Red Hat Ansible Automation Platform 2.4

Red Hat Ansible 安全自动化指南

使用 Ansible 识别和管理安全事件

Last Updated: 2024-10-24

Red Hat Ansible Automation Platform 2.4 Red Hat Ansible 安全自动化指南

使用 Ansible 识别和管理安全事件

法律通告

摘要

本指南提供与使用 Ansible 相关的识别、分类和响应安全事件的信息。

目录

对红帽文档提供反馈

如果您对本文档有任何改进建议，或发现错误，请通过 https://access.redhat.com 联系技术支持来创建一个请求。

第 1 章 使用 ANSIBLE 安全自动化进行防火墙策略管理

作为安全操作员，您可以使用 Ansible 安全自动化来管理多个防火墙策略。创建和删除防火墙规则，以阻止或取消阻止源 IP 地址访问目标 IP 地址。

1.1. 关于防火墙策略管理

机构的网络防火墙是防御安全攻击的第一道防线，也是维护安全环境的一个重要组件。作为安全操作员，您可以构建和管理安全网络，以确保防火墙仅允许由机构的防火墙策略定义的入站和出站网络流量。防火墙策略包含保护网络免受恶意传入和传出流量的安全规则。

对于安全团队而言，针对不同产品和供应商管理多个防火墙规则既具有挑战性又耗时。涉及复杂任务的手动工作流流程可能会导致错误，并最终导致在调查可疑行为或防止服务器被攻击时出现延迟。当安全组合中的每个解决方案都通过相同的语言实现自动化时，安全分析师和运营商只需很少的时间就可跨各种产品执行一系列操作。这一自动化流程可最大限度地提高安全团队的整体效率。

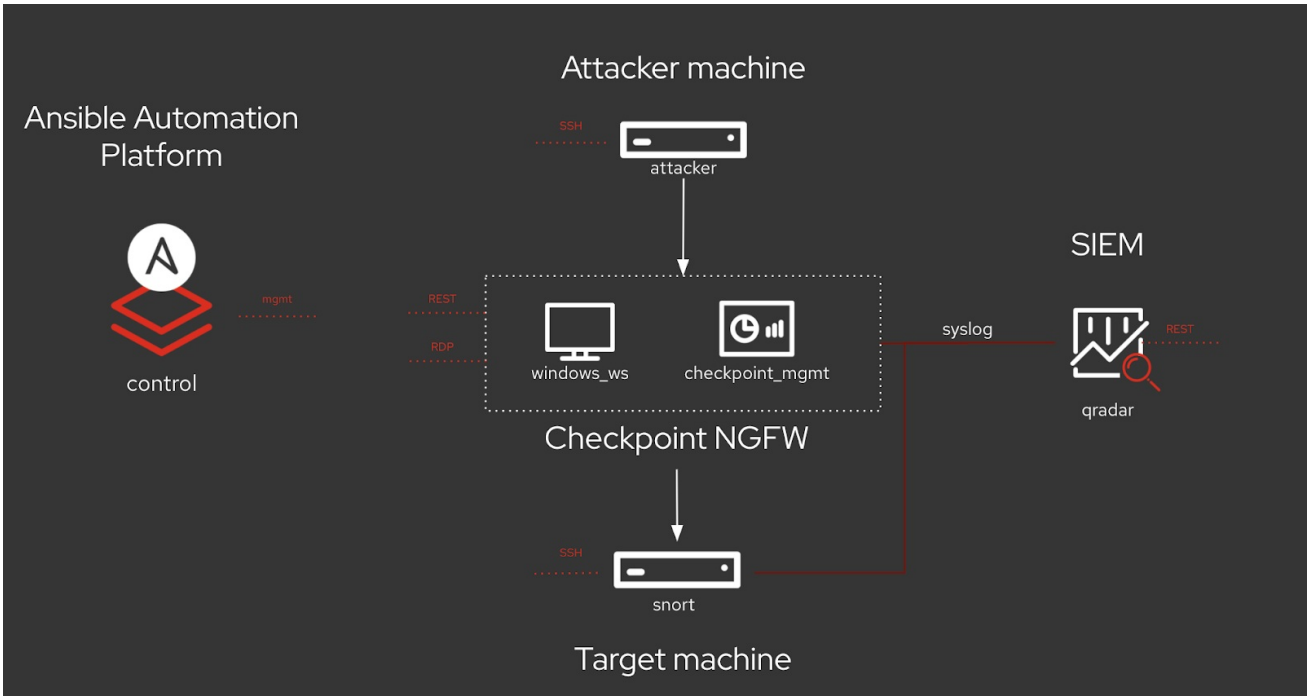
Ansible 安全自动化与来自供应商的各种安全技术交互。Ansible 支持安全团队以统一的方式管理不同的产品、接口和工作流，从而成功进行部署。例如，您的安全团队可以自动化任务，如在企业防火墙上阻塞和取消阻塞 IP 和 URL。

