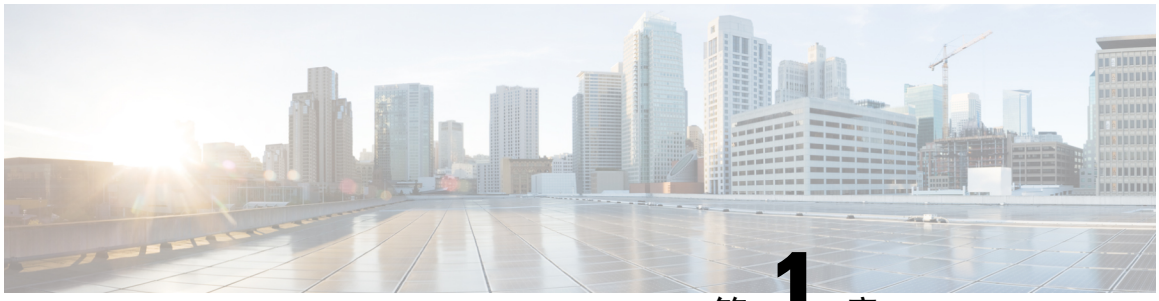
Snort 3 检查器参考

上次修改日期: 2025 年 2 月 27 日

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

目录



第 1 章

简介

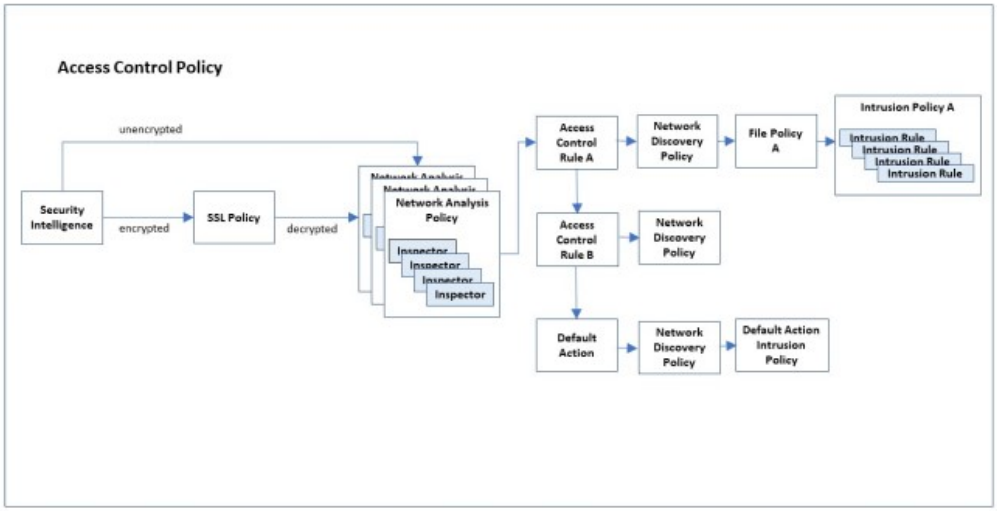
关于 Snort 3 检测

Snort 入侵防御系统 (IPS) 实时分析网络流量以提供深度数据包检查。Snort 可以检测并阻止流量异常以及网络探测和攻击。Snort 3 是 Snort 的最新版本。有关详细信息，请参阅https://snort.org/snort3。

Snort 是为高性能和可扩展性而设计的。Snort 包含一组称为 检查器的可配置插件。Snort 检查器可以检测和分析特定类型网络协议或探测的流量，规范化消息以增强数据包分析，并检查消息中嵌入的特定类型的文件。您可以在网络分析策略 (NAP) 中配置 Snort 检查器，并在入侵策略中启用入侵规则。

访问控制策略

访问控制策略在多个阶段处理流量。下图展示策略部署的示例。本文档中介绍的元素是入侵规则中使用的 Snort 3 检查器和规则选项（均以蓝色突出显示）。



通过网络分析策略，您可以配置 Snort 3 检查器，以确定流量协议并提取和规范化数据。可以配置多个网络分析策略，每个策略都使用唯一配置的一组 Snort 3 检查器对数据进行规范化。检查器在检测到数据流中的异常情况时可以发出警报，但其主要用途是为入侵规则准备数据。入侵策略应用其配置的入侵规则来检查数据中是否存在逃逸、入侵或攻击迹象。

在网络分析策略中，您可以通过设置处理协议的检查器特定的配置参数使用给定协议自定义该数据的检测行为。例如，要配置 POP 数据的检测行为，请为 POP 检查器设置配置参数。

也可以通过使用某些协议特定的规则选项编写自定义入侵规则来自定义某些协议的入侵策略。

如果使用多个网络分析策略和多个入侵策略建立复杂配置，系统会首先选择网络分析策略来处理数据。在网络分析策略应用相应的检查器来执行其分析后，数据不会自动传递到该协议的相应入侵策略。访问控制策略执行其他测试，以确定哪个入侵策略可获取数据。因此，在配置访问控制、网络分析和入侵策略时，请确保使用正确的网络分析和入侵策略对分析数据。有关详细信息，请参阅《Cisco Secure Firewall Management Center Snort 3 配置指南》。

入侵规则更新

思科定期以轻量级安全包 (LSP) 的形式发出入侵规则更新。这些更新可能会更改 Snort 3 检查器的配置参数和入侵规则选项的默认值。

检查器配置

您可以通过 Cisco Secure Firewall Management Center Web 界面启用和禁用 Snort 检查器，以及查看和更改其配置。 Cisco Secure Firewall Management Center Web 界面使用 JSON 格式描述检查器配置。有关详细信息，请参阅《Cisco Secure Firewall Management Center Snort 3 配置指南》。

要使用检查器，您必须通过 管理中心 Web 界面启用它。此外，对于服务检查器，您必须在 绑定程序 检查器中为服务检查器配置一个条目。有关详细信息，请参阅绑定程序检查器概述，第 13 页。

Snort 3 检查器参考反映了 Snort 3 检查器参数和内置入侵规则选项的默认设置。您的系统可能使用不同的默认值，具体取决于 LSP 更新或系统随附的基本网络访问策略。要最准确地了解网络访问策略的检查器设置，请在 管理中心 Web 界面中查看设置。

Snort 3 检查器简介

Snort 3 检查器是分析和规范化数据包的插件（类似于 Snort 2 预处理器）。Snort 3 中的检查器和设置列表不直接对应于 Snort 2 中的预处理器和设置列表。

Snort 3 检查器

- 流 ICMP 检查器， 第 189 页
- 流 IP 检查器， 第 191 页
- 流 TCP 检查器， 第 195 页
- 流 UDP 检查器， 第 207 页
- Telnet 检查器， 第 209 页

对于每个 Snort 3 检查器，本文档介绍了以下内容：

- 有关检查器的用途和功能的一般信息。
- 检查器的类型：
  - 服务：检查器分析互联网服务协议（HTTP、FTP、TCP 或 UDP）中使用的协议数据单元 (PDU)。示例包括： `http_inspect`、 `ftp_server`。
  - 被动：仅提供配置（`ftp_client`、 `ftp_server`）或促进其他处理（绑定程序）的检查器。
  - 数据包：在其他检查器进行处理之前对原始数据包执行处理的检查器。示例包括： 规范器。
  - 探测：检查器在所有检测完成后对所有数据包执行处理。示例包括： `port_scan`。
  - 流：执行流跟踪、互联网协议分片重组和 TCP 重组的检查器。示例包括： `stream_tcp`、`stream_ip`。
  - 基本模块：一个可配置的内置 Snort 3 组件，提供支持多种流量检测过程的功能。示例包括： `rate_filter`。
- 用法：
  - 检查器：在网络分析策略 (NAP) 中配置这些检查器。示例包括： `imap`、 `ssh`。
  - 全局、情景：配置这些检查器一次。示例包括： `port_scan`、`rate_filter`。
- 实例类型：
  - 单一实例：为网络访问策略中的单个实例配置这些检查器。有关详细信息，请参阅单例检查器， 第 5 页。
  - 多例：为网络访问策略 (NAP) 中的多个实例配置这些检查器。一个 NAP 可以包含多个以网络、端口或 VLAN 来区分的实例。每个实例都是唯一的，用于处理特定的流量分段。有关详细信息，请参阅多例检查器， 第 5 页。
- 需要其他检查器：许多检查器依赖其他检查器才能完全处理数据流。当检查器要求配置其他检查器时，文档会指定这些额外的检查器。
- 配置检查器的最佳实践：这些是针对每个检查器的最佳性能建议。
- 检查器的配置参数：您可以在 管理中心 Web 界面中的 **Policies>Access Control>Network Analysis Policy>*Policy Name*>Snort 3 Version>*Inspector Name*** 设置配置参数。

注释 在修改检查器参数之前，建议您了解检查器与已启用的入侵规则之间的交互。

- 规则：Snort 3 检查器使用规则生成事件。内置规则可能包含类类型、引用和其他元数据。
- 入侵规则选项：通过定义由检查器处理的数据类型的入侵规则选项来自定义入侵规则。有关管理自定义入侵规则的信息，请参阅 《Cisco Secure Firewall Management Center Snort 3 配置指南》。

注释 编写自定义入侵规则是一项高级活动，必须谨慎执行。您可能需要使用本文档中未介绍的检查器和规则选项。使用本文档中介绍的某些检查器和入侵规则选项需要对 Snort 开源文档中记录的检查器和规则选项进行特定设置。某些规则选项会对Snort快速模式匹配器或检测光标的位置产生影响。有关详细信息，请参阅 https://www.snort.org/snort3处提供的 Snort 3 开源文档。

单例检查器

网络访问策略 (NAP) 只能使用单例检查器的一个实例。

- 单一检查器不像多例检查器一样支持每个 NAP 多个实例。
- 单一检查器可能不适用于某些特定流。

例如：

```
{
   "normalizer":{
      "enabled":true,
      "type":"singleton",
      "data":{
         "ip4":{
            "df":true
         }
      }
   }
}
```

多例检查器

网络访问策略可以使用一个或多个多例检查器实例，您可以根据需要进行配置。多实例检查器支持根据特定条件（例如网络、端口和 VLAN）配置设置。一组受支持的设置包含实例。多示例提供一个默认实例，您还可以根据特定条件添加其他实例。如果流量符合自定义实例中的条件，则应用自定义实例中的设置。否则，将应用默认实例中的设置。默认实例的名称与检查器的名称相同。

您还可以创建新实例，但请确保为您创建的每个新实例包含 绑定程序 条件，以避免错误。

例如：

- 修改了默认实例的多例检查器：

```
{
   "http_inspect":{
      "instances":[
         {
            "name":"http_inspect",
            "data":{
               "response_depth":5000
            }
         }
      ]
   }
}
```

- 修改默认实例和默认 绑定程序 的多例检查器：

```
{
   "http_inspect":{
      "instances":[
         {
            "name":"http_inspect",
            "data":{
               "response_depth":5000
            }
         }
      ]
   },
   "binder":{
      "rules":[
         {
            "use":{
               "type":"http_inspect"
            },
            "when":{
               "role":"any",
               "ports":"8080",
               "proto":"tcp",
               "service":"http"
            }
         }
      ]
   }
}
```

- 多例检查器，其中添加了自定义实例和自定义 绑定程序:

```
{
   "http_inspect":{
      "instances":[
         {
            "name":"http_inspect1",
            "data":{
               "response_depth":5000
            }
         }
      ]
   },
   "binder":{
      "rules":[
         {
            "use":{
               "type":"http_inspect",
               "name":"http_inspect1"
            },
```

```
            "when":{
               "role":"any",
               "ports":"8080",
               "proto":"tcp",
               "service":"http"
            }
         }
      ]
   }
}
```

Snort 3 中的协议和服务标识

绑定程序 检查器执行影响所有 Snort 服务检查器的独特功能。 绑定程序 与 Snort 向导 模块一起确定哪个流或服务检查器可以检查网络流量。绑定程序 检查器中的配置包括端口、主机、CIDR 以及定义相同网络分析策略中的另一个检查器何时需要检查流量的服务。

该 向导 支持与端口无关的服务配置，可以检测恶意软件命令和控制信道。

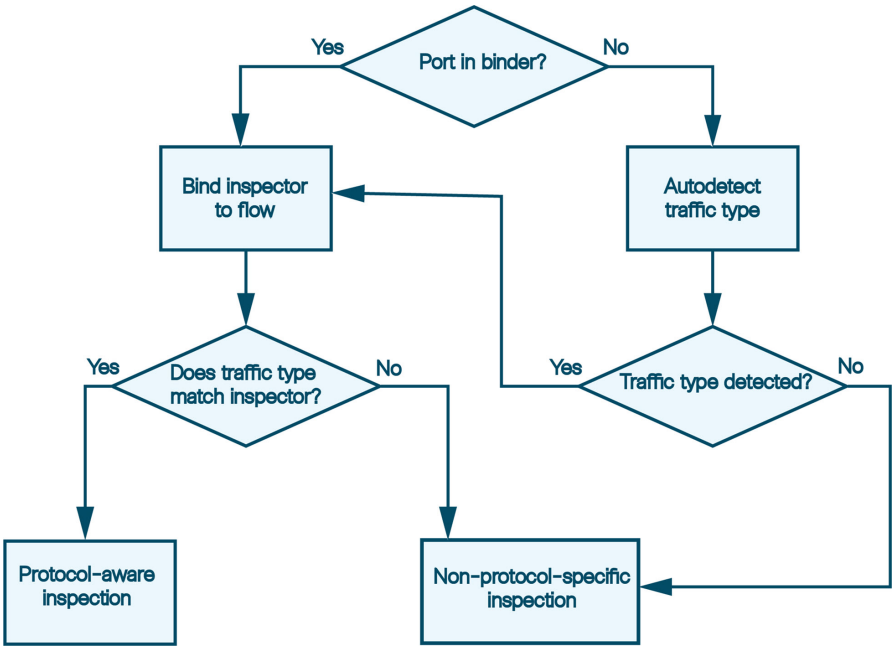
注释 您无法通过 Cisco Secure Firewall Management Center Web 界面配置该 向导 。

当流量到达防火墙设备时，绑定程序检查器会搜索入侵策略并选择要应用的适当网络访问策略(NAP)。在 NAP 中， 绑定程序 会确定用于数据流的相应流和服务检查器。稍后，如果与流关联的服务发生更改，NAP 将使用 绑定程序 选择不同的服务检查器。

绑定程序 检查器配置包括描述流量特征的 when 参数和 use 参数，这些参数用于指定要应用于匹配这些特征的流量的检查器。在确定哪个检查器应用于数据流时， 绑定 检查器会按从上到下的顺序将流量与其 when 子句进行比较，并应用与匹配流量的第一个 when 子句对应的 use 子句。

如果没有与数据流匹配的特定 绑定 程序条件， 向导 会分析数据流以确定服务。该 向导 调用绑定 程序 ，为该服务选择适当的检查器。如果无法识别任何服务， 绑定程序 通常会将流检查器绑定到流，并且系统会对数据包执行非协议特定的重组，而不考虑负载内容。

下图说明了检查器如何执行特定于协议或非特定于协议的检测。服务检测取决于绑定程序检查器中配置 端口、 主机、 服务 和 CIDR 参数的方式：



您可以通过在 管理中心 Web 界面中为 NAP 定义 绑定 检查器中的 use 和 when 参数，来自定义检查器选择条件。有关所 绑定程序 参数的详细信息，请参阅 绑定程序检查器概述，第 13 页。有关导航 管理中心 Web 接口以配置检查器的信息，请参阅 《Cisco Secure Firewall Management Center Snort 3 配置指南》。

如果 绑定程序 配置不正确，它就无法检测流的服务，也无法与其绑定检查器。如果规则引擎和自动检测无法理解和识别流量，配置 何时 条件（例如 绑定程序 检查器中的端口）不会强制执行检查。例如，如果在 绑定程序 中将端口 88 配置为 HTTP 端口， 则绑定程序 会将 http_inspect 检查器绑定到该端口上的任意数据流。但是，如果数据流不是HTTP，则规则引擎不会将数据作为HTTP进行检查，而是执行基于端口的检测。

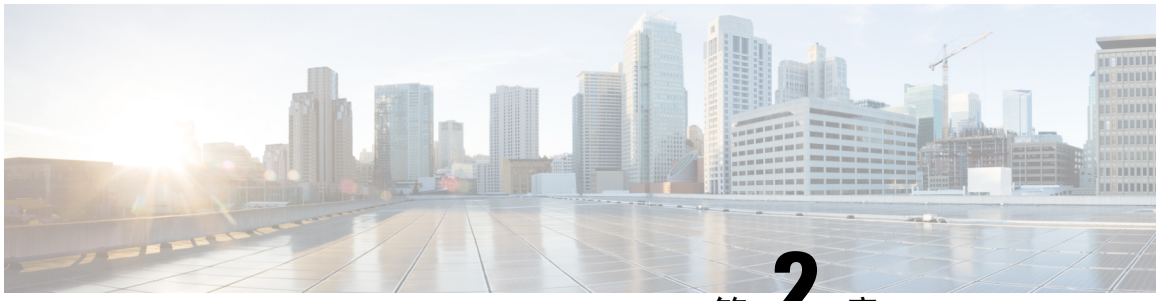
网络分析策略中的自动检测以及启用或禁用检查器

自动检测的行为会发生变化，具体取决于在网络分析策略中启用还是禁用目标检查器。如果在网络分析策略中启用了目标检查器，则自动检测将按上述方式工作。

如果在网络分析策略中禁用目标检查器，通常情况下，自动检测仍会将流检查器（例如流 TCP 或流 UDP）绑定到流。但是，规则引擎不执行服务检测或检测。对于 TCP 流，流 TCP 检查器会执行重组。

