基于语义的自动推理

使用SMT的AWS访问策略

John Backes，Pauline Bolignano，Byron Cook，Catherine Dodge，Andrew Gacek，
Kasper Luckow，Neha Rungta，Oksana Tkachuk，Carsten Varming
亚马逊网络服务

抽象云计算提供对IT的按需访问
通过互联网的资源。 这些资源的权限是
由富有表现力的访问控制策略定义。 本文介绍
Amazon Web Services（AWS）策略的正式化
语言和相应的分析工具，称为Z ELKOVA，
用于验证策略属性。 Z ELKOVA编码语义
策略进入SMT，比较行为，并验证属性。
它为用户提供了一种检测错误配置的可靠机制
他们的政策。 Z ELKOVA解决了PSPACE完整的问题
并且每天被调用数百万次。

I.我 引言

云计算提供对IT的按需访问
通过互联网来源。 访问的便利性
通过用户指定的访问权限使云中的源安全
控制政策。 访问控制策略是一种表达方式
可以访问哪些资源的规范，由谁，
在什么条件下。 正确配置的策略是
组织安全态势的重要组成部分。 该
基于云的服务的规模和多样性不断增长 -
（例如，无服务器计算，流分析，边缘 -
计算设备），以及组织使用的每个新产品
nization可能需要不同的访问策略配置。
而且，客户正在将这些服务结合起来
意味着复杂性越来越多地转移到政策中。
因此，许多客户面临着安全挑战
关于动态系统静态策略的推理之一
电信设备制造商。 云客户需要一种允许他们检查的工具
策略配置基于其安全要求。

Amazon Web Services（AWS）定义了一种策略语言
允许用户管理对AWS资源的访问。 权限
由政策授予的依赖于不同国家的相互作用 -
条件和条件。 政策语言支持
播放授予访问权限的语句（允许语句）
或撤销访问（拒绝声明）。 此外，条件
内部声明可以基于访问详细信息，例如
源地址，加密和其他配置选项。

用户希望保证他们的政策授予权利
权限。 要验证策略是否表达了预期的内容，
一些AWS用户已经实现了基于启发式的语法
检查检测策略中的某些模式，例如，使用
一个使资源可公开访问的通配符。 虽然
有用的，基于启发式的语法检查是不健全的，因为
他们没有充分考虑政策的语义
语言。 其他人试图明确列举所有可能的
请求政策，但很快发现这个难以处理。

在本文中，我们介绍了开发和应用
Z ELKOVA，一种旨在推理的政策分析工具
AWS访问控制策略的语义。 Z ELKOVA翻译
政策和财产成为可满足性模数理论
（SMT）公式并使用SMT求解器检查有效性
属性。 我们使用现成的解决方案和内部解决方案
Z3的扩展称为Z3A UTOMATA 。

Z ELKOVA关于允许的所有可能权限的原因
策略以验证属性。 例如，Z ELKOVA
可以回答这个问题“这个资源是否可以访问？
特殊用户？ “和”任意用户都可以写这个
资源？“ 。 要验证的属性在策略中指定
语言本身，消除了对不同规范的需求
财产的形式或形式。 此外，Z ELKOVA
为常见属性提供了许多内置检查。

SMT编码使用常规字符串理论
表达式，位向量和整数比较。 指某东西的用途
通配符*（任意数量的字符）和？ （恰好一个
字符串约束中的字符）决定了问题
PSPACE完成。 但是，我们与现实世界的经验
政策是99％的政策问题可以回答
小于160毫秒。

Z ELKOVA是基础政策分析引擎
越来越多的AWS服务。 使用了数百万
Z ELKOVA 每天 都会分析相关的政策
计算，存储，消息传递，搜索，分析的资源 -
ics和其他功能。 AWS服务的一个示例
集成Z ELKOVA包括Amazon S3（对象存储），
Amazon Confie（基于变更的资源审计员），Amazon Macie
（安全服务），AWS Trusted Advisor（符合AWS
最佳实践）和亚马逊GuardDuty（智能威胁
检测）。 此外，Z ELKOVA由内部AWS安全性使用
审计工具，以实施政策协议的安全最佳实践
形象，例如，禁止公共访问资源。

A.相关工作

政策语言已被用于各种领域，
例如，信任管理，分布式授权，基于
访问，资源访问控制[1] - [6]。 几项
语言被定义为Datalog程序，因为它
有效的财产核查 [2]，[6] - [10]。 A
策略语言是根据JSON序列化定义的，
并且旨在用于各种云服务
和访问控制的场景。 Z ELKOVA结合了所有
单个分析工具中的策略语言的组件。

菲斯勒等人。 定义一个包含的政策形式主义
在不同的环境状态之间转换
确定策略中的访问控制[2]。 访问控制
AWS中的模型还使用策略和动态环境
请求上下文以确定权限，但环境
在单个访问请求期间不会发展。 其他政策

政策→声明 *
声明→（效果，主体，行动，资源，条件？）
效果→ 允许 | 拒绝
校长→ 校长： 字符串*
动作→ 动作： 字符串*
资源→ 资源： 字符串*

框架合例如XACML[12]。允许在 员的工作的策略
Hughes和Bultan将XACML 转换为布尔值
可满足性问题并使用SAT求解器检查部分或 -
使用有界分析的政策之间的差异。 束缚了
然而，分析使它变得不健全。 相比之下，编码
Z ELKOVA的 SMT是合理的。 TRBAC政策模型使用
具体的时间单位授予或撤销访问权[13] 。 这个
在AWS策略语言中使用条件完成
在日期和时间。 最后，SecGuru工具 [14]进行了比较
使用SMT理论的网络连接策略
向量。

