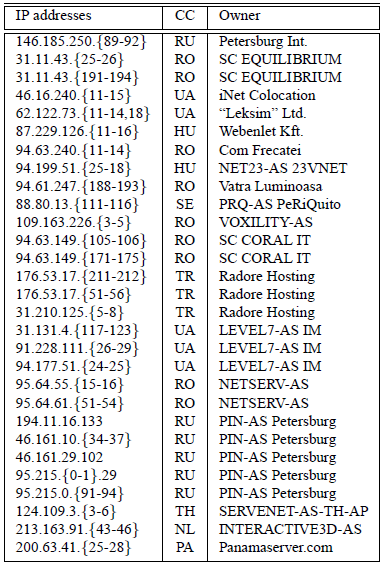


表4 BankPatch的C&C设施

7.4.1 BankPatch

我们选择BankPatch DGA作为样本进行分析，是因为这个僵尸网络在我们的实验过程中已经活动了几个月，并且感染数目很大。支持该僵尸网络的C&C设施令人印象深刻。26个不同的服务器集群充当该僵尸网络的C&C。僵尸网络的控制者不仅充分利用了DGA，同时平均每几周就将C&C迁移至不同的网络下。在我们C&C检测阶段，我们观察到欧洲CERT的IP地址。该CERT接管了这个僵尸网络的几个月的域名。我们尝试从多个角度去验证这些C&C设施的完整性和正确性。有关这些C&C的完整信息见表4。

该DGA的域名结构可以分为4字节的前缀以及一个字符串后缀。我们观察到的字符串后缀有seapollo.com, tomvader.com, aulmala.com, apontis.com, fnomosk.com, erhogeld.com, erobots.com, ndsontex.com, rtehedel.com, nconnect.com, edsafe.com, berhogeld.com, musallied.com, newnacion.com, susaname.com, tvolveras.com 和 dminmont.com。

前缀提供了该DGA的四字节的熵。我们观察不同日期的NXDomains之间的冲突，特别是只有一个后缀参数活跃的时候。因此，我们在2012年注册了10个域名作为一小组样本，希望能窥见僵尸网络的全貌。在为期一个月的对该DGA域名sinkhole数据的监控中，该僵尸网络总共感染了25个国家的270个不同的网络。通过观察我们sinkhole域名的递归DNS服务器信息，我们得知其中有4295个在美国。我们监控的递归服务器也在其中，并且我们可以在监控网络中发现平均86台感染主机。对sinkhole域名做DNS查询的前5个国家分别是美国、日本、加拿大、英国和新加坡。这些国家平均访问我们授权DNS服务器的递归DNS服务器的个数为22——相比于在美国的递归DNS服务器，这已经算非常少了。

**8** 局限与讨论

Pleiades有一些局限。例如，一旦发现一个新的DGA，Pleiades可以很准确地为它建立统计模型，它可以知道这种DGA产生的域名“长什么样”，但它不能通过学习重现它的域名生成算法。因此，Pleiades会产生一定程度的假正和假负。但是表1的数据表明，Pleiades可以建立一个很准确的DGA分类器模型，当时假正和假负都很低。与此同时，表3展示的C&C检测模块可以判断单一活跃域名的DGA种类，在大多数情况下表现很好。但是有一些情况下基于HMM的分类有些问题。我们认为这样的原因是HMM只考虑了域名的单一字符序列。在我们未来的工作中，我们计划尝试2-grams，即域名将被看成是一对字符的序列，从而对那些难以建模的DGA达到一个更好的分类精度。

例如，我们的HMM检测器对Boonana DGA不能达到一个较高的真正(TP)率。原因是Boonana DGA使用动态DNS提供商提供的在一些二级域名下的三级伪随机域名。在我们的评估阶段，感染了Boonana的主机联系了59个不同的有效二级域名的DGA。我们认为该DGA的三级域名的多变性以及如此多的二级域名的个数使得我们很难建立一个优秀的HMM模型，因此也导致了相对较低的真正率。然而在现实的部署情况下，通过关注感染主机查询的动态DNS，真正率会显著增加。例如，自从我们得知Boonana只使用动态DNS域名，我们可以过滤掉其他NXDomains，并不让他们输入到HMM中去。在这种情况下HMM只将动态DNS作为输入，这些域名只是每个主机查询过的活跃域名的一部分，因此可以显著降低误报的绝对数量。