第 I 部分

Snort 3 检查器



第 2 章

ARP 欺骗检查器

ARP 欺骗检查器概述

类型	检查器（网络）
使用方式	检测
实例类型	单例对象
所需其他检查器	无
已启用	`true`

地址解析协议 (ARP) 是一种无状态通信协议，在单个网络中用于地址解析。交换请求和响应时，ARP 不提供主机之间的身份验证。

ARP 欺骗是一种在局域网 (LAN) 中使用 ARP 的中间人攻击。攻击者通过拦截发往特定主机介质访问控制 (MAC) 地址的消息来更改与主机的通信。

`arp_spoof` 检查器分析 ARP 数据包并检测单播 ARP 请求。为了检测 ARP 缓存覆盖攻击，ARP 欺骗检查器会识别不一致的以太网到 IP 映射。

如果启用， `arp_spoof` 检查器：

- 检查以太网地址和 ARP 数据包中的地址。出现不一致时，检查器使用规则 112:2 或规则 112:3 生成警报，并在内联部署中丢弃违规数据包。
- 检查单播 ARP 请求。如果检测到单播 ARP 请求，检查器使用规则 112:1 生成警报，并且在内联部署中，检查器会丢弃违规数据包。

- 如果指定了 `host[]` 参数，则检查器会使用该信息检测 ARP 缓存覆盖攻击。如果检测到此类攻击，检查器会使用规则 112:4 生成警报，并在内联部署中丢弃违规数据包。

ARP 欺骗检查器参数

`arp_spoof` 检查器不会在 Cisco Secure Firewall Management Center Web 界面中提供默认配置参数值。

ARP 欺骗检查器规则

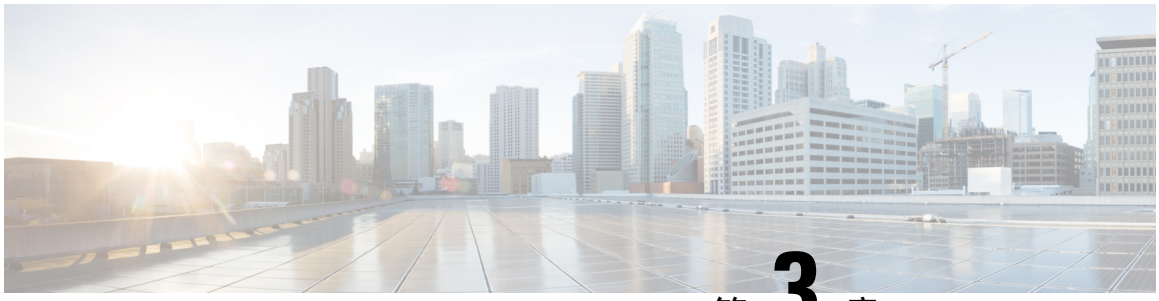
对 生成事件并在内联部署中丢弃攻击性数据包启用 `arp_spoof` 检查器规则。

表 *1: ARP* 欺骗检查器规则

GID:SID	Rule Message
112:1	单播 ARP 请求
112:2	源 的以太网/ARP 不匹配请求
112:3	对目标的以太网/ARP 不匹配请求
112:4	未尝试的 ARP 缓存覆盖攻击

ARP 欺骗检查器入侵规则选项

`arp_spoof` 检查器没有任何入侵规则选项。



第 3 章

绑定检查器

绑定程序检查器概述

类型	检查器 (被动)
使用方式	检测
实例类型	单例对象
所需其他检查器	取决于已建立的绑定
已启用	`true`

每个网络分析策略 (NAP) 都有一个 绑定程序 检查器。绑定程序 确定何时需要使用特定服务检查器来检查流量。绑定程序 检查器中的配置包括端口、主机、CIDR 以及定义相同网络分析策略中的另一个检查器何时需要检查流量的服务。当 绑定 程序规则与新流匹配时，目标检查器将绑定到该流。

绑定程序 检查器可以与自动检测 向导 配合使用，执行与端口无关的服务配置以及恶意软件命令和控制信道的检测。有关详细信息，请参阅Snort 3 中的协议和服务标识， 第 7 页。

在会话开始时评估绑定，如果在会话中识别到适当的服务，则会再次评估绑定。绑定是按从上到下的顺序评估的使用时间规则列表。Snort 会使用第一个匹配的网络和服务配置来检查流量。

示例

例如，如果要配置 NAP 以检查 CIP 流量：

- 在NAP的 绑定程序 检查器中，使用要检查的流量的正确端口、角色和协议信息更新 `"type":"cip"` 部分。
- 在 `cip` 检查器中查看同一 NAP 的默认值，并进行检查所需的任何调整以检查 CIP 流量。

以下是 `cip` 配置和绑定的示例。此示例使用 绑定程序检查器参数，第 16 页中所述的选项。

```
{
   "use": {
     "type":"cip"
   },
   "when": {
     "proto":"udp",
     "ports":"22222 33333",
     "role":"server"
   }
 },
 {
   "use": {
     "type":"cip"
   },
   "when": {
     "role":"server",
     "ports":"44818",
     "proto":"tcp"
   }
 },
```

自动检测无端口配置的服务

自动检测 向导 支持与端口无关的服务配置以及恶意软件命令和控制信道的检测。当流量到达时， 绑定程序 检查器会首先将自动检测 向导 附加到流，然后检查初始负载以确定流量正在使用的服务。例如， `GET` 将表示 HTTP， `HELO` 将表示 SMTP。确定服务后，Snort 会将相应的服务检查器绑定到流，并从流中分离自动检测 向导 。

注释 您不能通过 Cisco Secure Firewall Management Center Web 接口配置自动检测 向导 。

如果规则引擎和自动检测 向导 无法理解和识别流量，则在 绑定程序 检查器中配置端口不会强制执行检测。

自动检测和绑定程序配置

绑定程序 检查器按从上到下的顺序匹配入侵规则，并应用第一条规则来匹配流量。如果尚未为流中检测到的服务配置 绑定程序 检查器，则自动检测向导仍可以将流绑定到相关检查器。例如：

- 如果负载是 `GET` ，并且自动检测向导将流量类型识别为 HTTP，则 绑定程序 检查器会将 HTTP 检查器绑定到该流。
- 如果无法识别流量类型，规则引擎将执行非特定于协议的检测。

如果端口配置不正确， 绑定程序 检查器无法自动检测该流的服务，也无法向其绑定检查器。例如，如果将端口 88 配置为绑定程序中的 HTTP 端口，则 绑定程序 检查器会将 HTTP 检查器绑定到该端口上的任意数据流。但是，如果流不是 HTTP，则规则引擎不会将其作为 HTTP 进行检查。相反，侦查和检测将超时。

自动检测以及网络分析策略中检查器的启用或禁用

自动检测的行为会发生变化，具体取决于在网络分析策略中启用还是禁用目标检查器。如果在网络分析策略中启用了目标检查器，则自动检测将按预期工作。

如果在网络分析策略中禁用目标检查器，通常情况下，自动检测仍会将流检查器（例如流 TCP 或流 UDP）绑定到流。但是，规则引擎不执行服务检测或检测。对于 TCP 流，流 TCP 检查器会执行重组。

配置绑定程序检查器的最佳实践

配置绑定程序检查器时，请考虑以下最佳实践：

- 除非检查器需要，否则不要在绑定程序检查器中配置端口。如果规则引擎可以自动检测流量，则端口配置不会提高效率。但是，端口配置不正确可能会导致无法检测规避。
- 仅为一个检查器配置一个端口。如果同一个端口在绑定器中为不同的协议和检查器配置两次，它将自动触发第一个检查器。
- 如果在默认 绑定程序 检查器配置中看不到某个服务检查器的配置，请将其添加到 绑定程序 检查器。例如，如果要使用 `cip` 检查器，请将 `cip` 检查器的 `use` 和 `when` 选项添加到绑定程序。
- 对于流 TCP 检查器，将网络配置为自定义绑定操作系统配置。网络配置适用于所有端口。
- 对于服务检查器，如果绑定程序可以自动检测流中的协议，请避免硬端口绑定。如果协议不可检测，则硬端口绑定不能确保进行检测和检查。

需要端口配置的检查器

由于自动检测不适用于相关协议，因此在绑定程序检查器中为以下检查器配置端口：

- `cip`
- `gtp_inspect`
- `iec104`
- `modbus`
- `s7commplus`

不需要端口配置的检查器

请勿在绑定程序检查器中为以下检查器配置端口，因为自动检测对相关协议有效：

- arp_spoof
- dce_smb
- dce_tcp
- dnp3
- ftp_client
- ftp_server
- http_inspect
- imap
- normalizer
- 弹出
- port_scan
- sip
- smtp
- ssh
- stream_icmp
- stream_ip
- stream_tcp
- stream_udp
- telnet

绑定程序检查器参数

绑定程序

绑定程序包括一组定义为 when 和 use 对象对的规则。

类型： 数组

示例：

```
{
    binder: {
        rules: [
            {
                "when": {
                    ...
                },
                "use": {
```

```
                    ...
                }
            },
            {
                "when": {
                    ...
                },
                "use": {
                    ...
                }
            }
        ]
    }
}
```

binder[].use.type

指定当 when 参数中的条件匹配时要绑定到数据流的检查器。例如，要检测CIP流量，请添加 use.type 值为 cip。

类型： 字符串

有效值： 本文档中描述的任何 Snort 3 检查器的名称。

默认值： 绑定程序 检查器为每个支持的检查器包含一个 use.type 参数。

binder[].when.proto

指定流量必须匹配的协议，以便将数据流绑定到 use.type中指定的检查器。例如，如果网络分析策略配置为检查 TCP 流量，则 绑定程序 检查器必须将此参数设置为 tcp。

类型： enum

有效值： any、 ip、 icmp、 tcp、 udp、 user、 file

默认值： 绑定程序 检查器为每个协议包含一个 when.proto 参数。

binder[].when.ports

指定流量必须匹配的端口，以便将数据流绑定到 use.type中指定的检查器。例如，要检测 TCP 端口 80 上的流量，请将 when.proto 设置为 tcp ，将 when.ports 设置为 80。

指定一个或多个以十进制或十六进制整数表示的端口列表。使用空格分隔多个端口并用双引号将列表括起来。

类型： 字符串

有效范围： -1 至 65535

默认值： 65535 （此值可能因 when.proto的值而异。）

binder[].when.role

指定流量必须匹配的角色，以便将流绑定到 use.type中指定的检查器。

类型： enum

有效值： client、 server、 any

默认值： `any`

指定流量必须匹配的服务，才能将流绑定到 `use.type`中指定的检查器。

类型： 字符串

有效值： 可能封装传入数据的服务的名称，例如： `netbios-ssn` 或 `dcerpc`。

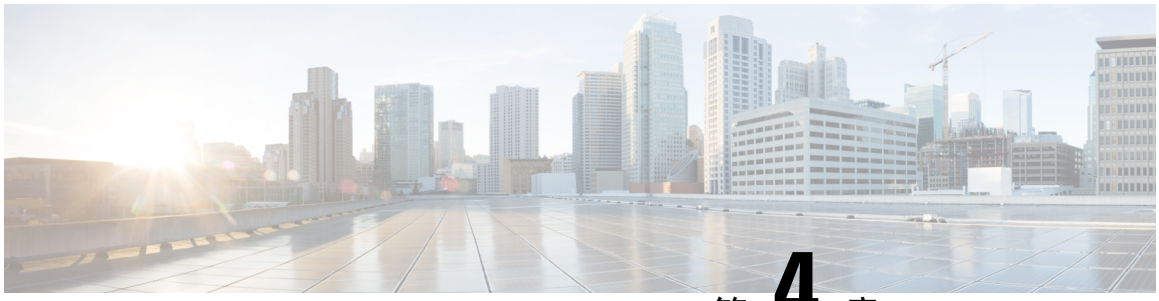
默认值： None

绑定程序检查器规则

绑定程序 检查器没有任何关联的规则。

绑定程序检查器入侵规则选项

绑定程序 检查器没有任何入侵规则选项。



第 4 章

CIP 检查器

CIP 检查器概述

类型	检查器（服务）
使用方式	检测
实例类型	多实例
所需其他检查器	`stream_tcp`
已启用	`false`

通用工业协议 (CIP) 是支持工业自动化应用的应用协议。EtherNet/IP (ENIP) 是基于以太网的网络中使用的 CIP 的实施。

`CIP` 检查器检测 TCP 或 UDP 上运行的 CIP 和 ENIP 流量，并将其发送给入侵规则引擎。可以使用自定义入侵规则中的 CIP 和 ENIP 关键字检测 CIP 和 ENIP 流量中的攻击。

注释 在 Snort 3 中， `cip` 检查器不支持 CIP 应用检测器。要实施 CIP 应用检测，您可以创建和导入自定义 CIP 入侵规则，并启用相应的 IPS 规则。有关详细信息，请参阅管理应用的 Snort 3 配置文档。

配置 CIP 检查器的最佳实践

在配置 cip 检查器时，请考虑以下最佳实践：

- 您必须在 绑定程序 检查器中添加默认 CIP 检测端口 44818 和任何其他 CIP 端口。
- 我们建议您使用入侵防御操作作为访问控制策略的默认操作。
- 要检测 CIP 和 ENIP 应用，您必须在相应的自定义网络分析策略中启用 cip 检查器。
- 要使用访问控制规则阻止 CIP 或 ENIP 应用流量，请确保在相应的网络分析策略中已启用规范器检查器和其内联模式选项（默认设置）。
- 要丢弃可触发 CIP 检查器规则和 CIP 入侵规则的流量，请确保在相应入侵策略中已启用 **内联时丢弃** 。
- cip 检查器不支持以下任一操作的访问控制策略默认操作：
  - **访问控制：信任所有流量**
  - **访问控制：阻止所有流量**
- CIP 检查器不支持 CIP 应用的应用可视性，包括网络发现。

CIP 检查器参数

CIP TCP 端口配置

绑定程序 检查器定义 CIP TCP 端口配置。有关详细信息，请参阅绑定程序检查器概述 ， 第 13 页。

示例：

```
[
    {
        "when": {
                    "role": "server",
                    "proto": "tcp",
                    "ports": "44818"
        },
        "use": {
                    "type": "cip"
        }
    }
]
```

embedded_cip_path

确定检查器是否检查嵌入式 CIP 连接路径。

类型： 字符串

有效值：

- "false"
- 用双引号引起来的 CIP 路径，例如 "0x2 0x36"。

默认值： "false"

unconnected_timeout

设置默认的无关联超时时间 (秒)。当 CIP 请求消息不包含协议特定超时值，并且达到每个 TCP 连接上并发无关联请求的最大数时，系统测定此参数指定的消息的秒数。如果计时器过期，则会删除此消息，以便腾出空间来存储未来的请求。

当指定 0 时，所有未配置特定于协议的超时时间的流量将首先超时。

类型： 整数

有效范围： 0 至 360

默认值： 300

max_unconnected_messages

设置每个 TCP 连接上并发无关联 CIP 消息的最大数。如果系统达到最大并发请求数，系统将关闭连接。

类型： 整数

有效范围： 1 至 10000

默认值： 100

max_cip_connections

设置系统允许的每个 TCP 连接上的同步 CIP 连接的最大数。

类型： 整数

有效范围： 1 至 10000

默认值： 100

CIP 检查器规则

对 生成事件并在内联部署中丢弃攻击性数据包启用 cip 检查器规则。

表 *2: CIP* 检查器规则

GID:SID	Rule Message
148:1	CIP 数据格式错误
148:2	CIP 数据不符合 ODVA 标准

GID:SID	Rule Message
148:3	超出 CIP 连接限制。已删除最近最少使用的连接
148:4	超出 CIP 未连接请求限制。已删除最早的请求

CIP 检查器入侵规则选项

cip_attribute

detection 参数指定其与 CIP 属性相匹配。

类型： 间隔

语法： `cip_attribute: <range_operator><positive integer>;` 或 `cip_attribute: <positive integer><range_operator><positive integer>;`

有效值： 介于 0 到 65535 之间的一个或多个整数的集合，以及 表 3: 范围格式 中指定的 `range_operator`。

示例： `cip_attribute: 100;`

cip_class

检测参数设置。

类型： 间隔

语法： `cip_class:` <range_operator><positive integer>中所述：或 `cip_class:` 中所述：

有效值： 介于 0 到 65535 之间的一个或多个整数的集合，以及 表 3: 范围格式 中指定的 `range_operator`。

示例： `cip_class: 25;`

cip_conn_path_class

检测参数，以匹配 CIP 连接路径类。

类型： 间隔

语法： `cip_conn_path_class: <range_operator><positive integer>;` 或 `cip_conn_path_class: <positive integer><range_operator><positive integer>;`