我们目前的工作在三个方面最突出。
首先，我们使用现有的工业政策语言
发展以满足数百万用户和用例的需求。 该
语言健壮且灵活，具有出现的功能
从实际需要。 其次，我们与服务密切合作
团队整合我们的工具并开发定制的预制
与每个服务的用户相关的属性。 最后，我们
用我们的工具吸引了数百万观众。

II。 一个 PPROACH

当向AWS服务发出访问请求时，a
生成请求上下文，其中包括主体
发出请求，请求的资源和
要求的具体行动。 政策评估引擎
将此请求上下文与用户的策略进行比较
以及确定是授予还是拒绝访问的资源。

Z ELKOVA通过推理全部来验证AWS策略
可能的请求上下文。 的基本机制
Z ELKOVA能够说出一项政策是否不那么平等 -
宽容比另一个。 可以将属性指定为绑定 -
表示上限或下限的ary策略
期望的行为。 Z ELKOVA的支持不那么均等
然后建立这些界限的正确性或找到一个
反例。

A.政策语言概述

AWS策略语言定义为序列化JSON 1 ，
然而，在本文中，我们描述了核心结构
简化抽象语法中的策略语言。 例子
在本文中也使用这种抽象语法。

图 1显示了策略的抽象语法
语言。 在这个语法中，？ 表示可选元素
和 * 表示列表值元素。 政策是一个
陈述清单。 每个Statement都包含一个元组
（校长，效果，行动，资源，条件？ ）。 该
条件是策略中的可选元素

```
条件→ 条件 ： 操作员*
运算符→ （OpName，KeyName，Value * ）
OpName→ StringEquals | StringEqualsIfExists | StringLike |
        StringNotEquals | IpAddress | ...
KeyName→ aws：sourceVpc | aws：sourceIp | s3：前缀 | ...
值→字符串| num | 布尔
```

图1. AWS策略语言的简化抽象语法

其他是必需的。 效果构造指出是否
该声明允许或拒绝访问。 默认情况下，访问
资源被拒绝。 允许语句覆盖默认值
权限和拒绝语句会覆盖权限
允许声明授予。 换句话说，要获得访问权限
一个资源，必须有一些允许声明授予
access和no deny语句撤销该访问权限。 那里
对策略中的语句没有排序约束。

Principal构造在策略中用于指定哪个
授予或拒绝用户，帐户，服务或实体
获取资源。 主体是唯一标识的
按字符串值。 Action构造指定了列表
对应的允许或拒绝的操作
资源。 各种AWS服务发布一组操作
可以由用户执行特定于那些资源的资源
服务。 Resource构造指定服务列表
授予或拒绝访问的特定资源。
每个AWS服务都有自己的一组资源和每个AWS
资源由字符串值唯一标识。 字符串值
for Action和Resource可以包含通配符*
匹配任意数量的字符和通配符？ 哪一个
恰好匹配一个字符。

Condition构造指定条件
访问被授予或拒绝。 在条件构造中
使用条件键上的运算符构造压力
价值对。 条件运算符按其分组
类型：字符串，数字，日期和时间，布尔值，二进制，IP
地址等。 运算符名称（OpName）表示
类型和比较器。 字符串条件运算符
字符串条件的比较，例如， StringEquals检查
字符串相等， StringLike根据模式检查字符串。
完整的运营商清单在IAM文件中定义 -
塔季翁 2 ，我们的实施支持。 运营商
应用于条件键（ConditionKey）。 每个条件
key被映射到相应的值。 某些条件键
在所有服务中全局定义，例如， aws：sourceIp ，
而其他条件键是特定于服务的， s3：前缀 。

1 https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements.html

2 https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_condition_operators.html

第3页

```
（（允许，
主要    ：  学生们，
行动    ：  的getObject，
资源    ：  cs240 / Exam.pdf），
（允许，
主要    ：  塔斯，
行动    ：  的getObject，
资源    ：  （cs240 / Exam.pdf，
             cs240 / Answer.pdf））
```

（a）政策X.

```
（（允许，
主要    ：  *
行动    ：  的getObject，
资源    ：  cs240 / *），
（否认，
主要    ：  学生们，
行动    ：  的getObject，
资源    ：  cs240 / Answer.pdf））
```

（b）政策Y.

