doi:10.11959/j.issn.1000-436x.2015193

基于攻击图的多源告警关联分析方法

刘威歆，郑康锋，武斌，杨义先
(北京邮电大学 信息安全中心，北京 100876)

摘 要：现有基于攻击图(attack graph)的告警关联分析方法难以全面处理告警关联关系，同时，漏报推断和告警预测带来大量冗余路径误报。针对以上问题提出了基于攻击图的多源告警关联分析算法，能够综合应用图关系和阈值限制进行联动推断和预测，达到更为全面解决攻击图中的告警漏报和减少误报数量的目的。同时，将告警处理算法并行化，提出了 AG-PAP 告警并行处理引擎。实验表明，该方法能够提升关联分析的有效性和性能表现。
关键词：告警关联；攻击图；多源分析；并行处理
中图分类号：TP302 **文献标识码**：A

Alert processing based on attack graph and multi-source analyzing

LIU Wei-xin, ZHENG Kang-feng, WU Bin, YANG Yi-xian
(Information Security Centre, Beijing University of Posts and Telecommunications, Beijing 100876, China)

Abstract: Current attack graph-based alert correlation cannot deal with graph relation between alerts properly, and a large number of redundant attack paths may arise when trying to find out missing alerts and predict future attacks. A multi-source alert analyzing method was proposed, fully utilizing graph relation and threshold to correlate mapped alerts and eventually reduce false positive rate as well as true negative rate. To improve the speed of the algorithm, a parallel alert processing system (AG-PAP) was proposed. AG-PAP is tested on distributed environment which gets satisfied effectiveness and performance.
Key words: alert correlation; attack graph; multi-source analyzing; parallel processing

1 引言

随着计算机安全成为人们迫切需要解决的威胁问题，告警关联的重要性吸引了越来越多的机构和研究者进行深入研究。在告警分析框架方面，Valeur 等[1]提出了一个通用的 pipeline 式告警处理流程，包含 10 个处理阶段，如标准化、告警融合、多步关联和影响分析等，最终输出 Intrusion Reports。Sebastian Roschke 等[2]提出了一个基于列式数据库, 告警内存存储和内存索引表的高效而灵活的分布式 IDS 告警关联平台，能够应用于不同的 IDS 探针和管理系统。梅海彬等[3]提出了基于警报序列聚类的多步攻击模式发现方法，无需依赖大量先验知识，易用性和有效性强。

在告警关系提取方面，攻击图(attack graph)成为了一个很重要的研究方向[4,5]。攻击图是一个强大的分析工具，能够通过汇聚网络中的网络可达性信息、漏洞信息和主机信息，描述网络中的主机之间的网络关系、服务关系和主机上漏洞的相互利用关系。Sushil Jajodia 等提出了 TVA(topological vulnerability analysis)[6]，此系统包含网络结构和全局漏洞信息，能够模拟渐进的网络渗透，建立完整的攻击场景图，同时能够显示所有网络中的可能路径。Ou 等[7]提出了利用 MulVal 来进行多主机网络中多步骤攻击的推导，并生成攻击图。

在基于攻击图的告警处理方面，第一个主要问题是对攻击图中的因果相关性的挖掘，现有方法往往是使用单一告警进行因果关系的推断，面临的挑战是攻击者日渐使用更为复杂的攻击手段进行系

收稿日期：2014-10-28；**修回日期**：2015-02-10
基金项目：国家自然科学基金资助项目（61101108）
Foundation Item: The National Natural Science Foundation of China (61101108)

统的入侵，单源的推断难以应对复杂的多步攻击且不能对告警的因果性做全面的挖掘；第二是由攻击图内部强相关性和数据规模带来分析效率问题。攻击图规模庞大，需要对图数据进行拆分以利并行，现有做法往往是把攻击图拆分成路径集合来减弱匹配知识数据的依赖性，但这种做法会给告警分析大量的复制开销，且难以应对攻击图的局部更新。同时，图计算往往是迭代计算[8]，传统的批量式并行架构需要以连续任务达到迭代效果，增加了大量的任务启停开销，因而攻击图的实时计算需要更为适合的迭代式并行处理方法。

基于上述分析，本文提出了基于攻击图的多源告警关联告警分析方法，综合应用告警的图关联关系进行联动推断和预测，通过阈值限制分析范围，从而进一步提升多步攻击和关联告警的挖掘能力以及降低误报率。同时，提出了告警并行处理引擎，将批处理式告警映射并行和迭代式告警分析并行结合，提高了运行效率。最后用实验数据验证分析处理的准确性，同时测试分析告警并行处理引擎的性能表现。

2 相关工作

目前，基于攻击图的告警关联分析有以下代表性研究：Wang 等[9]首次提出基于队列图的攻击图匹配结构，采用 BFS 树进行前件节点和后件节点的遍历，终止的条件是找到被其他告警激活的节点，仅能应对告警的直接前件和后件，且没有进行阈值限制，难以全面解决漏报且增加了无关路径的遍历开销。Sayed 等[10]提出了一种基于攻击图的混合告警关联模型，能够关联攻击图已知的告警，同时能够用相似性比较的方法关联攻击图未知告警，并能更新攻击图，但是对所有未直接关联的告警都会在阈值内向上深度搜索，能够解决前件告警漏报的问题且限制了误报的个数，但是不能解决后件漏报的问题。上述基于攻击图的告警关联往往只考虑单一前件的级联和后件的级联，却往往忽略了综合多个告警之间相关性进行下一步的推断和预测，难以对因果关联关系进行全面的挖掘和解决告警漏报问题。