有效值： 介于 0 到 65535 之间的一个或多个整数的集合，以及 表 3: 范围格式 中指定的 `range_operator`。

示例： `cip_conn_path_class: 85;`

cip_instance

Detection 参数以与 CIP 实例匹配。

类型： 间隔

语法： `cip_instance:` <range_operator><positive integer>中所述：或 `cip_instance:`中所述：

有效值： 介于 0 到 65535 之间的一个或多个整数的集合，以及 表 3: 范围格式 中指定的 `range_operator`。

示例： `cip_instance: 15;`

cip_req

Detection 参数以与 CIP 请求匹配。

语法： `cip_req;`

示例： `cip_req;`

cip_rsp

Detection 参数以与 CIP 响应匹配。

语法： `cip_rsp;`

示例： `cip_rsp;`

cip_service

Detection 参数以与 CIP 服务匹配。

类型： 间隔

语法： `cip_service: <range_operator><positive integer>;` 或 `cip_service: <positive integer><range_operator><positive integer>;`

有效值： 一组介于 0 到 127 之间的一个或多个整数，以及 表 3: 范围格式 中指定的 `range_operator`。

示例： `cip_service: 50;`

cip_status

检测参数以匹配 CIP 响应状态。

类型： 间隔

语法： `cip_status: <range_operator><positive integer>;` 或 `cip_status: <positive integer><range_operator><positive integer>;`

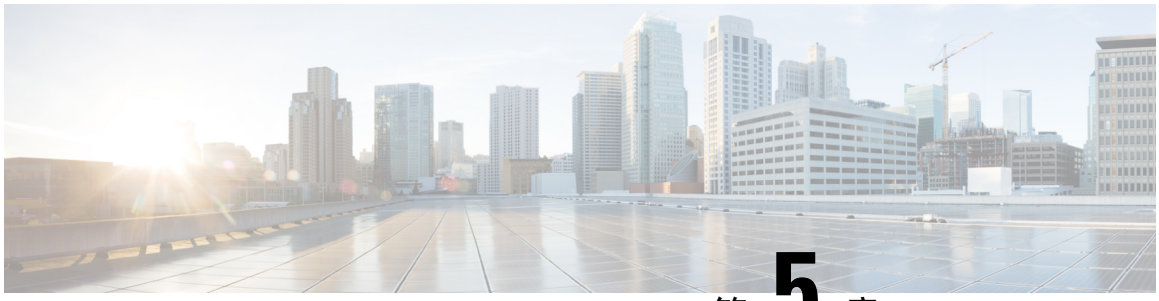
有效值： 一组介于 0 到 255 之间的一个或多个整数，以及 表 3: 范围格式 中指定的 `range_operator`。

示例： `cip_status: 250;`

表 3: 范围格式

范围格式	Operator	说明
operator i		

范围格式	Operator	说明
	<	少于
	>	大于
	=	平分
	≠	不等于
	⩽	小于或等于
	⩾	大于或等于
j operator k		
	<>	大于 j 且小于 k
	<=>	大于或等于 j 且小于或等于 k



第 5 章

DCE SMB 检查器

DCE SMB 检查器概述

类型	检查器（服务）
使用方式	检测
实例类型	多实例
所需其他检查器	无
已启用	`true`

DCE/RPC 协议使不同网络主机上的进程可以像在同一主机上一样进行通信。这些进程间通信一般通过 TCP 和 UDP 在主机之间传输。在 TCP 传输中，DCE/RPC 也可以进一步封装在 Windows 服务器消息块 (SMB) 协议或 Samba 中；Samba 是一种在由 Windows 和类似 UNIX 或 Linux 操作系统组成的混合环境中用于进程间通信的开源 SMB 实现。

虽然大多数 DCE/RPC 漏洞出现在针对 DCE/RPC 服务器（实际上可能是网络上运行 Windows 或 Samba 的任何主机）的 DCE/RPC 客户端请求中，但在服务器响应中也可能出现漏洞。

IP 封装所有 DCE/RPC 传输。TCP 传输所有面向连接 DCE/RPC，如 SMB。

`dce_smb` 检查器检测 SMB 协议中面向连接的 DCE/RPC，并使用协议特定特征（包括信头长度和数据分段顺序）来：

- 检测封装在 SMB 传输中的 DCE/RPC 请求和响应。
- 分析 DCE/RPC 数据流并检测 DCE/RPC 流量中的异常行为和逃避技术。
- 分析 SMB 数据流并检测异常 SMB 行为和逃避技术。

- 对 SMB 进行分片，对 DCE/RPC 进行分片重组。
- 规范化 DCE/RPC 流量，以便规则引擎进行处理。

下图说明了 DCE SMB 检查器开始为不同传输处理 SMB 流量的点。

dce_smb 检查器通常接收适用于 NetBIOS 会话服务的已知 TCP 端口 139 或以类似方式实现的已知 Windows 端口 445 上的 SMB 流量。由于 SMB 具有除传输 DCE/RPC 以外的许多功能，因此，检查器会首先测试 SMB 流量是否携带 DCE/RPC 流量，如果不是则停止处理，如果是则继续处理。

对 dce_smb 检查器参数和功能的描述包括 DCE/RPC 的 Microsoft 实施，包括 Microsoft 远程过程调用 (MSRPC)，以及 SMB 和 Samba。

基于目标的策略

Windows 和 Samba DCE/RPC 的实现有很大不同。例如，在对 DCE/RPC 流量进行分片重组时，所有 Windows 版本都在第一个分片中使用 DCE/RPC 上下文 ID，而所有 Samba 版本都在最后一个分片中使用上下文 ID。再如，Windows Vista 在第一个分片中使用操作编号报头字段来识别特定函数调用，而 Samba 及其他所有 Windows 版本都在最后一个分片中使用操作编号字段。

Windows 和 Samba SMB 的实现有很大不同。例如，Windows 在与命名管道配合使用时可识别 SMB OPEN 和 READ 命令，而 Samba 不能识别这些命令。

因此，dce_smb 检查器使用基于目标的方法。配置 dce_smb 检查器实例时，policy 参数指定 DCE/RPC SMB 协议的实施。将此信息与主机信息相结合，建立默认的基于目标的服务器策略或者，您可以配置以其他主机和 DCE/RPC SMB 实施为目标的其他检查器。默认的基于目标的服务器策略指定的 DCE/RPC SMB 实施适用于另一个 dce_smb 检查器实例没有作为目标的任何主机。

dce_smb 检查器可以使用 policy 参数定位的 DCE/RPC SMB 实施包括：

- WinXP（默认）
- Win2000
- WinVista
- Win2003
- Win2008
- Win7
- Samba
- Samba-3.0.37
- Samba-3.0.22

- Samba-3.0.20

文件检查

dce_smb 检查器支持 SMB 版本 1、2 和 3 的文件检查。

dce_smb 检查器检查正常的 SMB 文件传输。这包括通过文件处理检查文件类型和签名，以及为 file_data 规则选项设置指针。 dce_smb 检查器支持与 file_id 检查器配合使用时检查 SMB 版本 1、2 和 3 的正常 SMB 文件传输（详见 Snort 3 开源文档，可从 https://www.snort.org/snort3获取）。要启用文件检测，请根据需要配置 file_id 检查器，并设置 dce_smb 、smb_file_inspection 和 smb_file_length 参数。smb_file_length 参数表示 file_id 检查器检查的文件数据字节数（从 file_data IPS 规则选项指示的指针开始）。有关详细信息，请参阅 Snort 3 开源文档，网址为 https://www.snort.org/snort3）。

解除分区

dce_smb 检查器支持重组分片的数据包。此功能在内联模式下非常有用，可以在完整的分片重组完成之前于检测流程的早期阶段捕获漏洞攻击，或捕获利用分段隐藏自身的漏洞攻击。请注意，禁用分片重组可能会导致大量的漏报。

DCE SMB 检查器参数

DCE SMB 端口配置

绑定程序 检查器定义 DCE SMB 端口配置。有关详细信息，请参阅绑定程序检查器概述 ， 第 13 页。

示例：

```
[
   {
       "when": {
           "role":"any",
           "service":"netbios-ssn",
           "ports": ""
       },
        "use": {
            "type":"dce_smb"
       }
   }
]
```

max_frag_len

指定分片重组允许的最大分片长度（以字节为单位）。为了处理更大的分段，检查器会在进行分段重组之前考虑此大小的数据包内容。

注释 在此参数中指定的值必须大于或等于规则检查数据所需的深度以确保 检测到。要确保所有数据都通过检测，请使用默认值。

类型： 整数

有效范围： 1514 至 65535

默认值： 65535

smb_max_compound

指定一个 SMB 请求中要处理的最大命令数。

类型： 整数

有效范围： 0 至 255

默认值： 3

smb_max_chain

指定允许的链式 SMB AndX 命令的最大数量。通常，超过若干链式 AndX 命令即表示存在异常行为，可能代表有躲避行为。指定 1 表示不允许链式命令，指定 0 将会禁止检测链式命令数量。

dce_smb 检查器会首先计算链式命令数量，如果随附的 SMB 检查器规则已启用，并且链式命令数量等于或超过配置的值，预处理器将会生成事件。然后会继续进行处理。

您可以启用规则 133:20 来生成此参数的事件，并且在内联部署中丢弃违规数据包。

类型： 整数

有效范围： 0 至 255

默认值： 3

disable_defrag

指定是否对 DCE/RPC 流量进行分片重组。当启用时，dce_smb 检查器仍会检测异常并向规则引擎发送 DCE/RPC 数据，但可能会检测不出分片 DCE/RPC 数据中的漏洞。

尽管 disable_defrag 提供了不对流量进行碎片整理的灵活性，并且可以加快处理速度，但大多数 DCE/RPC 漏洞都试图利用碎片来隐藏漏洞。启用此参数将会忽略大多数已知漏洞，从而造成大量误报。

类型： boolean

有效值： true、 false

默认值： false

limit_alerts

指定是否将 DCE 警报限制为每个签名每个流最多一个。

类型： boolean

有效值： true、 false

默认值： true

reassemble_threshold

指定将重组的数据包发送到规则引擎之前，DCE/RPC 取消分片和分片重组缓冲区中待排队的最小字节数。此参数在内联模式下非常有用，因为它可以在内联模式下检测到潜在的漏洞攻击，可以在完整的分片重组完成之前提前捕获漏洞。

请注意，值越小，实现早期检测的可能性越高，但可能会对性能造成负面影响。如果启用此参数，应当测试性能所受的影响。

值 `0` 将禁用重组。

类型： 整数

有效范围： `0` 至 `65535`

默认值： `0`

策略

指定目标主机或受监控网段上主机使用的 Windows 或 Samba DCE/RPC 应用。

类型： enum

有效值： 从以下列表中选择的字符串：`Win2000`、`WinXP`、`WinVista`、`Win2003`、`Win2008`、`Win7`、`Smb`、`Smb-3.0.37`、`Sack-3.0.22`、`Smb-3.0.20`

默认值： `WinXP`

smb_max_credit

指定最大待处理请求数。

类型： 整数

有效范围： `1` 至 `65536`

默认值： `8192`

smb_file_depth

指定在 SMB 流量中检测到文件时检查的字节数，从 `file_data` IPS 规则选项指定的位置开始（如 Snort 3 开源文档所述，可从 https://www.snort.org/snort3获取）。

指定 `-1` 以禁用文件检查。

指定 `0` 以表示不受限制的文件检查。

类型： 整数

有效范围： `-1` 至 `32767`

默认值： `16384`

在 SMB 流量中检测到文件时，`smb_file_length` 参数表示检查器从 `file_data` IPS 规则选项中设置的指针开始检查的文件数据字节数。为了检查文件类型和签名，`dce_smb` 使用 `file_id`检查器中设置的 `enable_type`、`type_length`、`enable_signature` 和 `signature_length` 参数。有关 `file_id` 检查器的详细信息，请参阅 https://www.snort.org/snort3提供的 Snort 开源文档。

memcap

指定分配给检查器的最大内存限制（以字节为单位）。当达到或超过最大内存上限时， dce_smb 检查器会删除最近最少使用的数据以腾出更多空间。

类型： 整数

有效范围： 512 到 9,007,199,254,740,992 (maxSZ)

默认值： 8,388,608

smb_fingerprint_policy

因为检查器检测在 SMB Session Setup AndX 请求和响应中识别出的 Windows 或 Samba 版本。如果检测到的版本不同于为 策略 检查器参数配置的 Windows 或 Samba 版本，检测到的版本将会仅覆盖为该会话配置的版本。

例如，如果将策略 (Policy) 设置为 Windows XP，而检查器检测到 Windows Vista，则检查器将对该会话使用 Windows Vista 策略。其他设置仍然有效。

类型： enum

有效值： none, client, server 或 both

- 使用 客户端，检查该策略类型的服务器到客户端流量。
- 使用 服务器，检查该策略类型的客户端到服务器流量。
- 使用 两者，检查该策略类型的服务器到客户端流量和客户端到服务器流量。
- 使用 none 以禁用 Windows 或 Smba 版本检测。

默认值： none

smb_legacy_mode

如果 smb_legacy_mode (Legacy SMB Inspection Mode) 为真，则系统只会将 SMB 入侵规则应用于 SMB 版本 1 流量，并将 DCE/RPC 入侵规则应用于使用 SMB 版本 1 作为传输的 DCE/RPC 流量。

当 smb_legacy_mode 为 false时，系统将 SMB 入侵规则应用于使用 SMB 版本 1 和 2 的流量，并且：

- 对于版本 7.0 和 7.0.x，系统仅将 SMB 用于 SMB 第 1 版，将 DCE/RPC 入侵规则应用于 DCE/RPC 流量。
- 对于版本 7.1+，系统使用 SMB 作为 SMB 版本 1 和 2 的传输，将 DCE/RPC 入侵规则应用于 DCE/RPC 流量。

类型： boolean

有效值： true、 false

默认值： false

valid_smb_versions

指定要检测的 SMB 版本。使用空格字符分隔多个 SMB 版本。

类型： 字符串

有效值： `v1`, `v2`, `v3`, `all`

默认值： `all`

DCE SMB 检查器规则

启用 `dce_smb` 检查器规则，以 生成事件并在内联部署中丢弃攻击性数据包。

表 *4: DCE SMB* 检查器规则

GID:SID	Rule Message
133:2	SMB - NetBIOS 会话服务会话类型错误
133:3	SMB - 错误的 SMB 消息类型
133:4	SMB - 错误的 SMB ID (SMB1 不是 \xffSMB 或 SMB2 不是 \xfeSMB)
133:5	SMB - 错误字数计数或结构大小
133:6	SMB - 错误字节计数
133:7	SMB - 格式错误类型
133:8	SMB - 错误的偏移量
133:9	SMB - 零数据总数
133:10	SMB - NetBIOS 数据长度小于 SMB 信头长度
133:11	SMB - 剩余 NetBIOS 数据长度小于命令长度
133:12	SMB - 剩余 NetBIOS 数据长度小于命令字节数
133:13	SMB - 剩余 NetBIOS 数据长度小于命令数据大小
133:14	SMB - 剩余总数据计数小于此命令的数据大小
133:15	SMB - 发送的总数据 (STDu64) 大于命令预期的总数据
133:16	SMB - 字节数小于命令数据大小 (STDu64)
133:17	SMB - 命令数据大小对于字节数无效
133:18	SMB - 具有待处理树连接响应的树连接请求过多
133:19	SMB - 具有待处理读取响应的读取请求过多
133:20	SMB-命令链接过多

GID:SID	Rule Message
133:21	SMB - 多个链式登录请求
133:22	SMB - 多个链式树连接请求
133:23	SMB - 链式/复合登录和注销
133:24	SMB - 链式/复合树连接后树断开
133:25	SMB - 链式/复合开管道后接闭管道
133:26	SMB - 无效共享访问
133:44	SMB - 发现无效的 SMB 版本 1
133:45	SMB - 发现无效的 SMB 版本 2
133:46	SMB - 无效用户、树连接、文件绑定
133:47	SMB - 命令复合过多
133:48	SMB - 零数据计数
133:50	SMB — 超出最大未完成请求数
133:51	SMB — 具有相同 MID 的待处理请求
133:52	SMB - 已弃用协商的方言
133:53	SMB - 使用已弃用的命令
133:54	SMB - 使用异常命令
133:55	SMB - 命令的设置计数无效
133:56	SMB - 客户端尝试在会话中进行多个方言协商
133:57	SMB - 客户端尝试创建文件或将文件的属性设置为只读/隐藏/系统
133:58	SMB - 提供的文件偏移量大于指定的文件大小
133:59	SMB - SMB2 报头中指定的下一个命令超出负载边界

DCE 检查器入侵规则选项

dce_iface

指定以下以逗号分隔的元素：

- 服务接口的 UUID。
- 接口版本（可选）。默认设置会与任何版本匹配。
- 规则是否应匹配请求中的任何分段的指示符（可选）。默认设置仅匹配第一个分段。

在 DCE/RPC 协议中，客户端调用某个服务之前必须先绑定该服务。当客户端向 服务器发送绑定请求时，它可以指定要绑定的一个或多个服务接口。每个接口均由一个 UUID 表示，并且每个接口 UUID 都与一个唯一索引（或情景 ID）配对，将来的请求可用于引用客户端调用的服务。服务器使用其接受为有效的接口UUID进行响应，并允许客户端向这些服务发出请求。当客户端发出请求时，它将指定情景 ID，以便服务器知道客户端向哪个服务发出请求。