图2.学生和助教获得考试和答案的示例政策。

X_0 ： a = “getObject” ∧p= “students” ∧r= “cs240 / Exam.pdf”
X_1 ： a = “getObject” ∧p= “tas” ∧
(r = “cs240 / Exam.pdf” ∨r= “cs240 / Answer.pdf”)
X：$X_0 \vee X_1$

Y_0 ： a = “getObject” ∧r= “cs240 / *”
Y_1 ： a = “getObject” ∧p= “students” ∧r= “cs240 / Answer.pdf”
Y：$Y_0 \wedge \neg Y_1$

图3.来自图 2 的策略X和Y的SMT编码 。

确定政策X的允许程度是否小于或等于
策略Y，Z ELKOVA使用SMT求解器来检查是否

$$(X_0 \vee X_1) \Longrightarrow (Y_0 \wedge \neg Y_1)$$

是有效的，这是真的。 此检查的结果表明所有
策略Y允许策略X允许的请求。

但是，策略Y允许其他权限。 那里-
策略Y允许的allow语句中的源“cs240 / *”允许
“学生”和“tas”校长访问对象（文件）
除“Exam.pdf”和“Answer.pdf”之外，例如“Class-
Roster.pdf”。 政策Y另外授予校长以外的校长
“学生”和“塔斯”访问桶中的资源
“cs240”，因为deny声明只否认“学生”
访问“Answer.pdf”。 这导致公众可读
因为任何其他主体都可以执行getObject
对存储桶内容的操作。 因此这个政策确实如此
不代表用户的意图，并且它最好地违反安全性
实践。 这表明需要对政策进行合理分析。
Z ELKOVA通过减少数学政策来实现这一目标
公式并使用SMT求解器验证其属性。

III。 SMT E NCODING

在本节中，我们描述了Z ELKOVA的SMT编码。
2和使用字符串理论，正则表达式，
位向量和整数比较。 政策语言是
声明性的，没有诸如循环之类的编程结构
或动态分配的数组。 政策的语义
语言编码为SMT公式。 权限

B.例子

亚马逊简化抽象语法中的策略
简单存储服务（S3）如图1所示。 2.亚马逊
S3是存储逻辑单元的对象存储
叫一桶。 S3将数据存储为这些存储桶中的对象。
每个资源，例如桶和桶中的对象，
通过亚马逊资源名称唯一标识
（ARN）。 附加到存储桶的策略控制对访问
桶和桶中的对象。 图 2 （a）中的政策
说学生可以阅读考试和助教
可以阅读考试及其答案。 另一项政策，
如图 2（b）所示，说每个人都可以访问所有的
cs240的内容/除了学生不能访问
答案。

图 3显示了图2中策略的编码
每个策略的编码是三个变量p的公式，
a和r对应于主体，动作和资源
在资源请求上下文中。 公式评估为true
只要政策授予访问权限。 由于政策X有两个
允许可以授予访问权限的语句，它由表示
他们分开了。 另一方面，政策Y有一个允许
语句Y 0和一个拒绝语句Y 1 。 因此只有政策Y.
如果Y 0允许访问且Y 1不拒绝它，则授予访问权限：
ÿ0∧¬Y1。 请注意，我们在Y 0中滥用符号来表示
r = “cs240 / *” 因为这实际上将对应于一个表格
字符串匹配而不是相等。 我们讨论细节
第 III-A 节中的字符串匹配

由策略授予的编码为所有权限
由允许声明授予而不是由拒绝状态撤销 -
发言：：

$$\square_{S\in Allow}^{\vee}[[S]] \wedge \neg \square_{S\in Deny}^{\vee}[[S]] \quad (1)$$

这里Allow和Deny是允许和拒绝语句的集合
在政策中。 每个语句[[S]]的语义含义是
allow语句或者由permit语句授予的权限集
由deny语句撤消的权限集。

策略中的每个语句都对该约束进行编码
委托人，行动，资源和条件：

$$[[S]] := \square_{v\in P(S)}^{\vee} p = v \wedge \square_{v\in A(S)}^{\vee} a = v \wedge \square_{v\in R(S)}^{\vee} r = v \wedge \square_{O\in C(S)}^{\wedge}[[O]] \quad (2)$$

函数P（S）返回指定的所有字符串值
为校长。 类似地，A（S）和R（S）返回字符串
语句中的操作和资源的值。 该
函数C（S）返回给定的条件运算符集
声明。 变量p，a和r分别映射到
主体，动作和资源值。 一个中的权限
声明被授予作为对字符串值的析取
主体，动作和资源值以及连接
在方程式中所示的条件下。 （2）。

第4页

```
（允许，
主要    ：*
行动    ：的getObject，
资源    ：cs240，
条件    ：（StringEquals，aws：sourceVpc，vpc-111bbb222），
          （StringLike，s3：prefix，cs240 / Exam *））
```

图4.具有两个条件的示例策略。

策略中的每个条件都编码一个约束
相应的条件键：

$$[[O]] := \wedge_{\langle op,k,V\rangle \in CO(O)} condExists_k \wedge \square_{v\in V}^{\vee} op(k, v) \quad (3)$$

每个条件都映射到运营商名称，密钥名称和
通过函数CO（O）的值列表。 一个意思
条件由所有列出的值的析取编码。
布尔变量condExists k表示条件键k，
必须存在于请求上下文中。 变量k代表
条件密钥存在时的值。 运营商（op）
定义键和值对（k，v）上的操作，例如，
平等或不平等。

接下来，我们介绍几个重要类的编码
条件运算符。

A.字符串约束

Z ELKOVA中 的策略编码主要是通过
使用字符串约束。 这包括字符串相等
和不等式约束，以及模式匹配
常用表达。 主体，行动和资源
策略中的结构编码为字符串约束。 串
运算符及其相应的条件键也是
编码为字符串约束。 con的一个示例策略
ditions如图 4 所示。运算符StringEquals是
应用于条件键 aws：sourceVpc的值为
“vpc-111bbb222”，它限制对特定虚拟的访问
AWS云中的专用网络（VPC）3 。 字符串运算符
StringLike 应用于条件键 s3： 带有a的前缀
“grade / *”的值，它限制访问，以便只有对象
在“等级/”目录下可能会列出。