在攻击图的并行处理方面，Roschke 等[11,12]提出了一种分布式高性能的 IDS 关联系统，将告警映射、告警融合、告警依赖关系提取和最长攻击利用链分析并行化，依托于并行框架 OpenCL 或 MapReduce[13]，能够利用攻击图映射告警，生成告警依赖图，存在的问题是基于路径拆分的批量式处理方式难以应对攻击图结构在并行单元之间有效分配，同时文中也并未考虑攻击图中环路问题和提出具体的路径划分方法；其次局部图更新会带来大量路径的改变，难以应对现有云计算网络的动态变化。本文所采用的 GraphLab[14]并行处理框架是由 CMU 的 Select 实验室提出的，基于共享内存进行异步分布式图计算，是一个更为灵活的迭代式处理框架，能够嵌套 Pregel[15]式的处理流程。在并行处理速度上，GraphLab 也普遍优于其他的处理实现[16,17]，包括：Giraph[16,17]、Hadoop 和 YARN，因而能够很好地支撑攻击图的实时分析计算。

3 基于攻击图的多源告警关联分析

攻击图在告警分析中的优势就在于它能够很好地用图来展示不同主机的不同漏洞之间的逻辑因果关系。通过以多个告警为应证源能够更好地结合图中逻辑关系进行推断和预测。本文所提出的基于攻击图的多源告警关联分析是指在告警依据其属性映射到攻击的动作节点后，以其动作节点为中心向相邻节点发送前向预测消息和后向推断消息。收到消息的节点综合分析多个源节点发来的消息强度和消息方向，进行下一步的预测和推断消息发送。同时，通过控制消息强度，减少多余路径的分析开销，降低误报率。通过迭代的消息处理，达到推断漏报的前后件攻击和预测可能的下一步攻击，降低漏报率并提升预测可靠性。

3.1 基于攻击图的告警映射

根据常用的攻击图定义和攻击场景分析，可将基础攻击图定义如下。

定义 1 $AG=(A_{AG}, C_{AG}, E_{AG})$

1) A_{AG} 为攻击图的动作节点。

$A_{AG}=\{a_i|a_i=(imp, host, vul)\}$，其中，$imp$ 为攻击图节点 a_i 的 $impact$，$host$ 为 a_i 所在主机，vul 为 a_i 的漏洞信息。

2) C_{AG} 为攻击图的状态节点。

C_{AG} 为攻击图中的一系列状态节点，表示攻击动作会导致的后件安全状态或者能够导致供给动作成功执行的前件安全状态。

3) E_{AG} 为边集合。

$E_{AG}=\{e_i|e_i\in((A_{AG}\times C_{AG})\cup(C_{AG}\times A_{AG}))\}$, 攻击图是有向图，每个节点都有其出边和入边，一个节点的出边代表从自身（攻击）到后件攻击，而入边代表从前件攻击到自身（攻击）。

定义 2　告警格式定义为

$AL=Timstamp\times Host\times Port\times Host\times Port\times Class$

当 $al\in AL$ 时，$al=al(t,src,sp,dst,dp,c)$六元组。

1) $t\in Timstamp$ 为告警时间戳。

2) $src\in Host$ 为告警的源 IP。

3) $sp\in Port$ 为告警的源端口。

4) $dst\in Host$ 为告警的目的 IP。

5) $dp\in Port$ 为告警的目的端口。

6) $c\in Class$ 为告警的漏洞利用分类信息。

告警映射 $map(al)$依据的参数是告警的目标 IP 和告警对应的漏洞类型，将告警映射到攻击图中节点所在的动作节点 a，若映射失败则为空。

定义 3　告警映射定义为

$$map(al)\rightarrow\begin{cases} a:\exists v\in AG,(dst(al)=host(a))\wedge \\ (class(al)\ =\ vul(a)) \\ \phi:\nexists v\in AG,(dst(al)=host(a))\wedge \\ (class(al)\ =\ vul(a)) \end{cases}$$

3.2　基于攻击图的多源关联分析

基于攻击图的告警推断和预测主要包括 3 个方面：一是前件推断，即根据攻击告警推断攻击者执行此次攻击前的必要攻击步骤，能够解决前件告警的漏报和关联；二是后件预测，能够根据攻击告警推断攻击者执行此次攻击之后能够进行的攻击；三是前后件综合判断，Wang[9]和 Seyed[10]对前 2 种方面都进行了充分的考虑，但忽略了基于前后件的综合判断。当一个节点的某一前件和某一后件都被激活的时候，则很大程度上此节点代表的攻击已被攻击者成功执行，因而需要进行该节点其余前件与后件的分析。多源关联分析能够基于多源应证消息进行下一步的推断和预测工作能够较全面地解决已有告警的关联、漏报告警的修复和下一步攻击的预测。

本文基于攻击图提出了意图队列图(IQ_g)和意图消息($IMsg$)，IQ_g 基于队列图[9]和扩展队列图[10]，并加入推断强度和更多的状态进而根据前件推断和后件预测进行综合的处理。$IMsg$ 承载的是不同方向的激活、推断和预测消息。IQ_g 根据 $IMsg$ 进行自身状态的改变和下一步 $IMsg$ 的发送。

定义 4　$IQ_g=(V, E)$，其中

1) $V=\{Status, FwValue, BwValue, AlertInfo, AGData\}$。

① $Status=\{FALSE \mid HYP \mid HHYP \mid TRUE\}$，$FALSE$ 代表节点未被任何动作激活，HYP 代表节点被前向预测或后向推断消息激活，$HHYP$ 代表节点同时被前向预测或后向推断消息激活，$TRUE$ 代表节点被真实告警所激活。

② $FwValue$ 代表节点的最大前向预测强度。

③ $BwValue$ 代表节点的最大后向推断强度。

④ $AlertInfo$ 代表节点的最新激活告警信息，映射到相同节点的告警会覆盖更新节点的 $AlertInfo$，只有在 $Status$ 为 $True$ 的时候才非空。