1.2. 自动防火墙规则

Ansible 安全自动化使您可以自动执行各种防火墙策略，这些策略需要跨各种产品执行一系列操作。您可以使用 Ansible 角色（如 acl_manager 角色）管理许多防火墙设备的访问控制列表 (ACL)，如阻塞或取消阻塞 IP 或 URL。角色允许您根据已知文件结构自动加载相关的变量、文件、任务、处理程序和其他 Ansible 构件。在将自己的内容分组到角色后，您可以轻松地重复使用它们，并将它们与其他用户共享。

以下实验环境是现实企业安全架构的简化示例，该架构可能更为复杂，并包含其他特定于供应商的工具。在典型的事件响应场景中，您会收到入侵警报，并立即执行带有 acl_manger 角色来阻止攻击者 IP 地址的 playbook。

您的整个团队都可使用 Ansible 安全自动化来处理调查、威胁黑客和事件响应。Red Hat Ansible Automation Platform 为您提供了可轻松在安全团队内使用和重复使用的已认证内容集合。



其他资源

如需有关 Ansible 角色的更多信息，请参阅 docs.ansible.com 上的角色。

1.2.1. 创建新的防火墙规则

使用 acl_manager 角色创建新的防火墙规则，以阻止源 IP 地址访问目标 IP 地址。

先决条件

- 已安装最新版本的 ansible-core。
- 您可以访问检查点管理服务器来强制执行新策略

流程

1. 使用 ansible-galaxy 命令安装 acl_manager 角色。

```
$ ansible-galaxy install ansible_security.acl_manager
```

2. 创建一个新 playbook，再设置以下参数：例如，源对象、目的地对象、两个对象与您要管理的实际防火墙之间的访问规则，如 Check Point：

```
- name: block IP address
  hosts: checkpoint
  connection: httpapi

  tasks:
    - include_role:
        name: acl_manager
        tasks_from: block_ip
      vars:
        source_ip: 172.17.13.98
        destination_ip: 192.168.0.10
        ansible_network_os: checkpoint
```

3. 运行 playbook **$ ansible-navigator run --ee false <playbook.yml>**。

```
PLAY [checkpoint] **************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************
ok: [checkpoint]

TASK [include_role : acl_manager] **********************************************************************************

TASK [acl_manager : include_tasks] *********************************************************************************
included: /home/student1/.ansible/roles/acl_manager/tasks/providers/checkpoint/block_ip.yaml for checkpoint

TASK [acl_manager : Search source IP host object] ******************************************************************
ok: [checkpoint]

TASK [acl_manager : Create source IP host object] ******************************************************************
skipping: [checkpoint]

TASK [acl_manager : Search destination IP host object] *************************************************************
ok: [checkpoint]

TASK [acl_manager : Create destination IP host object] *************************************************************
skipping: [checkpoint]

TASK [acl_manager : Create access rule to deny access from source to destination] **********************************
changed: [checkpoint]

PLAY RECAP *********************************************************************************************************
checkpoint                 : ok=5    changed=1    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
```

验证

您已创建了一条阻止源 IP 地址访问目标 IP 地址的新防火墙规则。访问 MGMT 服务器，并验证是否已创建了新的安全策略。

其他资源

有关安装角色的更多信息，请参阅从 Galaxy 安装角色。

1.2.2. 删除防火墙规则

使用 acl_manager 角色删除安全规则。

先决条件

- 已安装 Ansible 2.9 或更高版本
- 您可以访问防火墙 MGMT 服务器来强制执行新策略

流程

1. 使用 ansible-galaxy 命令安装 acl_manager 角色：

```
$ ansible-galaxy install ansible_security.acl_manager
```

2. 使用 CLI，创建一个具有 acl_manger 角色的新 playbook，并设置参数（如源对象、目标对象、访问两个对象之间的访问规则）：

```
- name: delete block list entry
  hosts: checkpoint
  connection: httpapi

    - include_role:
        name: acl_manager
        Tasks_from: unblock_ip
      vars:
        source_ip: 192.168.0.10
        destination_ip: 192.168.0.11
        ansible_network_os: checkpoint
```