图 5显示了该示例的SMT编码。 该
布尔变量vpcExists和s3PrefixExists编码
是否条件 aws：sourceVpc和s3：prefix
存在于请求上下文中。 约束
“grade /” prefixOf s3Prefix对“grade /”为a进行编码
变量s3Prefix的前缀。 以下请求上下文
对应于对该组约束的令人满意的分配
在图 5中：

{校长：鲍勃，

```
a = “listBucket” ∧r= “cs240” ∧
vpcExists∧vpc= “vpc-111bbb222” ∧
s3PrefixExists∧ “grade /” prefixOf s3Prefix
```

图5.图 4中 的策略的SMT编码

```
（允许，
主要    ：*
行动    ：listBucket，
资源    ：*
条件    ：（StringEquals，s3：前缀，UpLoads），
          （StringEqualsIgnoreCase，s3：prefix，Uploads））
```

图6.具有混合情况的示例策略。

最多可以支持两个*通配符。 稍后我们会看到一个不同的
编码其他通配符。 当前的例子
编码在（4）中给出 。

$$\begin{aligned}&\text{“cs2 * / Exam *”} \mapsto \text{“cs2”}\ \text{prefixOfVar}\wedge\text{Var 包含 “/ Exam”}\\ &\text{“cs2 * / *考试”} \mapsto \\ &\quad\text{“cs2”}\ \text{prefixOfVir}\wedge\text{Var 包含 “/”}\ \wedge\ \text{“Exam”}\ \text{suffixOf Var}\\ &\text{“* 240 / *考试”} \mapsto \text{Var 包含 “240 /”}\ \wedge\ \text{“考试”}\ \text{后缀为 Var}\end{aligned} \quad (4)$$

当图案的不同部分可以重叠时，我们不允许
可能的重叠。 例如，“ab * bc”转换为
“ab” prefixOfVar∧ “bc” suffixOfVar∧Var = “abc”。
请注意，“abc”否则将满足前缀和后缀
约束，但它与模式“ab * bc”不匹配。

B.正则表达式约束

更复杂的字符串约束需要更强大
编码。 特别是，上述编码不能
用？来表示约束？ 通配符或两个以上*
通配符。 例如，以下编码失败，因为
它不会限制“b”出现在“c”之前。

$$\begin{aligned}&\text{“a * b * c * d”} \mapsto \text{“a”}\ \text{前缀Var}\wedge\wedge\text{Vir 包含 “b”}\\ &\quad\text{Var 包含 “c”}\ \wedge\ \text{“d”}\ \text{后缀为 Var}\end{aligned} \quad (5)^{\wedge}$$

在这种情况下，我们使用正则表达式对这些进行编码
限制。 例如，（6）显示了两个基于的编码
传统的正则表达式模式格式 “。”
代表任何单个字符，“*”代表Kleene星
运算符表示前一个或多个出现的运算符
字符。

$$\begin{aligned}&\text{“cs ??? / Exam *”} \mapsto \text{Var 匹配 “cs ... / Exam。*”}\\ &\text{“cs2 * / Exam / * / Results / *”} \mapsto \\ &\quad\text{Var 匹配 “cs2。* / Exam /.*/ Results /.*”}\end{aligned} \quad (6)$$

一些条件运算符区分大小写（StringEquals，
StringLike）而其他人不区分大小写（StringEqualsIg-

资源：listBucket，
条件：{aws：sourceVpc：vpc-111bbb222，
s3：前缀：grade / 2018 / final /}}

为了在字符串中编码*通配符，我们使用prefixOf，suffixOf，并包含字符串运算符。 有了这个编码我们

3 https://aws.amazon.com/vpc/

noreCase，Bool），使用哪种类型的运营商
相同的条件键确定大小写的确切编码
灵敏度。 当条件键仅受约束时
区分大小写的操作符，没有什么特别需要做的。
当条件键仅受大小写不敏感约束时
运营商，所有这些比较的目标都被转换
以小写字母解决问题。 困难的情况是

第5页

```
a = "listBucket" ∧s3PrefixExists∧
s3Prefix 匹配 "UpLoads" ∧
s3Prefix 匹配 "[uU] [pP] [lL] [oO] [aA] [dD] [sS]"
```

图7.图 6中 的策略的SMT编码

	Z3	CVC4	Z3A UTOMATA
UNSAT	965,092	34,908	0
SAT	959,543	39,932	525

图8.每个解算器在100万UNSAT中最快的次数
和一百万个SAT财产检查。

当条件键受约束区分大小写时
和不区分大小写的运算符。 以前的con方法
将不区分大小写的运算符的所有目标都转换为小写
会干扰区分大小写的运营商。 相反，案例
敏感的比较在目标时正常处理
不区分大小写的比较被编码为常规
表示所有可能的案例组合的表达式。 对于
例如，考虑人为的条件组合
如图 6 所示。这里有一个区分大小写和
对 s3：前缀条件键不区分大小写的约束 。
这些约束的Z ELKOVA编码如图7所示
我们使用[xX]形式的字符类来表示
一个匹配单个字符的正则表达式
“x”或“X”。