⑤ $AGData=(A_{AG}\times C_{AG})$，$AGData$ 代表的是原有攻击图中的节点信息，包括动作节点 A_{AG} 和状态节点 C_{AG}。

2) $E=E_{AG}$，IQ_g 中的边和攻击图中的边相同。

定义 5　$IMsg$ 结构定义为

$IMsg=\{Direction, FwValue, BwValue, AlertInfo\}$。

1) $Direction=\{FW \mid BW \mid FBW \mid AL\}$，代表消息方向，包含 4 个值：$FW$ 代表前向预测消息，由节点的出边传出；BW 代表后向推断消息，由节点的入边传出；FBW 代表前后向消息，说明消息融合了前向预测消息和后向推断消息；AL 代表告警激活消息，说明存在映射到该节点的告警。

2) $FwValue$:前向预测强度。当 $Direction$ 为 FW 或 FBW 时，$fwvalue\in[1,ITH]$。当 $Direction$ 为 BW 或 AL 时，$fwvalue$ 为空。ITH 为消息强度阈值。

3) $BwValue$:后向预测强度。当 $Direction$ 为 BW 或 AL 时，$bwvalue\in[1,TH]$。当 $Direction$ 为 FW 或 FBW 时，$bwvalue$ 为空。

4) $AlertInfo$:告警信息。只有当 $Direction$ 为 AL，$alertinfo$ 才非空。

推断和预测的流程，伪代码见算法 1。

算法 1　告警关联流程

1)　**if** *exploit* activated by *alert*(case 1) **then**

2)　　infer all *pre-exploits* and predict all *post-exploits* within *ITH* levels

3)　**if** *exploit* inferred(case 2) **then**

4)　　continue to infer all *pre-exploits* within remaining levels

5)　**if** *exploit* predicted(case 3) **then**

6)　　continue to predict all *post-exploits* within

remaining levels

7) **if** *exploit* been both inferred and predicted (case 4) **then**

8) infer all *pre-exploits* and predict all *post-exploits* within *ITH* levels

情况 1 当 a_5 节点被告警 *alert* 激活，说明这个 *exploit* 被攻击者执行而且被检测到，启动在节点周围的 *ITH* 层内推断前件节点和预测后件节点，即 a_2、a_4、a_6 和 a_8。*ITH* 表示推断和预测的层数。每个节点所需的推断和预测的层数会随着推断和预测消息的传播逐跳递减，如 a_6 会继续发出强度为 *ITH*−1 的前向消息。

情况 2 当 a_2 被后件节点 a_5 所推断，继续在剩余的层数内推断前件节点 a_1。

情况 3 当 a_6 被前件节点 a_5 所预测，继续在剩余的层数内预测后件节点 a_7。

情况 4 当 a_5 同时被前件 a_2 和后件 a_8 所预测，则认为此节点代表的漏洞已被攻击者利用，需要从该点为中心重新进行推断前件 a_4 和预测后件 a_6。此时消息强度重置为 *ITH*。本文和 Sayed[10]方法的区别在于能够根据前件推断和后件预测进行下一步的迭代分析，在阈值相同的情况下能够分析双向的可达路径，减少漏报。

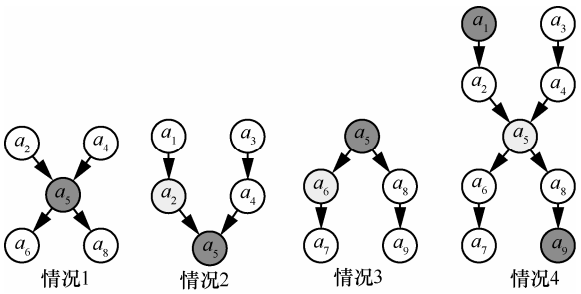

图 1 告警推断和预测

4 基于攻击图的告警处理并行化

本文第 3 节提出的多源关联分析方法把基于攻击图的告警关联问题转换成图节点消息处理问题，根据 GraphLab 中的 GAS(收集−应用−分发)[8, 19]编程模型能够很好地进行并行化处理。本文进而提出了基于攻击图的并行告警处理引擎 AG-PAP，能够很好地应对告警处理的不同并行需求，包括批量并行告警映射模块 AG-PM 和迭代并行告警分析模块 AG-GAS，将告警映射和告警的迭代分析的并行结合起来，形成完备的基于攻击图并行告警处理框架。

4.1 基于攻击图的并行告警处理系统结构

当 AG-PAP 引擎启动之后，会进行以下 3 个阶段的处理流程，如图 2 所示。

1) 并行告警映射(AG-PM)：负责将告警映射到攻击图节点 *v*，并通过本地内存交换和基于 TCP 的 RPC 交换同步给攻击图节点 *v* 的 master 所在的机器进行本地的消息处理。

2) 并行告警关联分析(AG-GAS)：首先在启动时通过分布式图划分策略分配各个处理单元需要读取的子图，存储于本地内存中。每个计算单元只会处理映射到本地 master 节点的告警，通过 MPI 和 Pthread 进行并行任务拆分和处理。通过多轮的迭代消息，更新攻击图节点的状态。

3) 结果图提取:在各个计算单元上，并行提取输出已激活的图节点以及相应边，形成结果图输出到各计算单元的本地硬盘或者 HDFS 上，方便网络管理员进行层次化的分析。