3. 运行 playbook $ ansible-navigator run --ee false <playbook.yml>：

```
PLAY [checkpoint] ***************************************************************************************************

TASK [Gathering Facts] **********************************************************************************************
ok: [checkpoint]

TASK [include_role : acl_manager] ***********************************************************************************

TASK [acl_manager : include_tasks] **********************************************************************************
included: /home/student1/.ansible/roles/acl_manager/tasks/providers/checkpoint/block_ip.yaml for checkpoint

TASK [acl_manager : Search source IP host object] *******************************************************************
ok: [checkpoint]

TASK [acl_manager : Create source IP host object] *******************************************************************
skipping: [checkpoint]

TASK [acl_manager : Search destination IP host object] **************************************************************
ok: [checkpoint]

TASK [acl_manager : Create destination IP host object] **************************************************************
skipping: [checkpoint]

TASK [acl_manager : Create access rule to deny access from source to destination] ***********************************
changed: [checkpoint]

TASK [include_role : acl_manager] ***********************************************************************************

TASK [acl_manager : include_tasks] **********************************************************************************
included: /home/student1/.ansible/roles/acl_manager/tasks/providers/checkpoint/unblock_ip.yaml for checkpoint

TASK [acl_manager : Delete access rule that deny access from source to destination] *********************************
changed: [checkpoint]

PLAY RECAP **********************************************************************************************************
checkpoint                 : ok=7    changed=2    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
```

验证

您已删除防火墙规则。访问 MGMT 服务器，并验证新的安全策略是否已移除。

其他资源

有关安装角色的更多信息，请参阅从 Galaxy 安装角色。

第 2 章 使用 ANSIBLE 自动执行网络入侵检测和观察系统 (IDPS)

您可以使用 Ansible 自动执行入侵检测和构建系统 (IDPS)。对于本指南，我们使用 Snort 作为 IDPS。使用 Ansible 自动化中心来使用内容集合，如任务、角色和模块等，以创建自动化工作流。

2.1. 要求和先决条件

在开始使用 Ansible 自动执行 IDPS 之前，请确保您具有成功管理 IDPS 所需的正确安装和配置。

- 已安装 Ansible-core 2.15 或更高版本。
- SSH 连接和密钥已配置。
- IDPS 软件 (Snort) 已经安装并配置。
- 您可以访问 IDPS 服务器 (Snort) 来强制执行新策略。

2.1.1. 验证您的 IDPS 安装

要验证 Snort 是否已成功配置，请通过 **sudo** 调用它并要求提供版本：

```
$ sudo snort --version

 ,,_     -*> Snort! <*-
o"  )~   Version 2.9.13 GRE (Build 15013)
""    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
      Copyright (C) 2014-2019 Cisco and/or its affiliates. All rights reserved.
      Copyright (C) 1998-2013 Sourcefire, Inc., et al.
      Using libpcap version 1.5.3
      Using PCRE version: 8.32 2012-11-30
      Using ZLIB version: 1.2.7
```

验证该服务是否正在通过 **sudo systemctl** 活跃运行：

```
$ sudo systemctl status snort
● snort.service - Snort service
   Loaded: loaded (/etc/systemd/system/snort.service; enabled; vendor preset: disabled)
   Active: active (running) since Mon 2019-08-26 17:06:10 UTC; 1s ago
  Main PID: 17217 (snort)
   CGroup: /system.slice/snort.service
           └─17217 /usr/sbin/snort -u root -g root -c /etc/snort/snort.conf -i eth0 -p -R 1 --pid-
path=/var/run/snort --no-interface-pidfile --nolock-pidfile
[...]
```