C.位向量约束

该 Ip地址条件运算符允许用户限制
基于IP地址的访问。 该 Ip地址运算符
与 aws 结合使用：SourceIp条件。 该
aws的 值：SourceIp必须属于Classless Inter-
域路由（CIDR）格式。 CIDR格式关联
网络掩码作为IP地址规范的一部分。 例如，
CIDR表示法中的IPv4 11.22.33.0/24表示第一个
24位IP地址被认为是重要的。 考虑
两个条件的翻译，其中一个是 aws：SourceIp
设置为11.22.33.0/24，另一个设置为11.22.0.0/16：

```
C 0 ： （IpAddress， aws：SourceIp，11.22.33.0 / 24）↦→
    ipV4Exists∧（0x0B162100 =（ipV4＆0xFFFFFF00））
C 1 ： （IpAddress， aws：SourceIp，11.22.0.0 / 16）↦→
    ipV4Exists∧（0x0B160000 =（ipV4＆0xFFFF0000））
```

布尔变量ipV4Exists编码存在
条件 aws：SourceIp和位向量变量ipV4
编码实际值。 使用按位AND运算
屏蔽约束中IP地址的无关紧要的位。
通过这种编码，我们[[C 0]] =⇒[[C 1]]有效。 那里
在约束C 0和中的IP地址中是24个有效位
约束C 1 中的IP地址中只有16个有效位 。
公共路由前缀是相同的。 因此，请求上下文
C 1也允许C 0允许的。

D.其他运营商

数值运算符的条件只执行整数
比较。 策略中没有算术运算
语言和数值之间没有相互作用
字符串值，例如，您不能取字符串的长度。
适用于布尔运算符的条件很简单
编码为布尔约束。 IfExists的 条件
后缀检查请求中是否存在条件键

上下文。 这个后缀可以添加到其他条件操作中 -
诸如 StringEquals之 类的 转子会产生一个新的运算符
StringEqualsIfExists 。 可以应用生成的运算符
到 aws：sourceVpc条件键。 例如：

```
（StringEqualsIfExists， aws：sourceVpc，“vpc-111bbb222”）↦→
awsSourceVpcExists =⇒awsSourceVpc= “vpc-111bbb222”
```

(8)

IV。 Z3A UTOMATA

Z3A UTOMATA是Z3的内部扩展设计
为常规理论提供完整的决策程序
表达式。 如第 III 部分所述，Z ELKOVA使用了
涉及两个以上问题的正则表达式
*通配符，任何？ 通配符，或者诸如的棘手组合
混合区分大小写和不区分大小写的字符串比较。
这种情况一般很少见，但在我们的规模上很常见
我们每天都会收到数百万条查询。
Z3和CVC4旨在有效地解决问题
单词方程，一个严格比一般问题更普遍的问题
表达匹配。 这有时会导致降级
纯正则表达式问题的表现。 对于前
很多，都无法回答“是否存在问题”
字符串匹配'ab * b * b * b'而不是'a * b * b * b'？ “。 更多
一般来说，两个解算器似乎都对小变化非常敏感
在输入编码中，我们的一个快速解决的问题
域名变为非终止。 然而，常规理论
表达式是可判定的，我们的问题仍然存在
(7)理论。 因此，Z3A UTOMATA填补了我们的重要利基
域。

图 8显示了哪个求解器对于一毫米来说最快 -
狮子会UNSAT和100万SAT Zelkova财产检查，
都随机选择。 请注意，对于UNSAT问题，
Z3A UTOMATA永远不是最快的解算器。 SMT问题
Z ELKOVA生成的包含简单和简单的混合
复杂字符串约束。 对于Z ELKOVA 的属性
根据我们的经验，UNSAT的结果总是到期
简单的字符串约束是不可满足的。 Z3和
因此，CVC4可以轻松有效地处理这种情况
Z3A UTOMATA从未获胜。 在约束的情况下
是可以满足的，必须考虑所有的约束，包括
复杂的。 在这里，Z3A UTOMATA经常能够获胜
在Z3和CVC4不终止的情况下。
Z3A UTOMATA使用。来解决正则表达式问题
通过确定有限自动机（DFA）的标准转换
非确定性有限自动机（NFA）。 它使用Hopcroft的
DFA最小化算法 [15]。 Z3A UTOMATA是
关于字符集的度量并努力生成
字符串仅使用字符集的可打印子集。 该

第6页

```
（允许，
校长：*，
行动：getObject，
资源：*，
condition：（StringEquals，aws：sourceVpc，vpc-111bbb222））
```

```
（（允许，
校长：*，
行动：发信息，
资源：*，
条件：（ArnEquals，aws：sourceArn，mytopic））
```

校长：*，
行动：putObject，listBucket，...，
资源：*）

图9。 仅允许来自getObject请求的策略检查VPC-111bbb222。

全方位的正则表达式（和自动机）功能
支持包括交集，联合和补充。

Z3A UTOMATA目前仅在SAT上与Z3集成level并将每个正则表达式匹配视为原子。 一个SMT社区要解决的未来挑战是如何将其整合到传统的Nelson-Oppen框架中。