使用 `dce_iface` 规则选项，规则可以询问检查器客户端是否已绑定到特定接口UUID，以及此客户端请求是否向该接口发出请求。这可以消除多个服务成功绑定到的误报，因为检查器可以将绑定UUID与请求中使用的情景 id 相关联。

`dce_iface` 选项需要在检查器中跟踪面向连接的 DCE/RPC 的客户端绑定和修改情景请求以及服务器绑定确认和修改情景响应。对于每个“绑定和修改情景”请求，客户端指定一个接口 UUID 列表以及将在 DCE/RPC 会话期间用于引用接口的每个接口 UUID 的句柄（或情景 ID）。服务器响应指示它允许客户端向哪些接口发出请求—它要么接受或拒绝客户端绑定到特定接口的希望。此跟踪是必需的，以便在处理请求时，可以将请求中使用的情景 ID 与其作为句柄的接口 UUID 相关联。

在以下条件下， `dce_iface` 规则选项匹配：

- 指定的接口 UUID 与 DCE/RPC 请求的接口 UUID（通过情景 ID 引用）相匹配

且

- 未提供 `版本` 参数，或提供了 `版本` 参数且该参数与 DCE/RPC 请求的接口 UUID 匹配

且

- 提供了 `any_frag` 参数，或者不存在 `any_frag` 参数并且 `dce_iface` 选项与初始请求分片中的UUID和版本条件匹配

示例：

```
dce_iface:4b324fc8-1670-01d3-1278-5a47bf6ee188;
dce_iface:4b324fc8-1670-01d3-1278-5a47bf6ee188, <2;
dce_iface:4b324fc8-1670-01d3-1278-5a47bf6ee188,any_frag;
dce_iface:4b324fc8-1670-01d3-1278-5a47bf6ee188, =1, any_frag;
```

dce_iface.uuid

DCE/RPC 请求可以指定 UUID 编号是以大端字节还是小端字节表示。接口 UUID 在请求中的表示形式根据请求中指定的字节顺序而不同。 `dce_rpc` 检查器会对 UUID 进行规范化。这意味着 `dce_iface` 规则选项中的 UUID 规范必须以相同的方式写入，而无论请求的字节顺序如何。

例如，小端字节绑定请求将表示如下 UUID：

```
|f8 91 7b 5a 00 ff d0 11 a9 b2 00 c0 4f b6 e6 fc|
```

高字节在前绑定请求将表示相同的 UUID，如下所示：

```
|5a 7b 91 f8 ff 00 11 d0 a9 b2 00 c0 4f b6 e6 fc|
```

在使用 dce_iface 选项的 Snort 3 规则中，无论请求的字节序如何，UUID 必须使用大端字节顺序在字符串中表示：

```
5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc
```

类型： 字符串

语法： `dce_iface: <UUID>;`

有效值： 其中：*UUID* 是 32 个十六进制数字，以连字符分隔五组，显示格式为：`xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`

示例： `dce_iface:5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc;`

dce_iface.version

服务接口具有与其关联的版本。某些接口版本可能不容易受到某些漏洞攻击。为此，您可以在 dce_iface 选项中指定一个或多个版本号，以确定是否需要检查特定攻击程序。

类型： 间隔

语法： `dce_iface: <range_operator><positive integer>;` or `dce_iface: <positive integer><range_operator><positive integer>;`

有效值： 表 5: 范围格式 中指定的一个或多个正版本号和 range_operator的集合。

示例： `dce_iface:=6;`

dce_iface.any_frag

可以将 DCE/RPC 请求分为一个或多个分段。在 DCE/RPC 信头中设置标志以指示当前分段是请求的第一个分段、中间分段还是最后一个分段。仅当 DCE/RPC 请求是第一分段（或完整请求）时，对 DCE/RPC 请求中数据的许多检查才具有相关性。因此，第一个分段之后的分段包含 DCE/RPC 请求更深层次的数据。例如，在请求的前五个字节（例如，长度字段）中查找数据的规则会在除第一个字节之外的分段上找到错误的数据。后续分段的开头距离请求的开头偏移一段长度。这可能是分段 DCE/RPC 流量中误报的一个来源。

因此，默认情况下，DCE_RPC 检查器仅匹配请求中的初始分片。要强制检查器检查匹配请求中的所有分片，请将 any_frag 添加到 dce_iface 规则选项。请注意，已进行分片重组的 DCE/RPC 请求被视为完整请求。

语法： `dec_iface: any_frag;`

示例： `dce_iface: any_frag;`

dce_opnum

匹配DCERPC操作编号、操作编号范围或操作编号列表。此选项表示可对接口进行的一个或多个特定函数调用。客户端绑定到特定服务接口并向其发出请求后，规则需要确定客户端对服务进行的函数调用，以检查是否存在可能存在于函数调用中的漏洞。函数调用被指定为用双引号括起来的操作编号 (opnums) 列表

类型： 字符串

语法： `dce_opnum: <opnum_list>;`

其中，*opnum_list* 是下列各项之一：

- 单个整数。
- 以逗号分隔的整数列表。
- 指定的整数范围，用连字符分隔范围内的最小和最大数字。
- 以上各项的任意。

有效值： DCE/RPC 请求 opnum 的列表。

示例：

```
dce_opnum: "15";
dce_opnum: "15-18";
dce_opnum: "15, 18-20";
dce_opnum: "15, 17, 20-22";
```

dce_stub_data

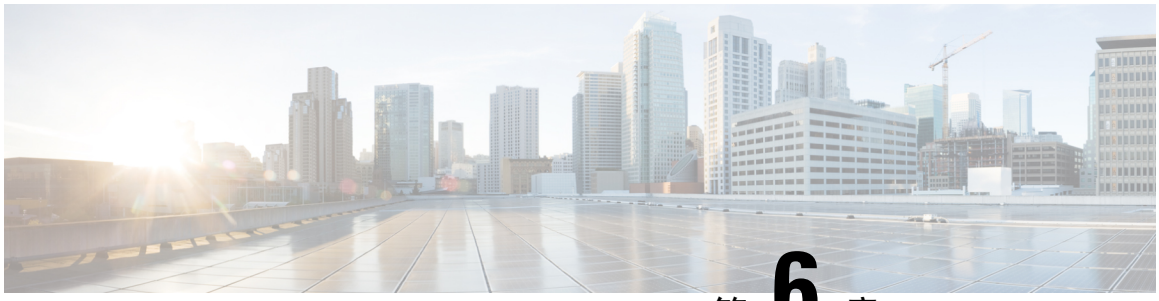
此选项将检测光标（用于在规则处理中遍历数据包负载）放在DCE/RPC末节数据的开头，不考虑前面的规则选项。如果存在 DCE/RPC 末节数据，则此选项匹配。

语法： `dce_stub_data;`

示例： `dce_stub_data;`

表 *5:* 范围格式

范围格式	Operator	说明
operator i		
	<	少于
	>	大于
	=	平分
	≠	不等于
	≤	小于或等于
	≥	大于或等于
j operator k		
	<>	大于 j 且小于 k
	<=>	大于或等于 j 且小于或等于 k



第 6 章

DCE TCP 检查器

DCE TCP 检查器概述

类型	检查器（服务）
使用方式	检测
实例类型	多实例
所需其他检查器	无
已启用	`true`

DCE/RPC 协议使不同网络主机上的进程可以像在同一主机上一样进行通信。这些进程间通信一般通过 TCP 在主机之间传输。在 TCP 传输中，DCE/RPC 也可以进一步封装在 Windows 服务器消息块 (SMB) 协议或 Samba 中；Samba 是一种在由 Windows 和类似 UNIX 或 Linux 操作系统组成的混合环境中用于进程间通信的开源 SMB 实现。

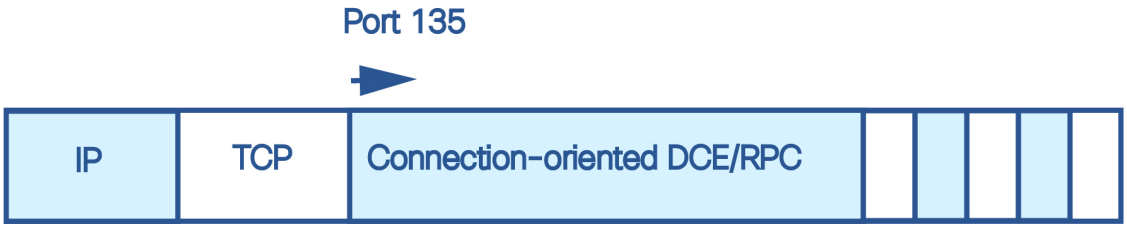
虽然大多数 DCE/RPC 漏洞出现在针对 DCE/RPC 服务器（实际上可能是网络上运行 Windows 或 Samba 的任何主机）的 DCE/RPC 客户端请求中，但在服务器响应中也可能出现漏洞。

IP 封装所有 DCE/RPC 传输。TCP 传输所有面向连接 DCE/RPC。DCE TCP 检查器检测面向连接的 DCE/RPC，并使用协议特定的特征（例如，报头长度和数据分片顺序）来：

- 检测封装在 TCP 传输（包括使用版本 1 RPC over HTTP 的 TCP 传输 DCE/RPC）中的 DCE/RPC 请求和响应。
- 分析 DCE/RPC 数据流并检测 DCE/RPC 流量中的异常行为和逃避技术。
- 对 DCE/RPC 进行分片重组。

- 规范化 DCE/RPC 流量，以便规则引擎进行处理。

下图说明了 DCE TCP 检查器开始为不同传输处理 TCP 流量的点。



已知 TCP 端口 135 识别 TCP 传输中的 DCE/RPC 流量。图中未包含 RPC over HTTP。对于 RPC over HTTP，面向连接 DCE/RPC 在完成 HTTP 初始设置序列后直接通过 TCP 传输（如图所示）。

基于目标的策略

Windows 和 Samba DCE/RPC 的实现有很大不同。例如，在对 DCE/RPC 流量进行分片重组时，所有 Windows 版本都在第一个分片中使用 DCE/RPC 上下文 ID，而所有 Samba 版本都在最后一个分片中使用上下文 ID。再如，Windows Vista 在第一个分片中使用操作编号报头字段来识别特定函数调用，而 Samba 及其他所有 Windows 版本都在最后一个分片中使用操作编号字段。

因此， `dce_tcp` 检查器使用基于目标的方法。在配置 `dce_tcp` 检查器实例时， `policy` 参数指定 DCE/RPC TCP 协议的特定实施。将此信息与主机信息相结合，建立默认的基于目标的服务器策略或者，您可以配置以其他主机和 DCE/RPC TCP 实施为目标的其他检查器。由默认的基于目标的服务器策略指定的 DCE/RPC TCP 实施适用于其他 `dce_tcp` 检查器实例不作为目标的任何主机。

DCE TCP 检查器可以使用 `policy` 参数确定的 DCE/RPC 实现包括：

- WinXP（默认）
- Win2000
- WinVista
- Win2003
- Win2008
- Win7
- Samba
- Samba-3.0.37
- Samba-3.0.22
- Samba-3.0.20

解除分区

DCE TCP 检查器支持在将分片的数据包发送到检测引擎之前对其进行重组。此功能在内联模式下非常有用，可以在完整的分片重组完成之前于检测流程的早期阶段捕获漏洞攻击，或捕获利用分段隐藏自身的漏洞攻击。请注意，禁用分片重组可能会导致大量的漏报。

DCE TCP 检查器参数

DCE TCP 端口配置

绑定程序 检查器定义 DCE TCP 端口配置。有关详细信息，请参阅绑定程序检查器概述，第 13 页。

示例：

```
[
   {
        "when": {
            "role": "any",
            "proto": "tcp",
            "service": "dcerpc",
            "ports": ""
        },
        "use": {
            "type": "dce_tcp"
        }
    }
]
```

max_frag_len

指定分片重组允许的最大分片长度（以字节为单位）。为了处理较大的分片，检查器会在进行分片重组之前将数据包内容视为指定的大小。

注释 在此参数中指定的值必须大于或等于规则检查数据所需的深度以确保 检测到。要确保所有数据都通过检测，请使用默认值。

类型： 整数

有效范围： 1514 至 65535

默认值： 65535

disable_defrag

指定是否对 DCE/RPC 流量进行分片重组。当此参数为 true时，检查器仍会检测异常并向规则引擎发送 DCE/RPC 数据，但可能会检测不出分片 DCE/RPC 数据中的漏洞。

尽管此参数提供了不对流量进行碎片整理的灵活性，并且可以加快处理速度，但大多数DCE/RPC漏洞都试图利用碎片来隐藏漏洞。启用此参数将会忽略大多数已知漏洞，从而造成大量误报。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

limit_alerts

指定是否将 DCE 警报限制为每个签名每个流最多一个。

类型： boolean

有效值： `true`、 `false`

默认值： `true`

reassemble_threshold

指定将重组的数据包发送到规则引擎之前，DCE/RPC 取消分片和分片重组缓冲区中待排队的最小字节数。此参数在内联模式下非常有用，因为它可以在内联模式下检测到潜在的漏洞攻击，可以在完整的分片重组完成之前提前捕获漏洞。

值越小，实现早期检测的可能性越高，但可能会对性能造成负面影响。如果启用此参数，应当测试性能所受的影响。

值 `0` 将禁用重组。

类型： 整数

有效范围： `0` 至 `65535`

默认值： `0`

策略

指定目标主机或受监控网段上主机使用的 Windows 或 Samba DCE/RPC 应用。

类型： enum

有效值： 从以下列表中选择的字符串： `Win2000`、 `WinXP`、 `WinVista`、 `Win2003`、 `Win2008`、 `Win7`、 `Smb`、 `Smb-3.0.37`、 `Sack-3.0.22`、 `Smb-3.0.20`

默认值： `WinXP`

DCE TCP 检查器规则

启用 `dce_tcp` 检查器规则，以 生成事件并在内联部署中丢弃攻击性数据包。

表 *6: DCE TCP* 检查器规则

GID:SID	Rule Message
133:27	面向连接的 DCE/RPC - 主版本无效
133:28	面向连接的 DCE/RPC - 无效的次版本

GID:SID	Rule Message
133:29	面向连接的 DCE/RPC - 无效的次版本
133:30	面向连接的 DCE/RPC - 分片长度小于信头大小
133:31	面向连接的 DCE/RPC - 剩余分片长度小于所需大小
133:32	面向连接的 DCE/RPC - 未指定情景项目
133:33	面向连接的 DCE/RPC - 不指定传输语法
133:34	面向连接的 DCE/RPC - 非最后分片上的分片长度小于客户端的最大协商分片传输大小
133:35	面向连接的 DCE/RPC - 分片长度大于协商的最大分片传输大小
133:36	面向连接的 DCE/RPC - 更改与绑定不同的上下文字节顺序
133:37	面向连接的 DCE/RPC - 非第一个/最后一个分段的呼叫 ID 与为分段请求建立的呼叫 ID 不同
133:38	面向连接的 DCE/RPC - 非第一个/最后一个分片的 opnum 与为分片请求建立的 opnum 不同
133:39	面向连接的 DCE/RPC - 非第一个/最后一个分段的情景 ID 与为分段请求建立的情景 ID 不同

DCE 检查器入侵规则选项

dce_iface

指定以下以逗号分隔的元素：

- 服务接口的 UUID。
- 接口版本（可选）。默认设置会与任何版本匹配。
- 规则是否应匹配请求中的任何分段的指示符（可选）。默认设置仅匹配第一个分段。

在 DCE/RPC 协议中，客户端调用某个服务之前必须先绑定该服务。当客户端向 服务器发送绑定请求时，它可以指定要绑定的一个或多个服务接口。每个接口均由一个 UUID 表示，并且每个接口 UUID 都与一个唯一索引（或情景 ID）配对，将来的请求可用于引用客户端调用的服务。服务器使用其接受为有效的接口UUID进行响应，并允许客户端向这些服务发出请求。当客户端发出请求时，它将指定情景 ID，以便服务器知道客户端向哪个服务发出请求。

使用`dce_iface`规则选项，规则可以询问检查器客户端是否已绑定到特定接口UUID，以及此客户端请求是否向该接口发出请求。这可以消除多个服务成功绑定到的误报，因为检查器可以将绑定UUID与请求中使用的情景 id 相关联。

dce_iface 选项需要在检查器中跟踪面向连接的 DCE/RPC 的客户端绑定和修改情景请求以及服务器绑定确认和修改情景响应。对于每个“绑定和修改情景”请求，客户端指定一个接口 UUID 列表以及将在 DCE/RPC 会话期间用于引用接口的每个接口 UUID 的句柄（或情景 ID）。服务器响应指示它允许客户端向哪些接口发出请求—它要么接受或拒绝客户端绑定到特定接口的希望。此跟踪是必需的，以便在处理请求时，可以将请求中使用的情景 ID 与其作为句柄的接口 UUID 相关联。