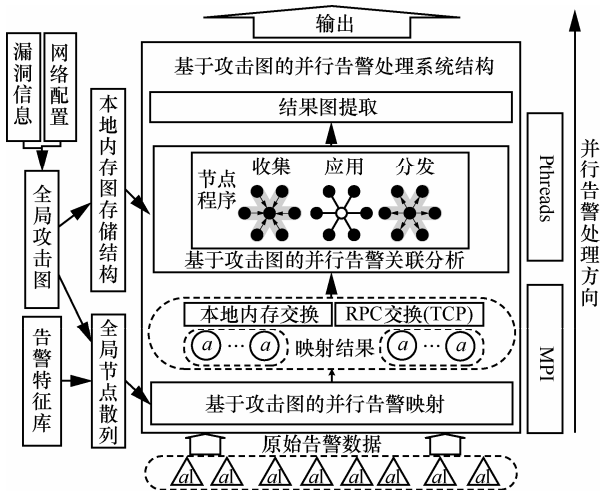

图 2 AG-PAP 引擎并行处理流程框架

4.2 基于攻击图的并行告警映射

告警映射并行化是依据告警相关属性和全局节点散列表，通过 *maptovertex*(*alert*)将告警映射到对应的攻击图动作节点，然后通过 *maptomachine*(*alert*,*vertex*)将非本地处理的告警同步到所在相应的 master 处理单元。本文和 Roschke[11]的映射方法不同在于攻击图节点的 master 只会存在其中一个处理单元上，因而本文每个告警在并行中的复制开销只会有一次；而后者需要把告警复制到所有包含其映射到的攻击图节点的路径上，可以看出本文所提出的并行映射方法的告警信息复制开销更小，更有

效率。

定义 6　并行告警映射(AG-PM)：

1) 原始告警映射到攻击图的动作节点

maptovertex(*alert*)→*vertex*=*map*(*alert*),

vertex∈*V*

2) 同步交换处理

maptomachine(*alert*,*vertex*)→*machine*,

machine∈*processing machine set*

4.3　基于攻击图的并行告警关联分析

并行告警分析(AG-GAS)流程如下。

1) 根据映射后的本地告警集合以消息形式通知相对应的攻击图动作节点。

2) 消息融合阶段：会对 4 种消息（告警激活消息、前向消息、后向消息和前后向消息）进行相互之间的融合。例如，前向（后向）消息之间的融合，方向不变，前向（后向）强度取较大值；前向和后向消息的融合成前后向消息，前向强度和后向强度均取较大值，如存在真实告警消息，忽略其他消息。

3) 初始化阶段：每个节点的节点程序接收合并过后的消息并缓存。

4) 收集阶段：在前向预测和后向推断中，跳过此阶段。只有当攻击图中产生新增节点时，才启动该阶段工作，新增动作节点根据周围节点状态决定自身状态和下一步的预测和推断。首先从入边拉取邻居节点的最大前向强度和从出边拉取邻居节点的最大后向强度，构造下一步前后向消息。若只有一个方向上的强度非零，则模拟消息方向与此方向相同；若 2 个方向上的强度都非零，则模拟消息的方向为前后向(FBW)；若 2 个方向上的强度都为 0，则不做任何处理，直接跳过之后的应用阶段和分发阶段。

5) 应用阶段：对接收到的消息缓存进行处理，更新自身状态。若节点未被前（后）向消息所激活，则会根据消息的前（后）向值更新自身的前后向值，并准备沿着消息方向进行下一次迭代，其中消息的强度衰减 1。若节点被前向与后向消息同时或依次激活，则认为该节点被执行的可能性极大，处理方式与被真实告警激活相同，需要以此节点为中心进行最大强度 ITH 的迭代消息处理。

6) 分发阶段：根据欲发送的消息和自身状态决定下一步的推断和预测。若为前向消息，则沿着所有出边发送。若为后向消息，则沿着所有入边发送。若为前后向消息，则沿着出边发送前向消息和沿着入边发送后向消息。

7) 结果图提取：即并行存储已更新的局部攻击图，从而提取出现有的攻击路径信息。若边两端的攻击图节点状态为 *TRUE*、*HYP*、*HHYP* 之一，则进行存储；若攻击图点状态为 *TRUE*、*HYP*、*HHYP* 之一，同样也进行存储。

具体算法的伪代码如下。

算法 2　消息融合

Input: Message *this*, Message *other*

Output: Message *merged_message*

1)　**if** *direction* of both are *FW* or *BW* or *FBW*(case 1)

2)　　**then** set *merged_message* to be with the same *direction*,the bigger *fwvalue* and *bwvalue*

3)　**if** *direction* of one is *FW* and the other is *BW*(case 2)

4)　　**then** set *merged_message* to be with *FBW* and the bigger *fwvalue* and *bwvalue*

5)　**if** *direction* of one is *FBW* and the other is not *AL*(case 3)

6)　　**then** set *merged_message* to be with *FBW* and the bigger *fwvalue* and *bwvalue*

7)　**if** *direction* of one is *AL* and the other is not *AL*(case 4)

8)　　**then** set *merged_message* to be the one with *AL*

9)　**if** both are *AL*(case 5)

10)　　**then** set *merged_message* to be the one with latest *alertinfo*

算法 3　分发

Input: a local-cached *sendmsg*, *EDGES*

Output: ouput *sendmsg*

1)　**if** Apply skipped **then** skip Scatter

2)　**if** *direction* of *sendmsg* is *FW* **then** send following *OUTEDGES*

3) **else if** *direction* of *sendmsg* is *BW* **then** send following *INEDGES*

4) **else if** *direction* of *sendmsg* is *FBW*

5)　　**then** change *direction* of *sendmsg* to *BW*

6)　　and send following *INEDGES*

7)　　change direction of *sendmsg* to *FW*

8)　　and send following *OUTEDGES*

算法 4 收集

Input: *vertex*, *EDGES*

Output: updated *vertex*,ouput *messages*

1) get the biggest fwvalue *b_fwvalue* from all adjacent vertexes following *INEDGES*

2) get the biggest bwvalue *b_bwvalue* from all adjacent vertexes following *OUTEDGES*