V. Z ELKOVA P ROPERTIES

使用云服务的组织需要保证用户创作或修改的策略不会违反一般安全最佳实践，遵守安全指南 - 由组织定义的行，不拒绝访问给目标用户。 这些属性的例子如下如下：“确保不允许不受限制的公开写入特定资源。 ”（安全最佳实践），”确保只允许从特定范围访问资源IP地址。 ”（组织安全检查）和”确保允许特定用户执行特定操作资源“（可用性属性）。 这些属性可以在策略语言中指定并由Z ELKOVA 检查 。 Z ELKOVA 对物业的验证确保了这一点一个中没有不恰当配置的资源组织。

A.组织安全检查

我们使用部分的示例 II来描述如何在organization可以使用策略语言指定属性可由Z ELKOVA 检查 。 图2 （b）中的例子允许校长“*”访问cs240资源并拒绝学生访问Answer.pdf。 主体设置为通配符可能导致用户未经授权访问对象没有大学的成员方法见 II 。 如保护措施，假设，大学管理员希望确保没有未经授权的数据访问在水桶里。 管理员和安全负责人大学决定要检查适当的财产是“CS部门S3存储桶上的getObject操作只允许来自vpc-111bbb222的请求。 “VPC是由大学拥有，因此从内部访问请求VPC是值得信赖的。

一个策略，指定属性“getObject actions只允许来自vpc-111bbb222“如图 9 所示 。 图 9中 的第一个allow语句仅允许getObject当请求来自vpc-111bbb222时。 第二allow语句允许所有其他不相关的操作与比较有关。 图 9中 的策略表示

（（允许，（一个）
校长：*，
行动：发信息，
资源：*
条件：（ForAllValues：ArnEquals，aws：sourceArn，mytopic））
（b）中

图10.受 aws 约束的策略：sourceArn。 （a）政策不允许世界可写性。 （b） ForAllValues语义允许世界可写性。

在该组请求上下文中的期望上限应该被允许。 只有在这种情况下才会违反此约束输入策略允许图 9不允许的请求 。 在在这种情况下，请求必须是getObject请求（因为第二个allow语句允许所有其他请求在图。 9）它必须来自vpc-111bbb222之外（因为允许VPC内的所有putObject请求第一个允许声明）。 这样的要求确实会违反拟议的财产。 另一方面，如果Z ELKOVA显示输入策略意味着图 9中 的策略然后是上层已建立约束，拟议的财产成立。

B.安全最佳实践

Z ELKOVA支持几种内置检查利用来检查各种安全最佳实践。 EX-其中的大量内容包括检查政策是否允许Amazon S3，Amazon等服务的全球可访问性SQS，Amazon SNS，Amazon Glacier，Amazon Elasticsearch，和AWS Lambda。 这些AWS服务提供计算，存储，消息传递和搜索功能。 这些检查是AWS内部使用以最佳地检查对安全性的遵守情况实践，也可通过外部客户获得Amazon Macie，AWS Config，AWS Trusted等服务顾问和Amazon S3控制台。 内置支票无需用户即可提供更高的安全保障定义属性。

考虑Amazon SQS的情况，这是一个完全托管的消息圣人排队服务。 Z ELKOVA提供内置支票了解Amazon SQS政策是否适合全球访问。 图 10 （a）显示了一个示例SQS策略，其中al-任何委托人都将sendMessage降低到任何资源，有条件的。 条件限制了来源（aws：sourceArn）消息是特定的源（mytopic）。 类似的政策如图 10（b）所示。 在这里，operator ForAllValues：ArnEquals应用于条件aws：sourceArn， 其值仅限于mytopic。 该运算符前缀 ForAllValues的语义表明if条件aws：sourceArn存在，那么它的值是mytopic。 SMT的公式如下：

$$\text{awsSourceArnExists} \Rightarrow (\text{awsSourceArn} = \text{mytopic})$$

当请求上下文没有条件键时aws：sourceArn 设置，上面的公式为true。因此任何

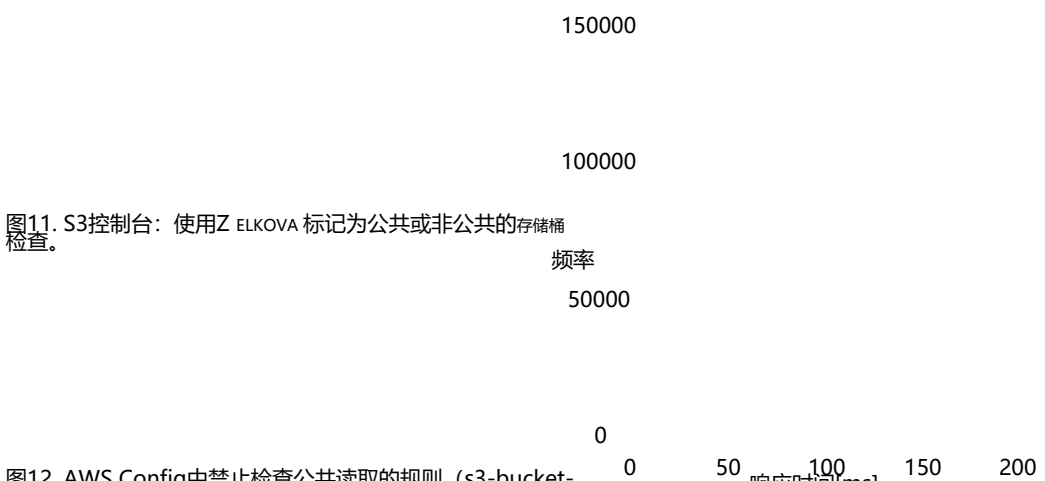

图11. S3控制台：使用Z ELKOVA 标记为公共或非公共的存储桶检查。

图12. AWS Config中禁止检查公共读取的规则（s3-bucket-禁止公开阅读和公开写作（s3-bucket-public-write-禁止使用Z ELKOVA 的S3铲斗。