在以下条件下， dce_iface 规则选项匹配：

- 指定的接口 UUID 与 DCE/RPC 请求的接口 UUID（通过情景 ID 引用）相匹配

且

- 未提供 版本 参数，或提供了 版本 参数且该参数与 DCE/RPC 请求的接口 UUID 匹配

且

- 提供了 any_frag 参数，或者不存在 any_frag 参数并且 dce_iface 选项与初始请求分片中的 UUID 和版本条件匹配

示例：

```
dce_iface:4b324fc8-1670-01d3-1278-5a47bf6ee188;
dce_iface:4b324fc8-1670-01d3-1278-5a47bf6ee188, <2;
dce_iface:4b324fc8-1670-01d3-1278-5a47bf6ee188,any_frag;
dce_iface:4b324fc8-1670-01d3-1278-5a47bf6ee188, =1, any_frag;
```

dce_iface.uuid

DCE/RPC 请求可以指定 UUID 编号是以大端字节还是小端字节表示。接口 UUID 在请求中的表示形式根据请求中指定的字节顺序而不同。 dce_rpc 检查器会对 UUID 进行规范化。这意味着 dce_iface 规则选项中的 UUID 规范必须以相同的方式写入，而无论请求的字节顺序如何。

例如，小端字节绑定请求将表示如下 UUID：

```
|f8 91 7b 5a 00 ff d0 11 a9 b2 00 c0 4f b6 e6 fc|
```

高字节在前绑定请求将表示相同的 UUID，如下所示：

```
|5a 7b 91 f8 ff 00 11 d0 a9 b2 00 c0 4f b6 e6 fc|
```

在使用 dce_iface 选项的 Snort 3 规则中，无论请求的字节序如何，UUID 必须使用大端字节顺序在字符串中表示：

```
5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc
```

类型： 字符串

语法： dce_iface: <UUID>;

有效值： 其中： *UUID* 是 32 个十六进制数字，以连字符分隔五组，显示格式为：
xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx

示例： dce_iface:5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc;

dce_iface.version

服务接口具有与其关联的版本。某些接口版本可能不容易受到某些漏洞攻击。为此，您可以在 `dce_iface` 选项中指定一个或多个版本号，以确定是否需要检查特定攻击程序。

类型： 间隔

语法： `dce_iface: <range_operator><positive integer>;` or `dce_iface: <positive integer><range_operator><positive integer>;`

有效值： 表 7: 范围格式 中指定的一个或多个正版本号和 `range_operator`的集合。

示例： `dce_iface:=6;`

dce_iface.any_frag

可以将 DCE/RPC 请求分为一个或多个分段。在 DCE/RPC 信头中设置标志以指示当前分段是请求的第一个分段、中间分段还是最后一个分段。仅当 DCE/RPC 请求是第一分段（或完整请求）时，对 DCE/RPC 请求中数据的许多检查才具有相关性。因此，第一个分段之后的分段包含 DCE/RPC 请求更深层次的数据。例如，在请求的前五个字节（例如，长度字段）中查找数据的规则会在除第一个字节之外的分段上找到错误的数据。后续分段的开头距离请求的开头偏移一段长度。这可能是分段 DCE/RPC 流量中误报的一个来源。

因此，默认情况下，DCE_RPC 检查器仅匹配请求中的初始分片。要强制检查器检查匹配请求中的所有分片，请将 `any_frag` 添加到 `dce_iface` 规则选项。请注意，已进行分片重组的 DCE/RPC 请求被视为完整请求。

语法： `dec_iface: any_frag;`

示例： `dce_iface: any_frag;`

dce_opnum

匹配DCE RPC操作编号、操作编号范围或操作编号列表。此选项表示可对接口进行的一个或多个特定函数调用。客户端绑定到特定服务接口并向其发出请求后，规则需要确定客户端对服务进行的函数调用，以检查是否存在可能存在于函数调用中的漏洞。函数调用被指定为用双引号括起来的操作编号 (opnums) 列表

类型： 字符串

语法： `dce_opnum: <opnum_list>;`

其中， *`opnum_list`* 是下列各项之一：

- 单个整数。
- 以逗号分隔的整数列表。
- 指定的整数范围，用连字符分隔范围内的最小和最大数字。
- 以上各项的任意。

有效值： DCE/RPC 请求 opnum 的列表。

示例：

```
dce_opnum: "15";
dce_opnum: "15-18";
dce_opnum: "15, 18-20";
dce_opnum: "15, 17, 20-22";
```

dce_stub_data

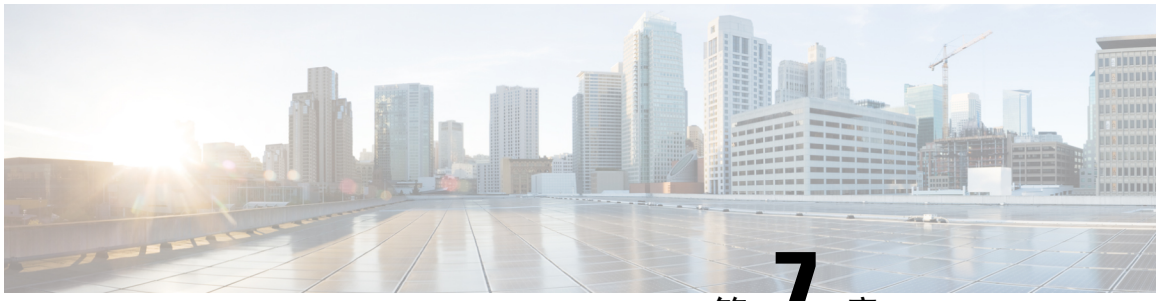
此选项将检测光标（用于在规则处理中遍历数据包负载）放在DCE/RPC末节数据的开头，不考虑前面的规则选项。如果存在 DCE/RPC 末节数据，则此选项匹配。

语法： `dce_stub_data;`

示例： `dce_stub_data;`

表 7: 范围格式

范围格式	Operator	说明
operator i		
	<	少于
	>	大于
	=	平分
	≠	不等于
	⩽	小于或等于
	⩾	大于或等于
j operator k		
	<>	大于 j 且小于 k
	<=>	大于或等于 j 且小于或等于 k



第 7 章

DNP3 检查器

DNP3 检查器概述

类型	检查器（服务）
使用方式	检测
实例类型	多实例
所需其他检查器	`stream_tcp`、 `stream_udp`
已启用	`false`

分布式网络协议 (DNP3) 是一种监控和数据采集 (SCADA) 协议，最初开发是为了在发电站之间提供一致的通信。DNP3 在水务、废弃物、运输行业中得到广泛使用。

`dnp3` 检查器会检测 DNP3 流量中的异常并分析 DNP3 协议。`dnp3` 入侵规则选项可访问某些 DNP3 协议字段。

DNP3 检查器参数

DNP3 TCP 端口配置

`绑定程序` 检查器定义 DNP3 TCP 端口配置。有关详细信息，请参阅绑定程序检查器概述， 第 13 页。

示例：

```
[
    {
```

```
        "when": {
            "role": "any",
            "service": "dnp3"
    },
        "use": {
            "type": "dnp3"
        }
    }
]
```

check_crc

指定是否验证DNP3链路层帧中包含的校验和。 dnp3 检查器会忽略具有无效校验和的帧。如果入侵规则 145:1 已启用，则 Snort 会针对无效校验和生成警报。

类型： boolean

有效值： true、 false

默认值： false

DNP3 检查器规则

启用 dnp3 检查器规则以 生成事件并在内联部署中丢弃攻击性数据包。

表 *8: DNP3* 检查器规则

GID:SID	Rule Message
145:1	DNP3 链路层帧包含错误的 CRC
145:2	DNP3 链路层帧已丢弃
145:3	DNP3 传输层网段在重组期间已丢弃
145:4	已清除 DNP3 重组缓冲区而未重组完整报文
145:5	DNP3 链路层帧使用保留地址
145:6	DNP3 应用层分片使用保留功能代码

DNP3 检查器入侵规则选项

dnp3_data

dnp3_data关键字将检测光标定位到应用层分片中DNP3数据的开头，无论前面的规则选项如何。使用此选项，您可以基于分段内的数据编写规则，而无需拆分数据和每 16 个字节添加 CRC。

语法： dnp3_data;

示例： dnp3_data;

dnp3_func

此选项与 DNP3 应用层请求/响应信头内的功能代码进行匹配。该代码可以是一个十进制数字或下方列表中的字符串。

类型： 字符串

语法： `dnp3_fuc:<DNP3_function>;`

有效值： *`DNP3_function`* 是以下值之一：

- 介于 `0` 和 `255`之间的整数
- `confirm` （对应于功能代码 `0`。）
- `read` （对应于功能代码 `1`。）
- `write` （对应于功能代码 `2`。)
- `select` （对应于功能代码 `3`。）
- `operate` （对应于功能代码 `4`。）
- `direct_operate` （对应于功能代码 `5`。）
- `direct_operat_nr` （对应于功能代码 `6`。）
- `imed_freeze` （对应于功能代码 `7`。）
- `imed_freeze_nr` （对应于功能代码 `8`。）
- `Freeze_clear` （对应于功能代码 `9`。）
- `Freeze_clear_nr` （对应于功能代码 `10`。）
- `Freeze_at_time` （对应于功能代码 `11`。）
- `freeze_at_time_nr` （对应于功能代码 `12`。)
- `cold_restart` （对应于功能代码 `13`。）
- `warm_restart` （对应于功能代码 `14`。）
- `Initialize_data` （对应于功能代码 `15`。）
- `Initialize_appl` （对应于功能代码 `16`。）
- `start_appl` （对应于功能代码 `17`。）
- `stop_appl` （对应于功能代码 `18`。）
- `save_config` （对应于功能代码 `19`。）
- `enable_unservices` （对应于功能代码 `20`。）
- `disable_un Requested` （对应于功能代码 `21`。）
- `Assign_class` （对应于功能代码 `22`。）

- delay_scale （对应于功能代码 23。）
- record_current_time （对应于功能代码 24。）
- open_file （对应于功能代码 25。）
- close_file （对应于功能代码 26。）
- delete_file （对应于功能代码 27。）
- get_file_info （对应于功能代码 28。）
- authenticate_file （对应于功能代码 29。）
- abort_file （对应于功能代码 30。）
- activate_config （对应于功能代码 31。）
- authenticate_req （对应于功能代码 32。）
- authenticate_err （对应于功能代码 33。）
- response （对应于功能代码 129。）
- unsolicited_response （对应于功能代码 130。）
- authenticate_resp （对应于功能代码 131。）

示例：

```
dnp3_func: 1;
dnp3_func: delete_file;
```

dnp3_ind

提供要根据 DNP3 应用层响应信头中的内部指示符标志进行匹配的内部指示符标志列表。如果在一个选项中提供多个标志，则在设置其中任何一个标志时，该规则都会触发。要对多个标志发出警报，请使用多个规则选项。

类型： 字符串

语法： `dnp3_ind: "<flag> ";`

有效值： 一个或多个 DNP3 内部指示器标志，其中 `flag` 为以下之一：

- all_stations
- class_1_events
- class_2_events
- class_3_events
- need_time
- local_control

- device_trouble
- device_restart
- no_func_code_support
- object_unknown
- parameter_error
- event_buffer_overflow
- already_executing
- config_corrupt
- reserved_2
- reserved_1

示例：

设备重启时或开始时间同步时发出警报：

```
dnp3_ind:"device_restart need_time";
```

关于 class_1 、class_2 和 class_3 事件的警报：

```
dnp3_ind:class_1_events; dnp3_ind:class_2_events; dnp3_ind:class_3_events;
```

dnp3_obj

在 DNP3 对象报头组及其变体上匹配。

类型： 整数

语法： dnp3_obj: <groupnum> 、<varnum>中所述：

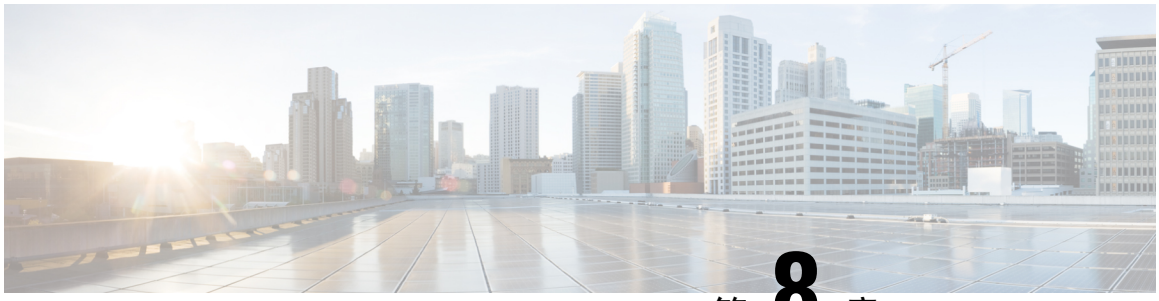
有效值： DNP3 对象组标识符和变体标识符，其中：

- *groupnum* 是指定 DNP3 对象组的 0 到 255 之间的整数。
- *varnum* 是介于 0 到 255 之间的整数，用于指定对象组内的变体。

示例：

DNP3 上的警报 日期和时间 对象：

```
dnp3_obj:50,1;
```



第 8 章

FTP 客户端检查器

FTP 客户端检查器概述

类型	检查器 (被动)
使用方式	检测
实例类型	多实例
所需其他检查器	`ftp_server`、 `stream_tcp`
已启用	`true`

文件传输协议 (FTP) 是用于通过 TCP/IP 在客户端和服务器之间传输文件的网络协议。在客户端和服务器建立连接后，客户端向服务器发出命令以将文件上传到服务器或从服务器下载文件，并解释服务器的响应。

`ftp_client` 检查器检查 FTP 命令通道上的响应并将其规范化。

给定 FTP 命令通道缓冲区， `ftp_client` 检查器解释 FTP 响应代码和消息。 `ftp_client` 检查器会强制执行参数的正确性，确定何时加密 FTP 命令连接，以及何时打开 FTP 数据通道。

FTP 客户端检查器参数

bounce

指定是否通过检查客户端发出的 `ftp 端口` 命令中的主机信息来检查 FTP 退回。当 `退回` 为 `true`时，如果 `ftp 端口` 命令中的主机信息与配置的客户端 IP 地址或主机信息不 上启用此选项，系统会生成警

报，并且在内联部署中丢弃违规的数据包。这可用于防止 FTP 退回攻击，并允许 FTP 数据通道目标与客户端不同的 FTP 连接。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

Ignore_telnet_erase_cmds

指定在规范化 FTP 命令通道时是否忽略擦除字符 (TNC EAC) 和擦除行字符 (TNC EAL) 的 telnet 转义序列。您应设置此参数以匹配 FTP 客户端处理 telnet 擦除命令的方式。请注意，新 FTP 客户端通常会忽略 Telnet 擦除命令，而旧客户端通常会进行处理。当 `ignore_telnet_erase_cmds` 参数为 `false` 时，检查器使用规则 125:1 生成警报，并在内联部署中丢弃违规数据包。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

max_resp_len

指定客户端接受的所有响应消息的最大长度（以字节为单位）。如果 FTP 响应的消息（3 位数返回代码后的所有内容）超过此长度，并且规则 125:6 已启用，则系统会生成警报，并且在内联部署中丢弃违规数据包。这用于检查 FTP 客户端内的缓冲区溢出漏洞攻击。

类型： 整数

有效范围： `0` 到 `4,294,967,295`（最大 32）

默认值： `4,294,967,295`

telnet_cmds

指定是否检查 FTP 命令通道上的 telnet 命令。如果存在此类命令，则可能表示 FTP 命令通道上存在逃逸尝试。

您可以启用规则 125:1 来生成此参数的事件，并且在内联部署中丢弃违规数据包。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

FTP 客户端检查器规则

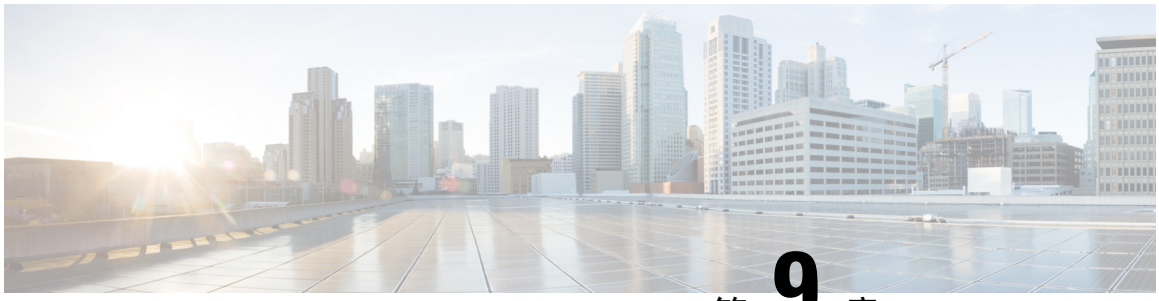
启用 `ftp_client` 检查器规则，以 生成事件并在内联部署中丢弃攻击性数据包。

表 *9: FTP* 客户端检查器规则

GID:SID	Rule Message
125:1	FTP 命令通道上的 TELNET cmd
125:6	FTP 响应消息过长
125:8	FTP 退回尝试

FTP 客户端检查器入侵规则选项

`ftp_client` 检查器没有任何入侵规则选项。



第 9 章

FTP 服务器检查器

FTP 服务器检查器概述

类型	检查器（服务）
使用方式	检测
实例类型	多实例
所需其他检查器	`ftp_client`、 `stream_tcp`
已启用	`true`

文件传输协议 (FTP) 是用于通过 TCP/IP 在客户端和服务器之间传输文件的网络协议。在客户端和服务器建立连接后，客户端向服务器发出命令以将文件上传到服务器或从服务器下载文件，并解释服务器的响应。