3) update itself and prepare *hyp_message* for Apply

4) **if** neither *b_fwvalue* or *b_bwvalue* is 0

5) **then** set *hyp_message* to *FBW* with *b_fwvalue* and *b_bwvalue*

6) **else if** only *b_fwvalue* is not 0

7) **then** set *hyp_message* to FW with *b_fwvalue*

8) **else if** only *b_bwvalue* is not 0

9) **then** set *hyp_message* to BW with *b_bwvalue*

10) **else if** *b_fwvalue* and *b_bwvalue* are both 0

11) **then** skip the following Apply and Scatter

算法 5 应用

Input: local-cached *message*, *vertex*

Output: the updated *vertex*

1)**if** direction of *message* is AL(case 1) then

2) **if** *status* is FALSE or HYP

3) **then** update *alertinfo*,set *vertex's fwvalue* and *bwvalue* to be *ITH*

4) change *status* to *TRUE*

5) build *sendmsg* with *FBW*

6) and *sendmsg.fwvalue=ITH*-1 and *sendmsg.bwvalue=ITH*-1

7) **else then** update *alertinfo* and set *status* to be *TRUE*

8)**if** *direction* of *message* is *FW*(case 2) **then**

9) **if** *message.fwvalue* is smaller than *vertex.fwvalue* then skip

10) **if** *vertex.bwvalue* is not 0

11) **then** set *status* to *HHYP*

12) set *vertex.fwvalue* and *vertex.bwvalue* to *ITH*

13) build *sendmsg* with *FBW* and

14) *sendmsg.fwvalue* and *bwvalue* to (*ITH*-1)

15) **else then** set *vertex.status* to be *HYP* and update *vertex.fwvalue*

16) and build *sendmsg* with *FW* and

17) *sendmsg.fwvalue* to (*ITH*-1)

18)**if** *direction* of *message* is *BW*(case 3) **then**

19) **if** *message.bwvalue* is smaller than *vertex.bwvalue* **then** skip

20)**if** *vertex.fwvalue* is not 0

21) **then** set *status* to *HHYP*

22) set *vertex.fwvalue* and *vertex.bwvalue* to *ITH*

23) build *sendmsg* with *FBW* and

24) *sendmsg.fwvalue* and *bwvalue* to (*ITH*-1)

25) **else then** set *vertex.status* to be *HYP* and update *vertex.bwvalue*

26) and build *sendmsg* with *BW* and

27) *sendmsg.fwvalue* to (*ITH*-1)

28)**if** *direction* of *message* is *FBW*(case 4) **then**

29) **if** *vertex.status* is *FBW* or *AL* **then** skip

30) **else then** set *vertex.status* to *HHYP*

31) set *vertex.fwvalue* and *bwvalue* to *ITH*

32) build *sendmsg* with *FBW* and *fwvalue* and *bwvalue* be *ITH*-1

5 实验分析

5.1 分析效果实验

本文首先搭建一个本地小型网络来测试预测分析的有效性，使用 5 台主机，其中包括一台 DMZ 主机 host1，内部网络中 host2 到 host5 都是存有不同资料的主机。每个主机上都有 3 个漏洞 ftp-rhost(CVE-2008-1396)、rsh login(CVE-1999-0180)和 local-setiud-bof (CVE-2006-3378)，攻击者能够通过 DMZ 主机 host1 进而攻击 host2 到 host5，为了简化攻击图，通过配置防火墙，内部网络中仅 host5 和 host3 之间存在网络可达关系。实验环境如图 3 所示，生成的攻击如图 4 所示。

为了检测多源关联分析的解决告警误报和漏报的能力，在 snort 上，本文只配置了 rsh 的检测规则，因而只能检测到 rsh 利用的相关告警。生成的结果如图 5 所示，可以看到处理引擎能够推断出阈值内的前件与后件，同时能够通过综合前后件关联提升中间的 *ftp_rhost*(1,2)、*ftp_rhost*(1,4)以及相关前后件状态的漏报强度，同时进行下一步的预测。

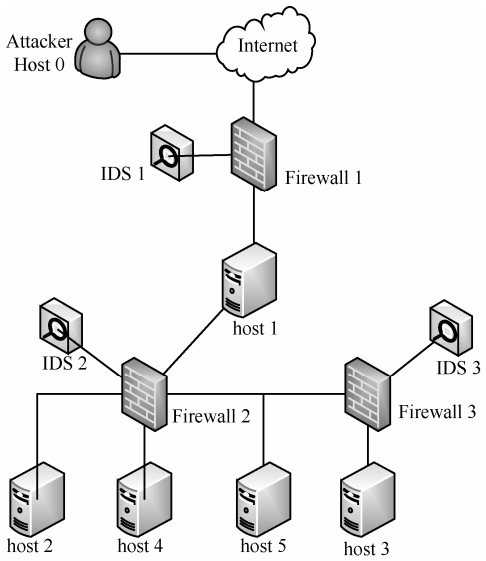

图 3　实验环境

同时，本文采用 Wang[9]和 Seyed[10]提出的攻击图关联方法进行了实验分析，Wang[9]在每个告警映射之后均会进行前后件的查找去寻找能够关联的告警，由于没有关联阈值的限制，会根据相关前件和相关后件搜索出所有的路径，但是攻击者不一定所有路径都会采用，这就带来了很大的无谓开销和误报。Seyed[10]的方法则会向上查询阈值内的前件告警，能够解决一定范围内的前件漏报，产生的结果图和攻击路径吻合，降低了误报数量，但是不能解决后件漏报与间接漏报，同时也没有预测能力。而本文提出的多源关联分析方法能够兼顾攻击图中的前后件漏报和间接漏报，同时通过阈值设置能够限制误报数量（由于本文是基于攻击图的分析，所以对未知攻击不做考虑，仅对已知攻击漏报进行处理）。表 1 则是解决漏报能力和误报数量的比较，可以看出在综合关联分析能力上优于 Wang[9]和 Seyed[10]提出的攻击图关联方法。