图13. Z ELKOVA对100万随机政策问题的表现

principal可以向SQS队列发送消息。Z ELKOVA

内置检查SQS世界可访问性正确标记
图10（a）不是世界可访问的，图10（b 世界的
无障碍。

VI。 我 NDUSTRIAL Ë XPERIENCE

Z ELKOVA集成在许多AWS服务中，包括
Amazon S3，AWS Config，Amazon Macie，AWS Trusted
顾问和亚马逊GuardDuty。另外，Z ELKOVA是
AWS安全团队由内部安全审核员使用。

Amazon S3控制台是基于Web的界面
用户可以配置桶; 管理存储桶，对象和
文件夹; 并设置对存储桶和对象的权限。最近
释放控制台添加了一个显示是否存储桶的视图
可公开访问（公共）或不公开（非公开）。该
基础检查由Z ELKOVA 执行。图11显示
这个观点的一个例子。

AWS Config目前支持多种托管规则
在Z ELKOVA 4，例如检查AWS Lambda函数
授予不受限制的访问权限，检查S3存储桶授权
不受限制的读访问权限，检查S3存储桶授权
不受限制的写访问权限，拒绝不执行的putObject请求
具有服务器端加密，并拒绝不允许的操作
https流量。Config将触发 基于Z ELKOVA的新检查
每当创建新资源或附加策略时
改变了。使用Config控制台，客户可以确定
如图所示，他们的S3存储桶符合这些规则
在图 12中，当权限改变时接收通知
或在控制台中查看权限历史记录。检查

4 https://docs.aws.amazon.com/config/latest/developerguide/管理规则按AWS-config.html

服务类似于AWS Config中的服务。

Z ELKOVA由内部安全审计工具使用，拥有
由AWS安全团队执行，扫描所有内部AWS
用于检查资源的意外配置的帐户。
内部帐户是AWS拥有的所有AWS账户
开发团队和人员。这些包括政策
附加到各种资源，如S3存储桶，SQS队列，
SNS主题，Glacier Vaults，KMS Keys，ElasticSearch Do-
主电源和AWS Lambda函数。安全审计
工具定期扫描所有资源并检查compli-
根据安全性最好的资源政策
实践。违反支票的行为将自动出票
发现，分配给所有者，并自动解决
当政策得到修复时。审核工具无需手动操作
安全工程团队的干预。

在Amazon Macie，AWS Config中可用的检查时，
Amazon S3控制台和AWS Trusted Advisor检查安全性
属性，Z ELKOVA集成在Amazon GuardDuty中
检查可用性属性。Z ELKOVA确保了
必要的权限在用户的策略中启用
正在加入这项服务。

A.实施

Z ELKOVA运行在AWS Lambda上，这是一种无服务器计算
在没有用户需要的情况下运行应用程序的平台
愿景或管理服务器。Z ELKOVA 的输入是JSON
由被比较的政策组成的结构，
或政策和内置Z ELKOVA支票的名称。该
来自Z ELKOVA的回复也是一个JSON结构
回答查询。为了比较政策，它会返回

有效载荷中的第一个策略是否更不宽容，更多
宽容，等同或无可比拟的
有效载荷中的第二个策略。对于每个内置支票，
Z ELKOVA采取政策并根据其返回真或假
是否满意。如果Z ELKOVA无法处理
策略中的任何构造或解算器超时，它返回
未知。

Z ELKOVA使用求解器Z3，Z3A ATA和CVC4
在后端解决查询[16]，[17]。解[illegible]供
字符串，正则表达式，位向量和[illegible]
整数比较理论。Z ELKOVA调用解算器
并行并在其中一个求解器后立即返回结果
提供了答案。我们使用传统的Z3求解器
序列字符串求解器。与其他求解器的实验
Z3Str3 [18]和其他基于自动机的解算器[1 其中的一部分
我们未来的工作。

B.使用统计

Z ELKOVA 的调用总数范围为a
一天内有几百万到几千万。 的数量
调用因调用Z ELKOVA 的服务而异。
某些服务 以某种常规节奏调用Z ELKOVA，
例如，内部安全审计工具，而其他服务，
例如，AWS Config 在更改时调用Z ELKOVA
在政策中指导。

图 13显示了Z ELKOVA在一百万上的表现
随机选择的政策问题。这些包含两项政策
比较和内置检查。总时间包括时间
解析输入JSON，将策略编码为SMT，
执行检查，并构造生成的JSON
退回。y轴表示计数，即数量
政策在时间内得到解决。图表显示99%的
政策在160毫秒内解决。

七。 C 结论

在本文中，我们提出了一个正式化的
用于控制对资源的访问的AWS策略语言。 这个
是第一个正式化AWS策略语言的实例
作为SMT公式。这种方法的优点是它
允许我们使用现成的SMT求解器来验证安全性
和可用性属性。鉴于该的分布式性质
政策语言，不同的服务建立自己的
条件键列表，这项工作提供了单一的合并
服务以推断适用的政策的语义
跨AWS的不同服务。先前的技术水平
在AWS服务的策略检查中使用语法检查
数百万人使用的复杂工业政策语言
以一天为周期。而且，这项工作是最大的和
工业中正式方法的最广泛使用。