`ftp_server` 检查器检查 FTP 命令通道并将其规范化。

给定 FTP 命令通道缓冲区， `ftp_server` 检查器可识别 FTP 命令和参数，并检查参数的正确性。`ftp_server` 确定何时加密 FTP 命令连接以及何时打开 FTP 数据通道。

FTP 服务器检查器参数

FTP 服务器端口配置

绑定程序 检查器定义 FTP 服务器配置。有关详细信息，请参阅绑定程序检查器概述， 第 13 页。

示例：

```
[
    {
        "when": {
            "role":"any",
            "service":"ftp",
            "ports": ""
        },
        "use": {
            "type":"ftp_server"
        }
    }
]
```

chk_str_fmt

指定要检查字符串格式攻击的 FTP 命令列表。您可以启用规则 125:5 来生成警报，在内联部署中，当检查器检测到这种情况时，丢弃违规数据包。使用空格字符分隔多个命令。

类型： 字符串

有效值： 有效 FTP 命令的列表。

默认值： None

data_chan_cmds

指定要检查格式正确的 FTP 命令列表。使用空格字符分隔多个命令。

类型： 字符串

有效值： 下列一个或多个命令的列表： `PORT PASV LPRT LPSV EPRT EPSV`。

默认值： None

data_xfer_cmds

指定数据传输命令列表。检查命令格式是否正确。使用空格字符分隔多个命令。

类型： 字符串

有效值： 包含以下一个或多个命令的列表： `RETR STOR STOU APPE LIST NLST`。

默认值： None

file_put_cmds

指定 `PUT` 命令列表。检查命令格式是否正确。使用空格字符分隔多个命令。

类型： 字符串

有效值： 以下一个或多个命令的列表： `STOR STOU APPE`。

默认值： None

⚠

注意 请勿改变 `file_put_cmds` 参数，除非支持人员指示执行此操作。

file_get_cmds

指定 GET 命令列表。检查命令格式是否正确。使用空格字符分隔多个命令。

类型： 字符串

有效值： GET 命令列表，例如 RETR。

默认值： None

⚠

注意 请勿改变 file_get_cmds 参数，除非支持人员指示执行此操作。

encr_cmds

指定与安全连接相关的命令列表。检查命令格式是否正确。使用空格字符分隔多个命令。

类型： 字符串

有效值： 与安全连接相关的命令列表，例如： AUTH。

默认值： None

login_cmds

指定与登录过程相关的命令列表。检查命令格式是否正确。使用空格字符分隔多个命令。

类型： 字符串

有效值： 指定一个或多个命令的列表： USER、 PASS。

默认值： None

check_encrypted

指定是否检查加密会话中的结束加密命令。与 encrypted_traffic 参数一起使用。

您可以为此参数启用规则 125:7 到 生成事件并在内联部署中丢弃攻击性数据包 。

类型： boolean

有效值： true、 false

默认值： false

cmd_validity[]

FTP 命令数组以及检查器用来验证这些命令的条件。这些有效性检查会覆盖 ftp_server 检查器 (RFC 959) 执行的默认检查。

您可以启用规则 125:2 和 125:4 来生成事件，并且在线路部署中，丢弃此参数的违规数据包。

类型： 数组（对象）

示例：

```
{
   "cmd_validity": [
       {
           "command": "CWD",
           "format": "abc",
           "length": 250
       }
   ]
}
```

cmd_validity[].command

指定要验证的 FTP 命令的名称。

类型： 字符串

有效值： 用双引号括起来的有效 FTP 命令。

默认值： None

cmd_validity[].format

描述 `cmd_validity[].command`的有效格式

类型： 字符串

有效值： 采用以下格式之一：

- `int` – 参数必须为整数
- `number` - 该参数必须为 1 到 255 之间的整数
- `char` *`chars`* - 参数必须为来自 *`chars`*的单个字符，chars 个数，由一个或多个字符组成，且之间无分隔符。
- `date` *`datefmt`* - 参数遵循指定的格式，其中 *`datefmt`* 使用以下元素构建：
  - `#` = 编号
  - `C` = Char
  - `[]` = 括起来的可选格式
  - `|` = 或
  - `{}` = 所包含格式的选择
  - .+- 文字字符
- `字符串` - 参数为不受限制的字符串。
- `host_port` - 根据 RFC 959，参数必须是主机端口说明符。
- `long_host_port` - 根据 RFC 1639，参数必须是长主机端口说明符。
- `Extended_host_port` - 根据 RFC 2428，参数必须是扩展主机端口说明符。
- |`{}`中所述 - 参数必须是大括号内用 | 分隔的选项之一。

- {}、 [] - 参数必须是大括号内的选项之一。可选值包含在方括号内。

默认值： None

cmd_validity[].length

为 cmd_validity[].command 指定最大长度（以字节为单位），覆盖 def_max_param_len中定义的默认值。如果 FTP 命令的参数超过了 cmd_validity[].length，并启用了规则 125:3，Snort 会生成警报。使用 cmd_validity[].length 将特定命令限制为小参数值。

指定 0 以表示长度不受限制。

类型： 整数

有效范围： 0 到 4,294,967,295（最大 32）

默认值： 0

def_max_param_len

指定检查器允许的所有服务器处理的 FTP 命令的默认最大长度（以字节为单位）。使用 def_max_param_len 进行基本缓冲区溢出检测。（可以使用 cmd_validity[].length为单个命令覆盖此项。）您可以为此参数启用规则 125:3 变为 生成事件并在内联部署中丢弃攻击性数据包 。

指定 0 以表示长度不受限制。

类型： 整数

有效范围： 0 到 4,294,967,295（最大 32）

默认值： 100

encrypted_traffic

指定是否检查加密的 FTP 流量。与 check_encrypted 参数一起使用。您可以为此参数启用规则 125:7 到 生成事件并在内联部署中丢弃攻击性数据包 。

类型： boolean

有效值： true、 false

默认值： false

ftp_cmds

除 RFC 959 中所述的命令外，服务器还支持 FTP 命令列表。（例如，如果安装使用 RFC 775 中指定的“X”命令，则可以使用此参数将其添加到检查器。）

类型： 字符串

有效值： 以空格分隔的有效 FTP 命令列表，用双引号引起来。

默认值： None

ignore_data_chan

指定是否忽略 FTP 数据信道。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

Ignore_telnet_erase_cmds

指定在规范化 FTP 命令通道时是否忽略擦除字符 (TNC EAC) 和擦除行字符 (TNC EAL) 的 telnet 转义序列。设置 `ignore_telnet_erase_cmds` 以匹配您的FTP服务器处理telnet擦除命令的方式。请注意，新 FTP 客户端通常会忽略 Telnet 擦除命令，而旧客户端通常会进行处理。

如果未忽略 telnet擦除命令，并且启用了规则125:1，则 Snort 会生成一个事件，并且在内联部署中会丢弃违规的数据包。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

print_cmds

指定是否在初始化时打印此服务器的每个 FTP 命令的配置。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

telnet_cmds

指定是否检查 FTP 命令通道上的 telnet 命令。如果存在此类命令，则可能表示 FTP 命令通道上存在逃逸尝试。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

FTP 服务器检查器规则

启用 `ftp_server` 检查器规则以 生成事件并在内联部署中丢弃攻击性数据包。

表 *10: FTP* 服务器检查器规则

GID:SID	Rule Message
125:1	FTP 命令通道上的 TELNET 命令
125:2	无效的 FTP 命令
125:3	FTP 命令参数过长
125:4	FTP 命令参数格式错误
125:5	包含的 FTP 命令参数的潜在字符串格式
125:7	FTP 流量已加密
125:9	FTP 命令信道上存在回避（不完整）TELNET cmd

FTP 服务器检查器入侵规则选项

`ftp_server` 检查器没有任何入侵规则选项。

第 10 章

GTP 检查检查器

GTP 检查检查器概述

类型	检查器（服务）
使用方式	检测
实例类型	多实例
所需其他检查器	`stream_udp`
已启用	`false`

通用分组无线业务 (GPRS) 隧道协议 (GTP) 实现通过 GTP 核心网络进行通信。

`gtp_inspect` 检查器检测 GTP 流量中的异常，并将命令通道信令消息转发到规则引擎以进行检查。

GTP 检查检查器参数

GTP 检查服务和端口配置

`绑定程序` 检查器定义 GTP 检查 `服务` 和 `端口` 配置。有关详细信息，请参阅绑定程序检查器概述， 第 13 页。

示例：

```
[
   {
       "when": {
           "service": "gtp_inspect",
```

```
            "role": any
        },
        "use": {
            "type": "gtp_inspect"
        }
    },
    {
        "when": {
            "proto": "tcp",
            "role": "server",
            "ports": "2123 2152 3386"
        },
        "use": {
            "type": "gtp_inspect"
        }
    }
]
```

version

指定有效的 GTP 版本。

类型： 整数

有效值： 0, 1, 2

默认值： 2

messages[]

指定有关有效 GTP 消息的信息数组。

类型： 数组（对象）

示例：

```
{
    messages: [
        {
            "type": 0,
            "name": ""
        }
    ]
}
```

messages[].type

指定有效的 GTP 消息类型。请参阅 表 12: GTP 消息类型 表。

类型： 整数

有效范围： 0 至 255

默认值： None

messages[].name

指定有效的 GTP 消息名称。请参阅 表 12: GTP 消息类型 表。

类型： 字符串

有效值： 有效的 GTP 消息名称

默认值： None

infos[]

指定 GTP 信息元素数组。

类型： 数组（对象）

示例：

```
{
    infos: [
        {
            "type": 0,
            "name": "echo_request",
            "length": 0
        }
    ]
}
```

infos[].type

指定有效的 GTP 元素类型代码。请参阅 表 13: GTP 信息元素 表。

类型： 整数

有效范围： 0 至 255

默认值： 0

infos[].name

指定有效的 GTP 元素名称。

类型： 字符串

有效值： 有效的 GTP 信息元素名称。请参阅 表 13: GTP 信息元素 表。

infos[].length

指定有效 GTP 信息元素的长度。

类型： 整数

有效范围： 0 至 255

默认值： 0

GTP 检查检查器规则

启用 gtp_inspect 检查器规则以 生成事件并在内联部署中丢弃攻击性数据包。

表 *11: GTP* 检查器规则

GID:SID	Rule Message
143:1	信息长度无效
143:2	信息元素长度无效
143:3	信息元素顺序错误
143:4	TEID 缺失

GTP 检查检查器入侵规则选项

通过 `gtp_inspect` 检查器入侵规则选项，您可以检查 GTP 命令通道中的 GTP 版本、消息类型和信息元素。

不能将 GTP 选项与 `content` 或 `byte_jump`结合使用。必须在使用 `gtp_info` 或 `gtp_type` 的每个规则中使用 `gtp_version`。

gtp_version

根据 GTP 控制消息的版本检查指定的 GTP 版本。

类型： 整数

语法： `gtp_version:<version>;`

有效值： `0, 1, 2`

示例： `gtp_version: 1;`

gtp_type

每条 GTP 消息由一种消息类型标识，消息类型由一个数值和一个字符串组成。根据 GTP 消息的类型检查指定的 GTP 类型。

可以为消息类型指定定义的十进制值，可以指定定义的字符串，或者指定包含这两项的任意组合的逗号分隔列表，如以下示例所示：

类型： 字符串

语法： `gtp_type: <message_type>;`

有效值： 在 GTP 消息类型表中列出。请参阅 表 12: GTP 消息类型 表。

示例： `gtp_type: "10, 11, echo_request";`

系统使用 OR 操作来匹配列出的每个值或字符串。值和字符串的列出顺序并不重要。列表中的任何一个值或字符串均与此关键字匹配。如果尝试保存包含无法识别的字符串或超出范围的值的规则，系统将生成错误。

请注意，不同的 GTP 版本有时会对同一种消息类型使用不同的值。例如，`sgsn_context_request` 这一消息类型在 GTPv0 和 GTPv1 中值是 `50`，但在 GTPv2 中值是 `130`。

`gtp_type` 选项匹配不同的值，具体取决于数据包中的版本号。例如，在 GTPv0 或 GTPv1 数据包中，`sgsn_context_request` 信息匹配值 `50`，在 GTPv2 数据包中，则匹配值 `130`。如果数据包中的消息类型值不是在数据包中指定的版本的已知值，此选项不会匹配数据包。

如果为消息类型指定一个整数，则当消息类型与 GTP 数据包中的该值匹配时，关键字将会匹配，无论数据包中指定的版本如何。

`gtp_message_type` 是 表 12: GTP 消息类型 表中的数值或关键字。

表 *12: GTP* 消息类型

类型	版本 0 的名称	版本 1 的名称	版本 2 的名称
1	echo_request	echo_request	echo_request
2	echo_response	echo_response	echo_response
3	version_not_supported	version_not_supported	version_not_supported
4	node_alive_request	node_alive_request	不适用
5	node_alive_response	node_alive_response	不适用
6	redirection_request	redirection_request	不适用
7	redirection_response	redirection_response	不适用
16	create_pdp_context_request	create_pdp_context_request	不适用
17	create_pdp_context_response	create_pdp_context_response	不适用
18	update_pdp_context_request	update_pdp_context_request	不适用
19	update_pdp_context_response	update_pdp_context_response	不适用
20	delete_pdp_context_request	delete_pdp_context_request	不适用
21	delete_pdp_context_response	delete_pdp_context_response	不适用
22	create_aa_pdp_context_request	init_pdp_context_activation_request	不适用
23	create_aa_pdp_context_response	init_pdp_context_activation_response	不适用
24	delete_aa_pdp_context_request	不适用	不适用
25	delete_aa_pdp_context_response	不适用	不适用
26	error_indication	error_indication	不适用
27	pdu_notification_request	pdu_notification_request	不适用

类型	版本 0 的名称	版本 1 的名称	版本 2 的名称
28	pdu_notification_response	pdu_notification_response	不适用
29	pdu_notification_reject_request	pdu_notification_reject_request	不适用
30	pdu_notification_reject_response	pdu_notification_reject_response	不适用
31	不适用	supported_ext_header_notification	不适用
32	send_routing_info_request	send_routing_info_request	create_session_request
33	send_routing_info_response	send_routing_info_response	create_session_response
34	failure_report_request	failure_report_request	modify_bearer_request
35	failure_report_response	failure_report_response	modify_bearer_response
36	note_ms_present_request	note_ms_present_request	delete_session_request
37	note_ms_present_response	note_ms_present_response	delete_session_response
38	不适用	不适用	change_notification_request
39	不适用	不适用	change_notification_response
48	identification_request	identification_request	不适用
49	identification_response	identification_response	不适用
50	sgsn_context_request	sgsn_context_request	不适用
51	sgsn_context_response	sgsn_context_response	不适用
52	sgsn_context_ack	sgsn_context_ack	不适用
53	不适用	forward_relocation_request	不适用
54	不适用	forward_relocation_response	不适用
55	不适用	forward_relocation_complete	不适用
56	不适用	relocation_cancel_request	不适用
57	不适用	relocation_cancel_response	不适用
58	不适用	forward_srns_contex	不适用
59	不适用	forward_relocation_complete_ack	不适用
60	不适用	forward_srns_contex_ack	不适用
64	不适用	不适用	modify_bearer_command

类型	版本 0 的名称	版本 1 的名称	版本 2 的名称
65	不适用	不适用	modify_bearer_failure_indication
66	不适用	不适用	delete_bearer_command
67	不适用	不适用	delete_bearer_failure_indication
68	不适用	不适用	bearer_resource_command
69	不适用	不适用	bearer_resource_failure_indication
70	不适用	ran_info_relay	downlink_failure_indication
71	不适用	不适用	trace_session_activation
72	不适用	不适用	trace_session_deactivation
73	不适用	不适用	stop_paging_indication
95	不适用	不适用	create_bearer_request
96	不适用	mbms_notification_request	create_bearer_response
97	不适用	mbms_notification_response	update_bearer_request
98	不适用	mbms_notification_reject_request	update_bearer_response
99	不适用	mbms_notification_reject_response	delete_bearer_request
100	不适用	create_mbms_context_request	delete_bearer_response
101	不适用	create_mbms_context_response	delete_pdn_request
102	不适用	update_mbms_context_request	delete_pdn_response
103	不适用	update_mbms_context_response	不适用
104	不适用	delete_mbms_context_request	不适用
105	不适用	delete_mbms_context_response	不适用
112	不适用	mbms_register_request	不适用
113	不适用	mbms_register_response	不适用
114	不适用	mbms_deregister_request	不适用
115	不适用	mbms_deregister_response	不适用
116	不适用	mbms_session_start_request	不适用
117	不适用	mbms_session_start_response	不适用