正如我们在第3节中提到的，检测DGA生成的C&C域名是有意义的，因为它们解析的IP可以用于更新C&C IP黑名单。反过来，这个IP黑名单可以在网络边界封堵C&C通信，因此提供了一种方式来削弱僵尸网络的恶意活动。显然，这一策略如果成功，C&C IP地址的变化频率就会低于DGA生成的新的伪随机域名的频率。这个假设适用于各种在实际中遇到的基于DGA的恶意程序。毕竟生成伪随机域名的目的就是使得松散而又集中的僵尸网络更难被取缔。但是，人们可以想象“混合”的僵尸网络，它使用DGA生成的域名来发现IP对集合，从而进入基于P2P的C&C设施。或者DGA生成的C&C域名可能是速变域名，即这些域名指向一个IP-fluxing网络。值得注意的是这样复杂的“混合”僵尸网络很难成功地去开发、部署和管理。

另一个潜在的局限基于这样一个事实：Pleiades不能辨认出使用同一种DGA算法的不同僵尸网络。在这种情况下，两个僵尸网络可能被不同的实体所控制，Pleiades会将监控网络下的这些感染主机归为一个DGA僵尸网络。

我们评价方法的一个局限是不能准确地计算ISP网络下感染主机的个数。由于我们监控节点设置的原因(递归DNS服务器下)，我们只能得到感染主机数目的一个最小值。因为我们是从ISP的DNS流量的角度来观察IP地址的，但我们不清楚每个IP背后到底有多少个主机。例如，一个产生DNS流量的IP地址很有可能是NAT、防火墙、DNS服务器或者其他充当代理或中继的复杂的设备。另外，ISP的DHCP流失率相对较低，因此我们不太可能对同一内部主机进行多次计数。

在Zeus.v3的案例中，DGA作为当P2P组件与C&C建立通讯信道失败时的一个寻找C&C机器的备选方案。这种将DGA组件作为冗余的发现C&C的方法在未来可能被其他恶意程序所利用。大量的新DGA可能会潜在地对Pleiades的监督模块产生负面影响，特别是基于HMM的C&C检测模块。事实上，由于我们需要辨认出大量的类别，DGA分类器产生的错误分类可能使得C&C检测中选择错误的HMM，从而导致了假正(FP)的增长。在我们未来的工作中计划估计这种错误分类对C&C检测精度的影响，并且调研在这种情况下使用IP的辅助信息(例如 IP信誉)是否可以显著地提高精度。

随着我们内部系统原理的公开，一些僵尸网络可能会试图躲避DGA的发现和C&C检测过程。正如我们讨论的那样，恶意软件的作者们希望创造大量的与DGA相关的NXDomains从而使得僵尸网络取缔更加困难。恶意程序可以在同一时间生成与发现C&C服务器无关的NXDomains从而误导我们现在实现了的Pleiades版本。这些“嘈杂”的NXDomains可能由两种方法产生：(1)随机，例如再使用一个不同的DGA；(2)使用同一种DGA，但是使用两套种子，其中一套用来产生噪声。对于第(1)种情况，它们聚类在一起的可能性很小。这意味着这些NXDomains最后不会成为最终聚类关联过程的一部分，它们也不会被看成是新的DGA簇。另一种情况(2)可能在学习过程中是一个麻烦，特别是HMM，因为真正的NXDomains和噪声会在一个聚簇中混在一起，从而使得学习一个准确域名模型变得更加困难。

**9** 总结

本文中我们提出了一种新型的检测系统Pleiades，它可以在一个监控的网络下精准地检测出感染了DGA僵尸网络的主机。Pleiades通过在递归DNS服务器下监控流量并且对解析不成功的DNS查询进行分析，而不是依赖于对僵尸程序和它们DGA算法的手动逆向。通过十几个月的实验，我们证明了Pleiades可以达到一个很高的精确度。此外，Pleiades部署在大型ISP网络下的这15个月来，它可以发现6个属于已知恶意程序家族的DGA和6个之前从未报出过的DGA。