未来工作有两种途径。一条途径是
改进Z ELKOVA中提供的现有功能。这个
包括Z3A UTOMATA的进一步工作，以使其更多
竞争的。第二个途径是增强功能
Z ELKOVA发动机本身。例如，我们想要
在Z ELKOVA中添加支持以向具体用户返回
使用SMT求解器生成的模型请求上下文
在执行检查时。具体的请求上下文将
向用户提供有关特定支票通过的原因的信息
或失败。我们还想添加对推荐的支持
政策未通过某项检查时的政策修复。

R EFERENCES

[1] DJ Dougherty，K。Fisler和S. Krishnamurthi， “指定和
关于动态访问控制策略的推理， “在IJCAR，第一卷。4130。
Springer，2006，pp.632-646。
[2] K. Fisler，S。Krishnamurthi，LA Meyerovich和MC Tschantz，
“访问控制政策的验证和变更影响分析”
软件工程，2005。ICSE 2005.会议录。第27届国际
会议。IEEE，2005，pp.196-205。
[3] DP Guelev，M。Ryan和P.-Y. Schobbens， “模型检查访问
控制政策， “在国际学习中心，第一卷。3225. Springer，2004年，第219-230页
[4] G. Hughes和T. Bultan， “访问控制的自动验证
使用SAT求解器的政策， “国际软件工具杂志”
技术转让（STTT），第一卷。10，不。6，pp.503-520,2008。
[5] G. Kolaczek， “基于角色的约束的规范和验证
访问控制， “在支持技术：协作基础设施 -
rative Enterprises，2003x2。WET ICE 2003.会议记录。第十二届IEEE
国际研讨会。IEEE，2003，pp.190-195。
[6] N. Li，BN Grosof和J. Feigenbaum， “代表团逻辑：逻辑 -
基于分布式授权的方法， “ACM Transactions on on
信息与系统安全（TISSEC），第一卷。6，不。1，pp.128-171，
2003。
[7] MY Becker和P. Sewell， “Cassandra：灵活的信任管理，
适用于电子健康记录， “在计算机安全基础
研讨会，2004年。会议录。第17届IEEE。IEEE，2004，pp.139-154。
[8] J. DeTreville， “Binder，一种基于逻辑的安全语言” ，在安全和
隐私，2002年。会议录。2002年IEEE研讨会。IEEE，2002，
第105-113页。
[9] T. Jim， “SD3：具有认证评估的信任管理系统” ，
安全和隐私，2001年。标准普尔2001年。会议录。2001 IEEE
研讨会。IEEE，2001，pp.106-115。
[10] N. Li和JC Mitchell， “有限制的数据记录：基础
信任管理语言， “在Padl，vol。3. Springer，2003年，第58页 -
73。
[11] A. Anderson，A。Nadalin，B。Parducci，D。Engovatov，H。Lockhart，
M. Kudo，P。Humenn，S。Godik，S。Anderson，S。Crocker et al。， “Ex-
张力访问控制标记语言（xacml）版本1.0， “OASIS，
2003。
[12] P. Bonatti，S。De Capitani di Vimercati和P. Samarati， “代数为
组成访问控制策略， “ACM信息交易

政策，或者，给定具体的请求上下文，策略评估引擎允许用户测试访问控制。
相比之下，我们对SMT的形式化提供了能力
合理地推断所有有效政策的属性
请求上下文。

对于AWS服务的客户，Z ELKOVA提供更深入的服务
深入了解政策语言，语义及其含义
阳离子。 该工具使客户能够自动维护
他们的安全姿势。对于SMT中的人和验证
社区，这项工作展示了SMT如何验证属性

[13] 和系统安全（TISSEC），第一卷。5，不。3，pp.…，2002。
基于访问控制模型，"ACM信息与信息交易
系统安全（TISSEC），第一卷。4，不。3，pp.191-233,2001。
[14] K. Jayaraman，N。Bjorner，G。Outhred和C. Kaufman，"自动化
分析和调试网络连接策略，"微软
研究，技术。Rep.MSR-TR-2014-102,2014。
[15] J. Hopcroft，"用于最小化有限域中状态的n n n算法
自动机，"在机器和计算理论，Z。Kohavi和
A. Paz，Eds。Academic Press，New York，1971，pp.189-196。
[16] C. Barrett，CL Conway，M。Deters，L。Hadarean，D。Jovanovic，T。King
A. Reynolds和C. Tinelli，"CVC4"，参加国际会议
计算机辅助验证。Springer，2011年，第171-177页。

第9页

[17] L. de Moura和N.Bjørner，"Z3：一个有效的SMT求解器"，工具和
系统构建和分析的算法，第337-340页，
2008年。
[18] M. Berzish，V。Ganesh和Y. Zheng，"Z3str3：一个字符串求解器
理论启发式启发式，"计算机辅助设计中的形式化方法"
TPAMI，2017，第一卷。10，不。14，p。2017年5月55日。
[19] A. Aydin，L。Bang和T. Bultan，"基于自动机的模型计数
对于字符串约束，"在国际计算机辅助会议上
验证。Springer，2015年，第255-272页。