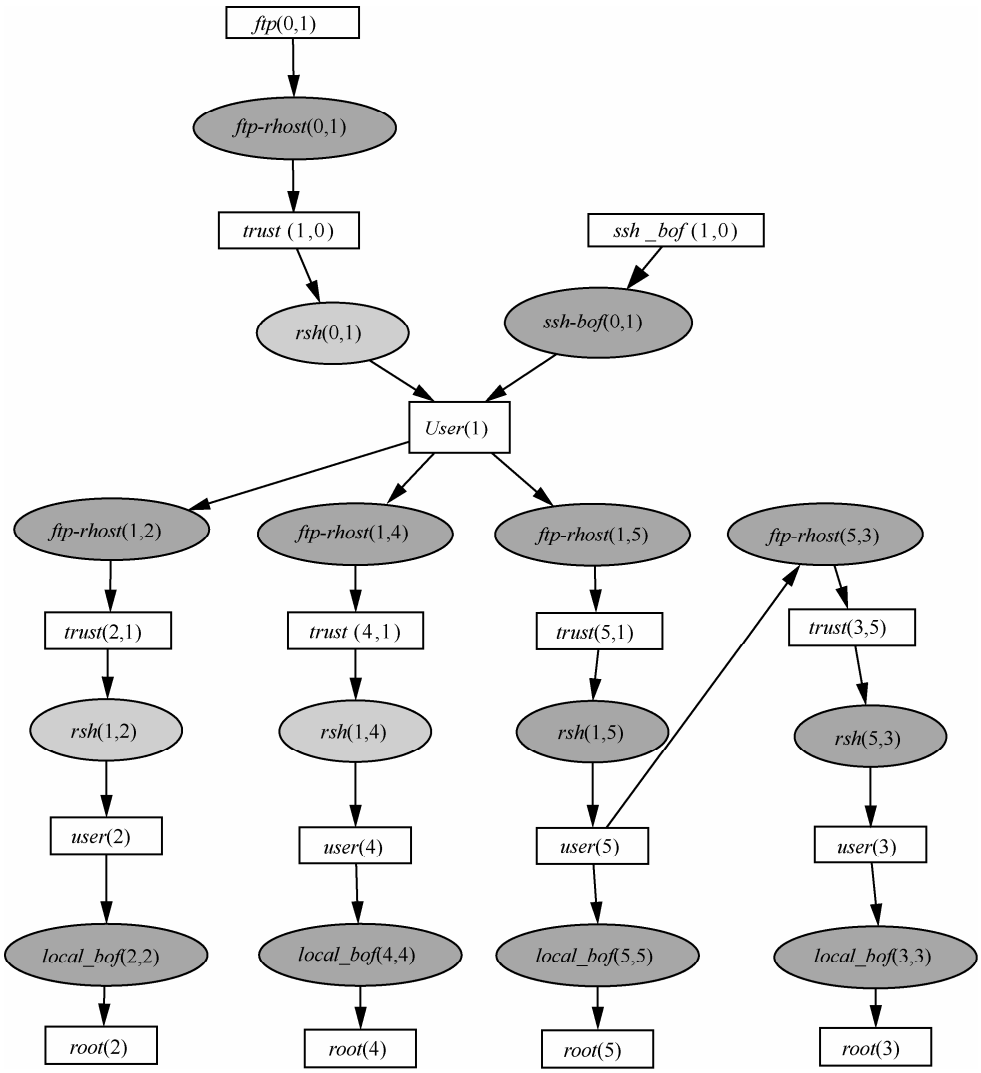

图 4　实验环境攻击

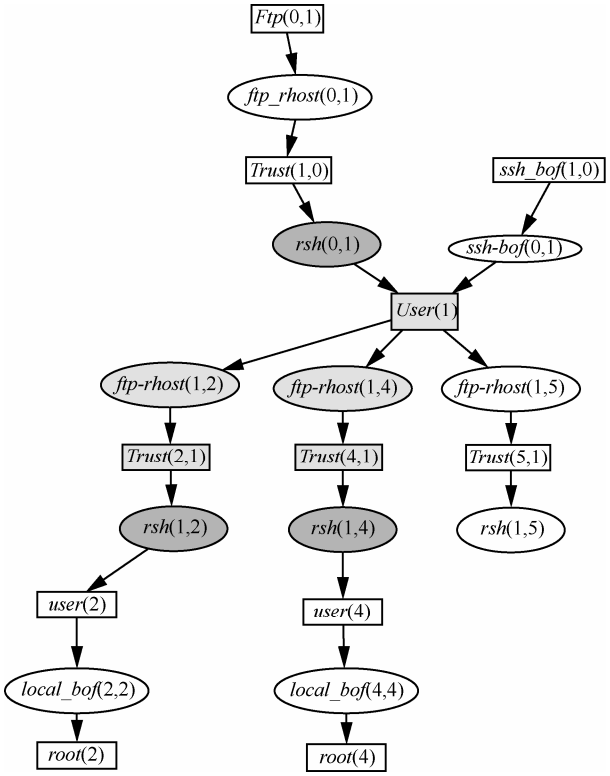

图 5　预测分析结果

表 1　　关联分析实验比较

方法	攻击数	映射告警	输出告警	漏报数	误报数
Wang	8	3	14	0	6
Seyed(*HT*=3)	8	3	7	2	1
本文(*ITH*=3)	8	3	11	0	3

通过增加实验网络中的主机数至 10 台，扩大攻击图规模至 72 个节点，然后对其中 host2、host3、host4 进行攻击，产生了 10 个映射告警。图 6 表明 Wang[9]的方法在更大规模的攻击图中会产生更多的误报，Seyed[10]的方法由于有阈值限制和仅进行前件搜索，在误报数量上表现更优，本文方法较 Wang[9]的方法在误报率上降低了近 40%，比 Seyed[10]的方法会多出后件预测带来的误报。