类型	版本 0 的名称	版本 1 的名称	版本 2 的名称
118	不适用	mbms_session_stop_request	不适用
119	不适用	mbms_session_stop_response	不适用
120	不适用	mbms_session_update_request	不适用
121	不适用	mbms_session_update_response	不适用
128	不适用	ms_info_change_request	identification_request
129	不适用	ms_info_change_response	identification_response
130	不适用	不适用	sgsn_context_request
131	不适用	不适用	sgsn_context_response
132	不适用	不适用	sgsn_context_ack
133	不适用	不适用	forward_relocation_request
134	不适用	不适用	forward_relocation_response
135	不适用	不适用	forward_relocation_complete
136	不适用	不适用	forward_relocation_complete_ack
137	不适用	不适用	forward_access
138	不适用	不适用	forward_access_ack
139	不适用	不适用	relocation_cancel_request
140	不适用	不适用	relocation_cancel_response
141	不适用	不适用	configuration_transfer_tunnel
149	不适用	不适用	detach
150	不适用	不适用	detach_ack
151	不适用	不适用	cs_paging
152	不适用	不适用	ran_info_relay
153	不适用	不适用	alert_mme
154 种	不适用	不适用	alert_mme_ack
155	不适用	不适用	ue_activity
156	不适用	不适用	ue_activity_ack

类型	版本 0 的名称	版本 1 的名称	版本 2 的名称
160	不适用	不适用	create_forward_tunnel_request
161	不适用	不适用	create_forward_tunnel_response
162	不适用	不适用	suspend
163	不适用	不适用	suspend_ack
164	不适用	不适用	继续执行
165	不适用	不适用	resume_ack
166	不适用	不适用	create_indirect_forward_tunnel_request
167	不适用	不适用	create_indirect_forward_tunnel_response
168	不适用	不适用	delete_indirect_forward_tunnel_request
169	不适用	不适用	delete_indirect_forward_tunnel_response
170	不适用	不适用	_access_bearer_request
171	不适用	不适用	release_access_bearer_response
176	不适用	不适用	downlink_data
177	不适用	不适用	downlink_data_ack
179	不适用	不适用	pgw_restart
180 个	不适用	不适用	pgw_restart_ack
200	不适用	不适用	update_pdn_request
201	不适用	不适用	update_pdn_response
211	不适用	不适用	modify_access_bearer_request
212	不适用	不适用	modify_access_bearer_response
231	不适用	不适用	mbms_session_start_request
232	不适用	不适用	mbms_session_start_response
233	不适用	不适用	mbms_session_update_request
234	不适用	不适用	mbms_session_update_response
235	不适用	不适用	mbms_session_stop_request
236	不适用	不适用	mbms_session_stop_response

类型	版本 0 的名称	版本 1 的名称	版本 2 的名称
240	data_record_transfer_request	data_record_transfer_request	不适用
241	data_record_transfer_response	data_record_transfer_response	不适用
254	不适用	end_marker	不适用
255	pdu	pdu	不适用

gtp_info

一条 GTP 消息可以包含多个信息元素，其中的每一个元素均由已定义的一个数值和一个字符串来识别。可以使用 `gtp_info` 选项在指定的信息元素开头开始检查，并将检查限于该信息元素。

可以为信息元素指定已定义的十进制值或字符串。可以指定一个值或字符串，也可以在一个规则中使用多个 `gtp_info` 选项来检查多个信息元素。

如果一条消息包含相同类型的多个信息元素，系统将会全部检查这些元素来进行匹配。如果信息元素按无效顺序出现，将仅检查最后一个实例。

根据版本的不同，GTP 消息可以对同一信息元素使用不同的值。例如，`cause` 这个信息元素在 GTPv0 和 GTPv1 中值是 `1`，但在 GTPv2 中值是 `2`。

`gtp_info` 选项匹配不同的值，具体取决于数据包中的版本号。在上述示例中，在 GTPv0 或 GTPv1 数据包中，此关键字匹配信息元素值 `1`，在 GTPv2 数据包中，则匹配值 `2`。如果数据包中的信息元素值不是在数据包中指定的版本的已知值，此选项不会匹配数据包。

如果为信息元素指定一个整数，则当消息类型与 GTP 数据包中的该值匹配时，选项将会匹配，无论数据包中指定的版本如何。

类型： 字符串

语法： `gtp_info: <identifier>;`

有效值： 在 表 13: GTP 信息元素 表中列出。

示例： `gtp_info: "qos";`

表 13: GTP 信息元素

类型	版本 0 的名称	版本 1 的名称	版本 2 的名称
1	cause	cause	imsi
2	imsi	imsi	cause
3	rai	rai	recovery
4	tlli	tlli	不适用
5	p_tmsi	p_tmsi	不适用
6	qos	不适用	不适用

类型	版本 0 的名称	版本 1 的名称	版本 2 的名称
8	recording_required	recording_required	不适用
9	authentication	authentication	不适用
10	不适用	不适用	不适用
11	map_cause	map_cause	不适用
12	p_tmsi_sig	p_tmsi_sig	不适用
13	ms_validated	ms_validated	不适用
14	recovery	recovery	不适用
15	selection_mode	selection_mode	不适用
16	flow_label_data_1	teid_1	不适用
17	flow_label_signalling	teid_control	不适用
18	flow_label_data_2	teid_2	不适用
19	ms_unreachable	teardown_ind	不适用
20	不适用	nsapi	不适用
21	不适用	ranap	不适用
22	不适用	rab_context	不适用
23	不适用	radio_priority_sms	不适用
24	不适用	radio_priority	不适用
25	不适用	packet_flow_id	不适用
26	不适用	charging_char	不适用
27	不适用	trace_ref	不适用
28	不适用	trace_type	不适用
29	不适用	ms_unreachable	不适用
71	不适用	不适用	apn
72	不适用	不适用	ambr
73	不适用	不适用	ebi
74	不适用	不适用	ip_addr

类型	版本 0 的名称	版本 1 的名称	版本 2 的名称
75	不适用	不适用	mei
76	不适用	不适用	msisdn
77	不适用	不适用	indication
78	不适用	不适用	pco
79	不适用	不适用	paa
80	不适用	不适用	bearer_qos
80	不适用	不适用	flow_qos
82	不适用	不适用	rat_type
83	不适用	不适用	serving_network
84	不适用	不适用	bearer_tft
85	不适用	不适用	tad
86	不适用	不适用	uli
87	不适用	不适用	f_teid
88	不适用	不适用	tmsi
89	不适用	不适用	cn_id
90	不适用	不适用	s103pdf
91	不适用	不适用	s1udf
92	不适用	不适用	delay_value
93	不适用	不适用	bearer_context
94	不适用	不适用	charging_id
95	不适用	不适用	charging_char
96	不适用	不适用	trace_info
97	不适用	不适用	bearer_flag
99	不适用	不适用	pdn_type
100	不适用	不适用	pti
101	不适用	不适用	drx_parameter

类型	版本 0 的名称	版本 1 的名称	版本 2 的名称
103	不适用	不适用	gsm_key_tri
104	不适用	不适用	umts_key_cipher_quin
105	不适用	不适用	gsm_key_cipher_quin
106	不适用	不适用	umts_key_quin
107	不适用	不适用	eps_quad
108	不适用	不适用	umts_key_quad_quin
109	不适用	不适用	pdn_connection
110	不适用	不适用	pdn_number
111	不适用	不适用	p_tmsi
112	不适用	不适用	p_tmsi_sig
113	不适用	不适用	hop_counter
114	不适用	不适用	ue_time_zone
115	不适用	不适用	trace_ref
116	不适用	不适用	complete_request_msg
117	不适用	不适用	guti
118	不适用	不适用	f_container
119	不适用	不适用	f_cause
120	不适用	不适用	plmn_id
121	不适用	不适用	target_id
123	不适用	不适用	packet_flow_id
124	不适用	不适用	rab_contex
125	不适用	不适用	src_rnc_pdcp
126	不适用	不适用	udp_src_port
127	charge_id	charge_id	apn_restriction
128	end_user_address	end_user_address	selection_mode
129	mm_context	mm_context	src_id

类型	版本 0 的名称	版本 1 的名称	版本 2 的名称
130	pdp_context	pdp_context	不适用
131	apn	apn	change_report_action
132	protocol_config	protocol_config	fq_csid
133	gsn	gsn	信道
134	msisdn	msisdn	emlpp_pri
135	不适用	qos	node_type
136	不适用	authentication_qu	fqdn
137	不适用	tft	ti
138	不适用	target_id	mbms_session_duration
139	不适用	utran_trans	mbms_service_area
140	不适用	rab_setup	mbms_session_id
141	不适用	ext_header	mbms_flow_id
142	不适用	trigger_id	mbms_ip_multicast
143	不适用	omc_id	mbms_distribution_ack
144 个	不适用	ran_trans	rfsp_index
145	不适用	pdp_context_pri	uci
146	不适用	addi_rab_setup	csg_info
147	不适用	sgsn_number	csg_id
148	不适用	common_flag	cmi
149	不适用	apn_restriction	service_indicator
150	不适用	radio_priority_lcs	detach_type
151	不适用	rat_type	ldn
152	不适用	user_loc_info	node_feature
153	不适用	ms_time_zone	mbms_time_to_transfer
154 种	不适用	imei_sv	throttling
155	不适用	camel	ARP

类型	版本 0 的名称	版本 1 的名称	版本 2 的名称
156	不适用	mbms_ue_context	epc_timer
157	不适用	tmp_mobile_group_id	signalling_priority_indication
158	不适用	rim_routing_addr	tmgi
159	不适用	mbms_config	mm_srvcc
160	不适用	mbms_service_area	flags_srvcc
161	不适用	src_rnc_pdcp	nmbr
162	不适用	addi_trace_info	不适用
163	不适用	hop_counter	不适用
164	不适用	plmn_id	不适用
165	不适用	mbms_session_id	不适用
166	不适用	mbms_2g3g_indicator	不适用
167	不适用	enhanced_nsapi	不适用
168	不适用	mbms_session_duration	不适用
169	不适用	addi_mbms_trace_info	不适用
170	不适用	mbms_session_repetition_num	不适用
171	不适用	mbms_time_to_data	不适用
173	不适用	bss	不适用
174	不适用	cell_id	不适用
175	不适用	pdu_num	不适用
177	不适用	mbms_bearer_capab	不适用
178	不适用	rim_routing_disc	不适用
179	不适用	list_pfc	不适用
180 个	不适用	ps_xid	不适用
181	不适用	ms_info_change_report	不适用
182	不适用	direct_tunnel_flags	不适用
183	不适用	correlation_id	不适用

类型	版本 0 的名称	版本 1 的名称	版本 2 的名称
184	不适用	bearer_control_mode	不适用
185	不适用	mbms_flow_id	不适用
186	不适用	mbms_ip_multicast	不适用
187	不适用	mbms_distribution_ack	不适用
188	不适用	reliable_inter_rat_handover	不适用
189	不适用	rfsp_index	不适用
190	不适用	fqdn	不适用
191	不适用	evolved_allocation1	不适用
192 个	不适用	evolved_allocation2	不适用
193	不适用	extended_flags	不适用
194	不适用	uci	不适用
195	不适用	csg_info	不适用
196	不适用	csg_id	不适用
197	不适用	cmi	不适用
198	不适用	apn_ambr	不适用
199	不适用	ue_network	不适用
200	不适用	ue_ambr	不适用
201	不适用	apn_ambr_nsapi	不适用
202	不适用	ggsn_backoff_timer	不适用
203	不适用	signalling_priority_indication	不适用
204	不适用	signalling_priority_indication_nsapi	不适用
205	不适用	high_bitrate	不适用
206	不适用	max_mbr	不适用
251	charging_gateway_addr	charging_gateway_addr	不适用
255	private_extension	private_extension	private_extension

第 11 章

HTTP 检查器

HTTP 检查检查器概述

类型	检查器（服务）
使用方式	检测
实例类型	多实例
所需其他检查器	`stream_tcp`
已启用	`true`

超文本传输协议(HTTP)是支持在客户端和服务器之间交换超媒体（音频、视频、图像和文本）的应用层协议。HTTP是无状态协议，需要进行可靠的消息传输。客户端与服务器之间的通信采用HTTP请求和响应的形式。

HTTP/1.1 服务器通常使用 TCP/IP 上的端口 80。安全版本的 HTTP（HTTP/TLS 或 HTTPS）使用端口 443。HTTP 定义了协议中的访问控制和身份验证机制。

HTTP/2包含一些改进，以提高速度和推送比客户端请求更多的信息，但运行的端口和协议与HTTP/1.1相同。HTTP/2 特定规则使用 `service:http2`进行配置。

HTTP/3 是无连接协议，使用 QUIC（快速 UDP 互联网连接）协议而不是 TCP，并且可以支持具有更好丢失恢复功能的更多活动流。HTTP/3 使用的消息传递与 HTTP 的早期版本相同。HTTP/3 特定规则使用 `service:http3`进行配置。

HTTP 检查器以相同的方式支持 HTTP 的所有三个版本。

`http_inspect` 检查器检测并分析 HTTP 消息的协议数据单元 (PDU)。`http_inspect` 从 TCP 数据流接收 TCP 负载，并检查封装的 HTTP 消息。

HTTP 检查器可以检测以下 HTTP 消息部分：

- 请求行
- 状态行
- 信头
- Content-Length 消息正文（内容长度报头定义的消息正文）
- 分块消息正文
- 上一个消息正文（没有 Content-Length 报头的消息正文
- 帧尾

`http_inspect` 检查器检测并规范化所有 HTTP 信头字段以及 HTTP URI 的组件。`http_inspect` 检查器不会规范化 TCP 端口。

`http_inspect` 检查器可以检测四种类型的 URI：

- 星号 (*)：未规范化
- 颁发机构：与 HTTP CONNECT 方法配合使用的 URI
- 来源：以斜杠开头的 URI（不显示方案或授权）
- 绝对：包括方案、主机和绝对路径的 URI

HTTP URI 可以包括：

- 方案（ftp、http 或 https）
- 主机（服务器的域名）
- TCP 端口
- 路径（目录和文件）
- 查询（请求参数）
- 片段（文件的一部分）

您可以将 `http_inspect` 检查器配置为对 HTTP 消息的部分发出警报。例如：

- 指定要从 HTTP 请求或响应正文读取的字节数
- 启用 JavaScript 检测和规范化
- 处理各种类型的文件解压缩
- 自定义 HTTP URI 的解码

注释 http_inspect 检查器可以部分检查流 TCP 负载。

配置 HTTP 检查检查器的最佳实践

配置 http_inspect 检查器时，请考虑以下最佳实践：

- 如果您的 HTTP 流量包括大型视频文件，请设置 request_depth 和 response_depth 参数。
- 对 HTTP URI 检查参数使用默认设置：

```
"utf8": "true"
"plus_to_space": "true"
"percent_u": "true"
"utf8_bare_byte": "true"
"iis_unicode": "true"
"iis_double_decode": "true"
```

HTTP 检查器检查器参数

HTTP 服务配置

绑定程序 检查器定义 HTTP 服务配置。有关详细信息，请参阅绑定程序检查器概述，第 13 页。

示例：

```
[
    {
        "when": {
             "service": "http",
             "role": any
        },
        "use": {
            "type": "http_inspect"
        }
    }
]
```

request_depth

指定要从 HTTP 消息请求正文读取的字节数。

指定 -1 以对要检查的字节数不设限制。我们建议您指定 request_depth 和 response_depth 参数，以限制要分析的 HTTP 正文数据量。

要仅检查 HTTP 信头，请将 request_depth 设置为 0。

类型： 整数

有效范围： -1 到 9,007,199,254,740,992（最大 53）

默认值： -1

response_depth

指定要从 HTTP 消息响应正文读取的字节数。

指定 -1 以对要检查的字节数不设限制。我们建议您指定 request_depth 和 response_depth 参数，以限制要分析的 HTTP 正文数据量。

要仅检查 HTTP 信头，请将 response_length 设置为 0。

类型： 整数

有效范围： -1 到 9,007,199,254,740,992（最大 53）

默认值： -1

解压缩

指定在检查邮件正文之前是否将 gzip 文件解压缩并缩小邮件正文。关闭解压缩时，HTTP 检查器无法处理 HTTP 消息正文的所有部分。http_inspect 检查器可以处理 HTTP 信头。

类型： boolean

有效值： true、false

默认值： true

maximum_host_length

指定 主机 HTTP 信头值中允许的最大字节数。

指定 -1，则表示对信头值长度没有限制。

类型： 整数

有效范围： -1 到 9,007,199,254,740,992（最大 53）

默认值： -1

maximum_chunk_length

指定 HTTP 消息正文数据块中允许的最大字节数。

指定 -1 以对 HTTP 数据块中的字节数不设限制。

类型： 整数

有效范围： -1 到 9,007,199,254,740,992（最大 53）

默认值： -1

normalize_utf

指定是否规范化HTTP响应正文中的UTF编码（UTF-8、UTF-7、UTF-16LE、UTF-16BE、UTF-32LE和 UTF-32BE）。 `http_inspect` 检查器根据 HTTP `Content-Type` 信头确定 UTF 字符编码。

类型： boolean

有效值： `true`、`false`

默认值： `true`

decompress_pdf

指定是否解压缩在 HTTP 响应正文中找到的 `application/pdf` (PDF) 文件的 deflate 兼容压缩部分。`http_inspect` 检查器使用 `/FlateDecode` 流过滤器解压缩 PDF 文件。