如果 Snort 服务没有运行，使用 **systemctl restart snort** 重新启动它并重新检查状态。

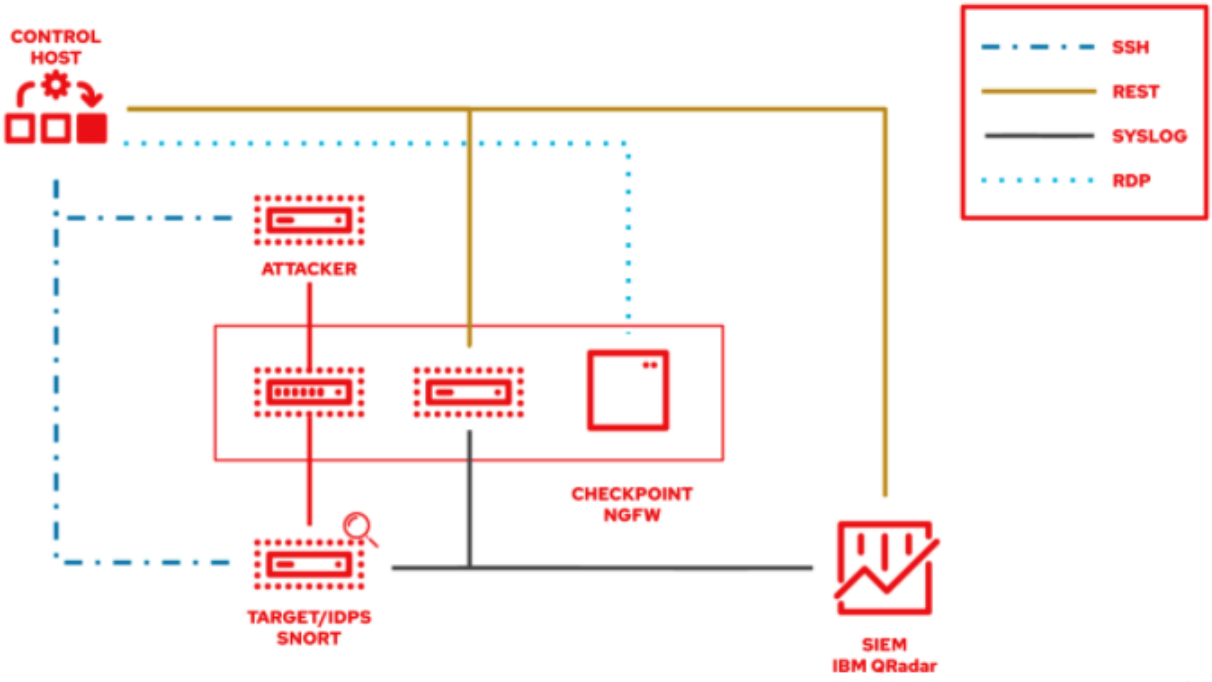
确认服务处于活动状态后，通过同时按 **CTRL** 和 **D** 退出 Snort 服务器，或者在命令行中键入 **exit**。所有进一步交互都将从 Ansible 控制主机通过 Ansible 进行。

2.2. 使用 ANSIBLE 自动执行您的 IDPS 规则

要自动化您的 IDPS，请使用 **ids_rule** 角色创建和更改 Snort 规则。Snort 使用基于规则的语言来分析网络流量并将其与给定规则集进行比较。

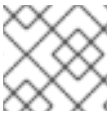
以下实验环境演示了 Ansible 安全自动化集成的样子。名为"Attacker"的计算机在运行 IDPS 的目标计算机上模拟潜在的攻击模式。

请记住，现实世界的设置会包含其他供应商和技术。



2.2.1. 创建新的 IDPS 规则

使用 **ids_rule** 角色管理您的规则和 IDPS 的签名。例如，您可以设置一条新规则，该规则将查找与防火墙上之前的攻击一致的特定模式。



注意

目前，**ids_rule** 角色只支持 Snort IDPS。

先决条件

- 您需要 **root** 特权才能在 Snort 服务器上进行任何更改。

流程

1. 使用 ansible-galaxy 命令安装 **ids_rule** 角色：

```
$ ansible-galaxy install ansible_security.ids_rule
```

2. 新建一个名为 **add_snort_rule.yml** 的 playbook 文件。设置以下参数：

```
- name: Add Snort rule
  hosts: snort
```

3. 添加 **become** 标志，以确保 Ansible 处理特权升级。

```
- name: Add Snort rule
  hosts: snort
  become: true
```

4. 通过添加以下变量来指定 IDPS 供应商的名称：

```
- name: Add Snort rule
  hosts: snort
  become: true

  vars:
    ids_provider: snort
```

5. 将以下任务和特定于任务的变量（如规则、Snort 规则文件以及规则 - present 或 absent 的状态）添加到 playbook 中：

```
- name: Add Snort rule
  hosts: snort
  become: true

  vars:
    ids_provider: snort

  tasks:
    - name: Add snort password attack rule
      include_role:
        name: "ansible_security.ids_rule"
      vars:
        ids_rule: 'alert tcp any any -> any any (msg:"Attempted /etc/passwd Attack";
uricontent:"/etc/passwd"; classtype:attempted-user; sid:99000004; priority:1; rev:1;)'
        ids_rules_file: '/etc/snort/rules/local.rules'
        ids_rule_state: present
```

任务是在目标计算机上更改的组件。由于您使用定义这些任务的角色，**include_role** 是唯一需要的条目。

ids_rules_file 变量指定 **local.rules** 文件的定义位置，而 **ids_rule_state** 变量则表示如果该规则不存在，则应创建该规则。

6. 运行以下命令来运行 playbook：

```
$ ansible-navigator run add_snort_rule.ym --mode stdout
```

运行 playbook 后，除新创建的规则外，所有任务还将执行。您的 playbook 输出将确认您的 PLAY、TASK、RUNNING HANDLER 和 PLAY RECAP。

验证

要验证您的 IDPS 规则是否已成功创建，通过 SSH 连接到 Snort 服务器并查看 **/etc/snort/rules/local.rules** 文件的内容。