5.2　分布式性能分析

在并行分析环境的搭建上，本文在 IBM System x3650 m4(处理器 E5-2620，内存 64 GB）上放置 4 个 CentOS 虚拟机，每个虚拟机均是双核配置，8 GB 内存，吉比特网卡，且支持 GraphLab Powergraph 和 MPI。本文采取的告警数据为校园网中抓取的告警数据，总共 50 480 条。实验所用的攻击图使用 MulVAL[20]生成，包含 5 013 个节点和 12 036 条边。

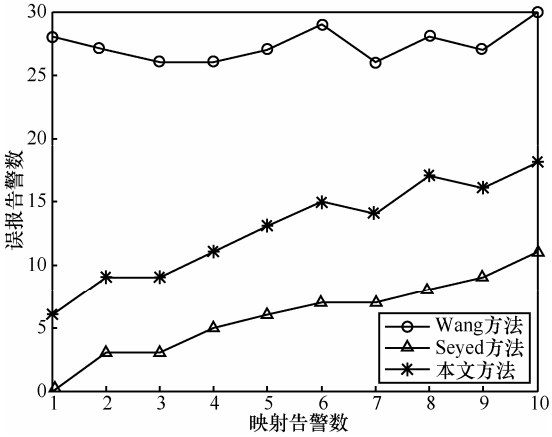

图 6　误报趋势

在分布式性能分析上，本文从针对以下几个方面进行测试。

1) 告警处理时间横向对比。

2) 不同的并发处理告警数量对于告警处理时间的影响。

3) 不同的图规模对于告警处理时间的影响。

表 2 给出本文提出的 AG-PAP 和 Sebastian[11]提出的方法在运行时间上的比较（本文环境和其实验环境参数基本一致），映射时间上降低了 80%，关联分析时间降低了 98%。可以看出，本文引擎在映射时间和关联分析时间上均优于后者，这是由于后者映射需要经过告警到节点、节点到路径和路径到处理单元的 3 层映射和交换而本文仅需要经过告警到节点和节点到处理单元的两层映射；同时关联分析需要分析告警所在所有路径上的其他告警，路径会有重叠，其余告警也会有重复处理，会带来大量的重复分析。

表 2　　实验比较

方法	告警数	总时间/ms	映射时间/ms	关联分析时间/ms
本文	50 480	1 187	21	1 166
Sebastian[11]	43 485	52 617	108	52 509.5

图 7 给出不同规模的并发处理告警数量对于告警处理时间的影响。测试告警来自于采集告警的复制。可以看到在并发告警数量从 5×10^3 到 5×10^4 时，所用时间反而降低了 201 ms；当并发告警从 5×10^4 到 5×10^6，时间只有小幅的增加；当并发告警从 5×10^6 到 5×10^7，时间仅增加了 2.7 倍。这是因为当告警数量少的时候，网络交换通信的开销占了主导，当告警数量增大的时候，单点负载的运算量加

大，因而单点计算开销占了主导，使用时间的增长趋势与告警数量增长趋势一致。同时说明本文的并行告警处理系统更适用于大量告警的并发处理，并发数量增大，性能反而有所提升。

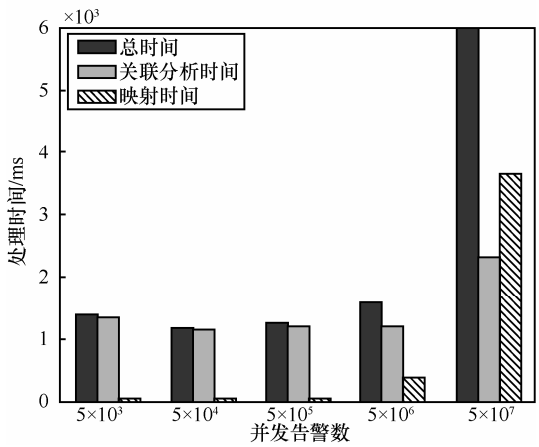

图 7　告警处理速度

大规模图的构建是采用实验网络中的攻击图作为基础，随机插入动作节点和相关的状态节点构成。根据攻击图中的因果关系和经验知识，限制动作节点只能存在到状态节点的出边，状态节点只能存在到动作节点的出边。从图 8 可以看出，图规模也会对告警处理速度产生影响，当图规模每增加一个数量级，处理时间增长为 42%、117%、375%，处理时间涨幅基本来自于关联分析，映射时间基本保持不变。可以看出，处理时间的增幅远小于图规模的增幅，AG-PAP 告警处理引擎对图规模有着较好的适应性。本文提出的基于 GraphLab[21]的分布式并行告警处理方法不仅速度上有大幅提升，同时能够处理更大规模的攻击图，而且能够应对图的实时更新。

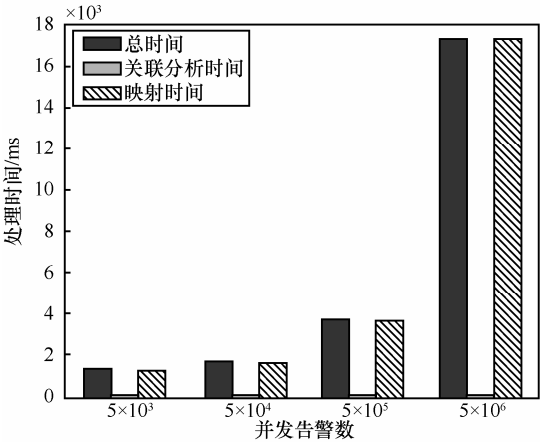

图 8　图规模对告警分析速度的影响

表 3 对比了现有基于攻击图的告警处理方法在各方面的优劣性，可以看出本文所提出的并行处理方法在并行性和关联预测效果上均优于其余 3 种方法。

表 3　基于攻击图的告警处理方法横向对比

方法	大规模图规模处理能力	实时图更新能力	分布式性能	关联预测
Wang[9]	弱	无	无	中
Seyed[10]	无	无	无	中
Roschke[11]	中	弱	中	弱
本文	强	强	强	强

