Ruby Mustache Templates RCE

惊鸿一瞥最是珍贵 / 2019-08-05 07:29:00 / 浏览数 3427 安全技术 漏洞分析

前言

Mustache被称为无逻辑的Ruby模板；然而，当客户端提供的模板呈现在服务器端时，会带来一些意想不到的副作用。
这篇文章探讨了在Ruby
web应用程序中发现一种(有点有限的)RCE形式，导致终端用户能够控制正在使用的Mustache模板变量。利用此RCE能够读取和删除所有客户数据，并且可能扩展为用来执行
提醒大家，这不是Mustache库或语法的漏洞，而是目标网站滥用库导致的RCE。
赏金计划目前不允许披露漏洞-鉴于bug的性质，可以使用一些Ruby示例来模拟漏洞。

Mustache模板简介

在例子中，我们将使用Ruby Mustache包，因为这是在目标上发现漏洞的位置。我们还将使用Ruby
ActiveSupport包，因为它同样存在于目标环境中，并且允许分配Ruby对象。
Mustache模板语法是一个Web模板系统，在大多数流行的编程语言中都有实现。Mustache缺乏流量控制系统。

```
irb(main):001:0> Mustache.render("Hello {{name}}!", name: "Rhys")
=> "Hello Rhys!"
```

您还可以提供对象，执行方法链访问嵌套值：

```
irb(main):001:0> user = User.first
=> true
irb(main):002:0> Mustache.render("Hello {{user.name}}!", user: user)
=> "Hello Rhys!"
```

目标

在测试目标时，我遇到了一个集成，其中PagerDuty
webhook被定向到目标上的服务，PagerDuty发出的任何事件警报都使用Mustache模板转换为面向客户的通信。
我很早就发现，用于接收PagerDuty
webhooks的目标端点没有经过身份验证，因此我很快就能够找到一种方法来模拟PagerDuty服务。这使得我可以充当PagerDuty，但我通常不会使用由PagerDuty
webhook系统生成的任意内容。

发现

在研究Mustache语法时，我很快发现Mustache并没有限制标签的使用。
我尝试了不同的值和转义，但都是徒劳的。但功夫不负有心人，当我渲染context的时候：

```
irb(main):001:0> Mustache.render("{{context}}")
=> "#<Mustache::Context:0x00007ffa3b8ef638>"
```

恍然大悟，我们在这里看到的是一个已分配对象的Ruby字符串表示，这意味着我们可以访问一个Ruby变量;这意味着当遇到用户内容时，这个库可能不是很安全。
鉴于模板是在服务器端渲染的，并且考虑到我可以查看Ruby对象并对其进行操作，这里是不是就存在一个RCE。然而，我很快确定Mustache变量名只能包含字母、数字和点
这意味着，我不能创建如下payload

```
context.class.ancestors.new.instance_method_send(:eval)
```

输入任意对象

然而，鉴于我能够提供用户输入(通过PagerDuty
webhook模拟)，并且考虑到我可以访问对象（如context示例所示），我决定尝试使用方法链接来处理对象本身的方法。

```
irb(main):001:0> Mustache.render("{{value.class}}", value: "Rhys")
=> "String"
```

然后我想起Active
Support#constantize方法。此方法是ActiveSupport提供的实用程序，它尝试使用字符串中指定的名称查找声明的常量。我认为这可能允许我与应用程序范围内的任意对

```
irb(main):001:0> Mustache.render("{{value.constantize.new}}", value: "Time")
=> "2019-07-31 12:31:54 -0700"
```

完美！这就是我想要的结果！

利用ActiveRecord模型

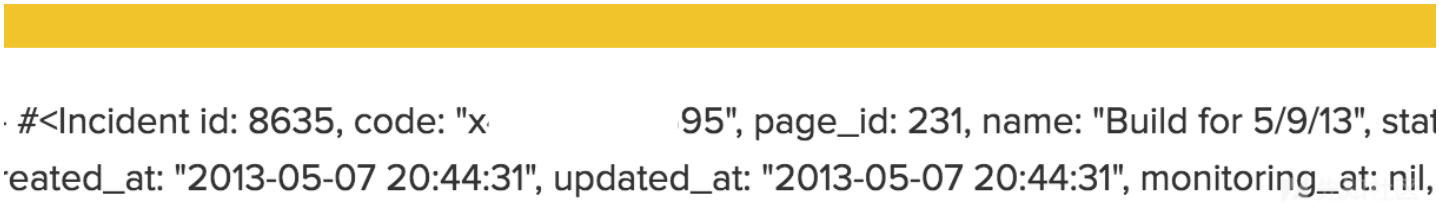
对于PoC，我决定证明可以使用我们自己的恶意Mustache模板
a）读取所有客户数据，
以及
b）删除所有客户数据。要实现这一点，可以使用我们使用方法链接对任意Ruby类进行操作的能力来锁定应用程序的ActiveRecord模型。ActiveRecord是为Rails应用程序提
我首先创建了一个名为Incident的PagerDuty事件，并更新了我的模板，使其具有以下内容。

```
{{ incident.name.constantize.all.first }}
```

细分，这个方法链执行以下操作：

1.incident.name：从伪造的PagerDuty webhook获取值。本例中，值是Incident。
2.constanalize：尝试使用前面的方法中的名称查找声明的常量。在本例中，先前值是Incident-我们希望定位的ActiveRecord模型的名称。
3.all：对事件模型的所有记录执行SQL SELECT操作。
4.first：将结果限制为一条记录

然后我调用了我的假webhook，在面向公众的网站上刷新页面时，遇到了以下输出。



好极了！这就是RCE的一个真实模拟，还应该注意的是，典型的ActiveRecord方法(如#delete_all)也可用，这意味着我们可以使用此bug删除所有数据。
需要提供参数的方法(如#update)不可用-因为Mustache验证器不允许形成有效语法所需的字符。

结论

此bug的根本原因在于服务器端模板库在主Web应用程序实例上渲染用户提供的模板。安全的做法是，模板应该是a)在沙箱中渲染-例如AWS
Lambda，b)过滤不适当的变量，或c)使用Javascript库实现，不暴露Ruby点表示法。
自报告bug以来，该程序引入了模板标签的安全列表，并允许在通信模板中使用。我还没找到绕过方法

■■■■■https://rhys.io/post/rce-in-ruby-using-mustache-templates

上一篇：Linux Kernel Expl... 下一篇：VxWorks 高危漏洞 CVE-...
1. 0 条回复
   - 动动手指，沙发就是你的了！