6　结束语

针对现有基于攻击图的关联分析方法在图关联完整性和并行性上的缺陷，本文首先提出了基于攻击图的综合告警关联分析方法，能够解决级联前后件以及间接前后件的关联，增强了告警分析的关联预测能力和减少关联预测带来的误报告警的数量。其次，提出了基于攻击图的并行告警处理框架 AG-PAP，将映射过程和告警分析并行化，形成完整的基于攻击图的告警处理流程，解决了以往攻击图数据难以有效拆分和实时更新，以及基于攻击图数据的关联分析难以有效并行的问题，并通过本地网络实验和模拟数据验证了关联分析的有效性和并行性能方面的提升。下一步的研究工作包括攻击图中的不确定性研究和并行效率的进一步优化。

参考文献：

[1] VALEUR F, VIGNA G, KRUEGEL C, *et al*. A comprehensive approach to intrusion detection alert correlation[J]. IEEE Transactions on Dependable and Secure Computing, 2004, 1(3): 146-169.

[2] ROSCHKE S, CHENG F, MEINEL C. An alert correlation platform for memory-supported techniques[J]. Concurrency and Computation-practice & Experience, 2012, 24(10): 1123-1136.

[3] 梅海彬，龚俭，张明华. 基于警报序列聚类的多步攻击模式发现研究[J]. 通信学报, 2011, 32(5): 63-69.
MEI HB, GONG J, ZHANG MH. Research on discovering multi-step attack patterns based on clustering IDS alert sequences[J]. Journal on Communications, 2011, 32(5): 63-69.

[4] ROSCHKE S, CHENG F, MEINEL C. Using vulnerability information and attack graphs for intrusion detection[A]. Information Assurance and Security (IAS), 2010 Sixth International Conference on IEEE[C]. 2010.68-73.

[5] NOEL S, JAJODIA S. Advanced vulnerability analysis and intrusion detection through predictive attack graphs[A]. Critical Issues in C4I, Armed Forces Communications and Electronics Association (AFCEA) Solutions Series International Journal of Command and Control[C].

2009.

[6] JAJODIA S, NOEL S. Topological Vulnerability Analysis[M]. Springer, 2010.139-154.

[7] OU X, BOYER W, MCQUEEN M. A scalable approach to attack graph generation[A]. ACM[C]. 2006. 336-345.

[8] GONZALEZ J E, LOW Y, GU H, *et al*. PowerGraph: distributed graph-parallel computation on natural graphs[A].OSDI[C]. 2012, 12(1): 2.

[9] WANG L, LIU A, JAJODIA S. Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts[J]. Computer Communications, 2006, 29(15): 2917-2933.

[10] AHMADINEJAD S H, JALILI S, ABADI M. A hybrid model for correlating alerts of known and unknown attack scenarios and updating attack graphs[J]. Computer Networks, 2011, 55(9): 2221-2240.

[11] ROSCHKE S, CHENG F, MEINEL C. High-quality attack graph-based IDS correlation[J]. Logic Journal of the Igpl, 2013, 21(4I): 571-591.

[12] ROSCHKE S, CHENG F, MEINEL C. A new alert correlation algorithm based on attack graph[J]. Computational Intelligence in Security for Information Systems, 2011, 6694: 58-67.

[13] DEAN J, GHEMAWAT S. MapReduce[J]. Communications of the ACM, 2008, 51(1): 107.

[14] LOW Y, BICKSON D, GONZALEZ J, *et al*. Distributed graphlab: a framework for machine learning and data mining in the cloud[J]. Proceedings of the VLDB Endowment, 2012, 5(8): 716-727.

[15] MALEWICZ G, AUSTERN M H, BIK A J C, *et al*. Pregel: a system for large-scale graph processing[A].Proceedings of the 2010 ACM SIGMOD International Conference on Management of data[C]. ACM, 2010.

[16] LI K, GIBSON C, HO D, *et al.* Assessment of machine learning algorithms in cloud computing frameworks[Z]. IEEE, 2013.98-103.

[17] GUO Y, BICZAK M, VARBANESCU A L, *et al*. Towards benchmarking graph-processing platforms[A].The International Conference for High Performance Computing, Networking, Storage and Analysis[C]. 2013.

[18] CHING A, KUNZ C. Giraph: large-scale graph processing infrastructure on Hadoop[J]. Hadoop Summit, 2011, 29(6).

[19] LOW Y, GONZALEZ J, KYROLA A, *et al*. Graphlab: a new parallel framework for machine learning[A]. UAI[C]. 2010.340-349.

[20] OU X, GOVINDAVAJHALA S, APPEL A W. MulVAL: a logic- based network security analyzer[A]. 14th USENIX Security[C]. 2005. 1-16.

[21] LOW Y. GraphLab: A Distributed Abstraction for Large Scale Machine Learning[D]. University of California, 2013.

作者简介：

刘威歆（1987-），男，广东深圳人，北京邮电大学博士生，主要研究方向为网络攻防、入侵检测和日志分析。

郑康锋（1975-），男，山东烟台人，北京邮电大学副教授，主要研究方向为网络与信息安全。

武斌（1981-），男，山东泰安人，北京邮电大学讲师，主要研究方向为网络安全。

杨义先（1961-），男，四川盐亭人，北京邮电大学教授、博士生导师，主要研究方向为信息安全与密码学。