类型： boolean

有效值： `true`、`false`

默认值： `false`

decompress_swf

指定是否解压缩在 HTTP 响应正文中找到的 `application/vnd.adobe.flash-movie` (SWF) 文件。

注释 您只能解压缩在 HTTP GET 响应中找到的文件的压缩部分。

类型： boolean

有效值： `true`、`false`

默认值： `false`

decompress_vba

指定是否解压缩在 HTTP 响应正文中找到的 Microsoft Office Visual Basic for Applications 宏文件。

类型： boolean

有效值： `true`、`false`

默认值： `false`

decompress_zip

指定是否解压缩在 HTTP 响应正文中找到的 `application/zip` (ZIP) 文件。

类型： boolean

有效值： `true`、`false`

默认值： `false`

script_detection

指定是否在检测脚本结束元素 (<\script>) 后检查 JavaScript 内容。当 `http_inspect` 检测到脚本末尾时，它会立即转发部分读取的消息正文以便及早检测。脚本检测使 Snort 能够快速阻止可能包含恶意 JavaScript 的响应消息。

类型： boolean

有效值： `true`、`false`

默认值： `false`

normalize_javascript

指定是否使用传统机制规范化 HTTP 响应正文中的 JavaScript。此选项配置旧版 JavaScript 规范器。`http_inspect` 检查器可规范化混淆的 JavaScript 数据，包括 `unescape` 和 `decodeURI` 函数以及 `String.fromCharCode` 方法。HTTP 检查器会规范化 unescape、decodeURI 和 decryptURIComponent 函数中的编码：%XX、%uXXXX、XX 和 uXXXXi。

`http_inspect` 器检测连续空格，并将其规范化为一个空格。当 `normalize_javascript` 启用时，您可以设置 `max_javascript_whitespaces` 以限制模糊 JavaScript 中连续空格的数量。

类型： boolean

有效值： `true`、`false`

默认值： `false`

js_norm_bytes_depth

指定要规范化的输入 JavaScript 字节的数量。此选项特定于增强型 JavaScript 规范器。

注释 如果使用增强型 JavaScript 规范器，则会使用轻量级安全软件包 (LSP) 和 Snort 3 中的默认设置。JavaScript 特定配置会从网络分析策略 (NAP) 用户界面阻止。要覆盖默认设置并自定义规范器设置，您可以修改 `/ftd/app_data/Volume/root1/ngfw/var/cisco/deploy` 中的 `NAPOverride.lua` 文件。

`http_inspect` 检查器检测连续空格，并将其规范化为一个空格。检查器会跟踪不同 PDU 中的脚本（其中 start <script> is in one PDU and the end </script> 位于另一个 PDU 中），以有效地规范化流量。新缓冲区 `js_data` 已被添加到 Snort 3 IPS 缓冲区，该缓冲区使用实时 (JIT) 方法来检测和规范化 JavaScript 代码，其中仅在规则中使用此选项时才会调用规范器。

`http_inspect` 检查器规范化与 JavaScript 代码关联的函数名称、变量名称和标签名称。此外，检查器会使用 application/javascript 或类似的 MIME 类型规范以外部脚本形式传输的 JavaScript 代码。规范器会自动执行分号插入，其中 JavaScript 功能不会根据客户端的原始输入进行更改。

`http_inspect` 检查器可规范化混淆的 JavaScript 数据，包括 `unescape`、 `decodeURI` 和 `decryptURIComponent` 函数以及 `String.fromCharCode` 和 `String.fromCodePoint` 方法。HTTP 检查器的增强规范化程序可规范化 unescape、decodeURI 和 decodeURIComponent 函数中的编码：%XX、%uXXXX、\uXX、\u{XXXX}\xXX、十进制代码点和十六进制代码点。

http_inspect 检查器还会对 Javascript 加号 (+) 运算符进行规范化，并使用该运算符连接字符串。

指定 -1，以便对 JavaScript 字节数不设限制。

类型： 整数

有效范围： -1 到 9,007,199,254,740,992（最大 53）

默认值： -1

js_norm_identifier_depth

指定要规范化的唯一 JavaScript 标识符的最大数量。此选项特定于增强型 JavaScript 规范器。

注释 如果使用增强型 JavaScript 规范器，则会使用轻量级安全软件包 (LSP) 和 Snort 3 中的默认设置。JavaScript特定配置会从网络分析策略(NAP)用户界面阻止。要覆盖默认设置并自定义规范器设置，您可以修改 /ftd/app_data/Volume/root1/ngfw/var/cisco/deploy 中的 NAPOverride.lua文件。

类型： 整数

有效范围： 0 至 65536

默认值： 65536

js_norm_max_bracket_depth

指定要规范化的 JavaScript 括号的最大嵌套深度。此选项特定于增强型 JavaScript 规范器。

注释 如果使用增强型 JavaScript 规范器，则会使用轻量级安全软件包 (LSP) 和 Snort 3 中的默认设置。JavaScript特定配置会从网络分析策略(NAP)用户界面阻止。要覆盖默认设置并自定义规范器设置，您可以修改 /ftd/app_data/Volume/root1/ngfw/var/cisco/deploy 中的 NAPOverride.lua文件。

类型： 整数

有效范围： 1 至 65535

默认值： 256

js_norm_max_scope_depth

指定要规范化的 JavaScript 范围嵌套的最大深度。此选项特定于增强型 JavaScript 规范器。

注释 如果使用增强型 JavaScript 规范器，则会使用轻量级安全软件包 (LSP) 和 Snort 3 中的默认设置。JavaScript特定配置会从网络分析策略(NAP)用户界面阻止。要覆盖默认设置并自定义规范器设置，您可以修改 /ftd/app_data/Volume/root1/ngfw/var/cisco/deploy 中的 NAPOverride.lua文件。

类型： 整数

有效范围： 1 至 65535

默认值： 256

js_norm_max_tmpl_nest

指定要规范化的 JavaScript 模板文字的最大深度。此选项特定于增强型 JavaScript 规范器。

注释 如果使用增强型 JavaScript 规范器，则会使用轻量级安全软件包 (LSP) 和 Snort 3 中的默认设置。JavaScript特定配置会从网络分析策略(NAP)用户界面阻止。要覆盖默认设置并自定义规范器设置，您可以修改 /ftd/app_data/Volume/root1/ngfw/var/cisco/deploy 中的 NAPOverride.lua文件。

类型： 整数

有效范围： 0 至 255

默认值： 32

max_javascript_whitespaces

指定 JavaScript 模糊数据中允许的最大连续空格数。

类型： 整数

有效范围： 1 至 65535

默认值： 200

percent_u

指定是否规范化 %uNNNN 和 %UNNNN 编码。四个 N 字符表示与 Microsoft 互联网信息服务 (IIS) Unicode 代码点关联的十六进制编码的值。合法的客户端很少使用 %u 编码，因此我们建议您对使用 %u 编码的 HTTP 流量进行规范化。

类型： boolean

有效值： true、false

默认值： false

utf8

指定是否规范化 URI 中的标准 UTF-8 Unicode 序列。`http_inspect` 检查器可以将两个或三个字节 UTF-8 字符规范化为单个字节。

类型： boolean

有效值： `true`、`false`

默认值： `true`

utf8_bare_byte

指定是否规范化包括非 URL 或百分比编码的字节的 UTF-8 字符。我们建议您启用 `utf8_bare_byte` 参数。

类型： boolean

有效值： `true`、`false`

默认值： `false`

iis_unicode

指定是否使用 Unicode 代码点规范化 HTTP 消息中的字符。

注释 我们建议启用 `iis_unicode` 参数。Unicode 在攻击和规避尝试中很常见。

类型： boolean

有效值： `true`、`false`

默认值： `false`

iis_unicode_code_page

指定是否使用 IIS Unicode 映射文件的代码页。

类型： 整数

有效范围： `1` 至 `65535`

默认值： `1252`

iis_double_decode

指定是否通过执行 URL 编码字符的双重解码来规范化字符。通过在请求 URI 中形成两条通道，解码 IIS 双编码流量。我们建议您启用 `iis_ouble_decode` 参数。重复编码通常仅见于攻击情景。

类型： boolean

有效值： `true`、`false`

默认值： `true`

oversize_dir_length

指定 URL 目录允许的最大字节数。

类型： 整数

有效范围： 1 至 65535

默认值： 300

backslash_to_slash

指定是否用 URI 中的反斜线 (\) 替换正斜线 (/)。

类型： boolean

有效值： true、false

默认值： true

plus_to_space

指定是否将加号 (+) 替换为<sp>在 URI 中检索到的数据。

类型： boolean

有效值： true、false

默认值： true

simplify_path

指定是否将 URI 目录路径简化为最简单形式。包含额外遍历的 URI 目录路径可能包括：．、 ..和 /。

类型： boolean

有效值： true、false

默认值： true

xff_headers

指定要检查的 X-Forwarded-For HTTP 信头的类型。在 xff_headers 参数中，按优先级从高到低列出 X-Forwarded-For 信头。

您可以定义自定义 X-Forwarded-For 类型信头。携带原始客户端 IP 地址的 HTTP 信头可以有一个供应商特定的信头名称。在这种情况下， xff_headers 参数提供了一种将自定义信头引入 HTTP 检查器的方法。

xff_headers 默认值为 x-forwarded-for true-client-ip，这是两个常见的信头。如果两个默认信头都在流中存在，则 x-forwarded-for 优先于 true-client-ip。指定多个 X-Forwarded-For HTTP 信头时，请使用空格分隔信头名称。

类型： 字符串

有效值： x-forwarded-for、 true-client-ip

默认值： `x-forwarded-for true-client-ip`

HTTP 检查检查器规则

启用 `http_inspect` 检查器规则以 生成事件并在内联部署中丢弃攻击性数据包。

表 *14: HTTP* 检查检查器规则

GID:SID	Rule Message
119:1	URI 包含非保留字符的百分比编码
119:2	URI 按百分比编码，结果再次按百分比编码
119:3	URI 使用非标准 %u 样式 Unicode 编码
119:4	URI 的 Unicode 编码中包含未按百分比编码的字节
119:6	URI 采用两字节或三字节 UTF-8 编码
119:7	URI 使用 unicode 映射代码点编码
119:8	URI 路径包含连续的斜杠字符
119:9	URI 的路径部分中出现的反斜杠字符
119:10	URI 路径中包含/./ 模式的重复当前目录
119:11	URI 路径包含 /../ 模式向上移动的目录
119:12	HTTP 起始行中的制表符
119:13	HTTP 起始行或报头行以 LF 终止，无 CR
119:14	规范化 URI 包含 bad_characters 列表中的字符
119:15	URI 路径包含的网段长度超过 oversize_dir_length 参数
119:16	数据块长度超过配置的 maximum_chunk_length
119:18	URI 路径包含 /../，位于根目录之上
119:19	HTTP 标头行超过 4096 字节
119:20	HTTP 消息具有 200 多个信头字段
119:21	HTTP 消息包含多个 Content-Length 信头值
119:24	主机信头字段出现多次或具有多个值
119:25	HTTP 主机信头字段值的长度超过 maximum_host_length 选项

GID:SID	Rule Message
119:28	不带内容长度或数据块的 HTTP POST 或 PUT 请求
119:31	Snort 未知 HTTP 请求方法
119:32	HTTP 请求使用称为 HTTP/0.9 的原始 HTTP 格式
119:33	HTTP 请求 URI 包含未进行百分比编码的空格字符
119:34	HTTP 连接有超过 100 个并发管道请求尚未得到应答
119:102	无效的 HTTP 响应状态代码
119:104	HTTP 响应包含规范化失败的 UTF 字符集
119:105	HTTP 响应采用 UTF-7 字符集
119:109	不止一个级别的 JavaScript 模糊处理
119:110	连续的 JavaScript 空格数超过允许的最大值
119:111	JavaScript 混淆数据中的多种编码
119:112	SWF 文件 zlib 解压缩失败
119:113	SWF 文件 LZMA 解压缩失败
119:114	deflate 文件解压缩失败
119:115	PDF 文件的压缩类型不受支持
119:116	应用了多种压缩的 PDF 文件
119:117	PDF 文件解析失败
119:201	不是 HTTP 流量或无法恢复的 HTTP 协议错误
119:202	数据块长度包含过多前导零
119:203	HTTP 消息前或消息之间的空格
119:204	不带 URI 的请求消息
119:205	HTTP 响应原因短语中的控制字符
119:206	起始行中存在非法的额外空格
119:207	损坏的 HTTP 版本
119:209	HTTP 信头中的格式错误
119:210	存在数据块信头选项

GID:SID	Rule Message
119:211	URI 格式错误
119:212	URI 中的百分比编码无法识别
119:213	HTTP 数据块格式不正确
119:214	与数据块长度相邻的空格
119:215	信头名称中存在空格
119:216	过度 gzip 压缩
119:217	gzip 解压缩失败
119:218	另一个请求后跟 HTTP 0.9 的请求
119:219	正常请求之后的 HTTP 0.9 请求
119:220	消息同时具有 Content-Length 和 Transfer-Encoding
119:221	表示无正文与 Transfer-Encoding 或非零 Content-Length 组合的状态代码
119:222	传输编码不以分块结束
119:223	使用分块前的编码进行传输-编码
119:224	格式错误的 HTTP 流量
119:225	使用了不受支持的内容编码
119:226	使用的内容编码未知
119:227	应用了多个 Content-Encoding
119:228	服务器在客户端请求之前响应
119:229	PDF/SWF/ZIP 解压缩的服务器响应过大
119:230	HTTP 消息信头名称中存在非打印字符
119:231	HTTP 信头中的 Content-Length 值错误
119:232	HTTP 信头行换行
119:233	HTTP 信头行以 CR 终止，无 LF
119:234	以非标准分隔符终止的数据块
119:235	以 LF 终止，无 CR 的数据块长度
119:236	状态代码为 100 的多个响应

GID:SID	Rule Message
119:237	100 状态代码未响应预期信头
119:238	1XX 不是 100 或 101 的状态代码
119:239	已发送无消息正文的预期信头
119:240	具有传输编码信头的 HTTP 1.0 消息
119:241	用作 HTTP 信头的 Content-Transfer-Encoding
119:242	分块的消息报尾中的非法字段
119:243	信头字段不适当地出现两次或具有两个值
119:244	内容编码信头中的分块值无效
119:245	206 响应已发送到没有范围信头的请求
119:246	版本字段中的 HTTP 并非全部大写
119:247	关键信头值中嵌入了空格
119:248	gzip 压缩数据后跟意外的非 gzip 数据
119:249	HTTP 参数密钥重复次数过多
119:253	包含消息正文的 HTTP CONNECT 请求
119:254	在 CONNECT 请求之后但在 CONNECT 响应之前的 HTTP 客户端到服务器流量
119:255	带有 Content-Length 信头的 HTTP CONNECT 2XX 响应
119:256	具有 Transfer-Encoding 信头的 HTTP CONNECT 2XX 响应
119:257	带有 1XX 状态代码的 HTTP CONNECT 响应
119:258	请求消息完成之前的 HTTP CONNECT 响应
119:259	HTTP Content-Disposition 文件名参数格式错误
119:260	HTTP Content-Length 消息正文被截断
119:261	HTTP 分块消息正文被截断
119:262	HTTP URI 方案超过 10 个字符
119:263	HTTP/1 客户端请求的 HTTP/2 升级
119:264	授予 HTTP/1 服务器 HTTP/2 升级权限

GID:SID	Rule Message
119:265	JavaScript 中的错误令牌
119:266	JavaScript 中出现意外的脚本开始标记
119:267	JavaScript 中出现意外的脚本结束标记
119:268	外部脚本标记下的 JavaScript 代码
119:269	简短形式的脚本开始标记
119:270	唯一 JavaScript 标识符的最大数量
119:271	JavaScript 方括号嵌套超出容量
119:272	HTTP Accept-Encoding 信头中的连续逗号
119:273	JavaScript 规范化期间遗漏的 PDU
119:274	JavaScript 范围嵌套已超出容量
119:275	除 1.0 或 1.1e 以外的 HTTP/1 版本
119:276	起始行中的 HTTP 版本为 0
119:277	起始行中的 HTTP 版本高于 1

HTTP 检查检查器入侵规则选项

http_client_body

将检测光标设置到 HTTP 请求的正文时。当 HTTP 消息未指定 HTTP 标头时，Snort 会使用 URI 规范化对 `http_client_body` 进行规范化。URI 规范化通常应用于 `http_header`。

语法： `http_client_body;`

示例： `http_client_body;`

http_cookie

将检测光标设置为提取的 HTTP `Cookie` 信头字段。 `http_cookie` 规则选项包括以下参数：`http_cookie.request`、 `http_cookie.with_header`、 `http_cookie.with_body`和 `http_cookie.with_trailer`。

语法： http_cookie:<parameter> 、<parameter>

示例： `http_cookie: request;`

http_cookie.request

匹配在 HTTP 请求消息中找到的 HTTP cookie。检查 HTTP 响应时，请使用 HTTP 请求 cookie。`http_cookie.request` 参数是可选参数。

语法： `http_cookie:request;`

示例： `http_cookie: request;`

http_cookie.with_header

指定规则只能检查 HTTP 消息信头。`http_cookie.with_header` 参数是可选的。

语法： `http_cookie:with_header;`

示例： `http_cookie:with_header;`

http_cookie.with_body

指定规则的其他部分而不是 `http_cookie` 规则选项检查 HTTP 消息正文。 `http_cookie.with_body` 参数是可选的。

语法： `http_cookie:with_body;`

示例： `http_cookie:with_body;`

http_cookie.with_trailer

指定由规则的其他部分而不是`http_cookie` 规则选项检查 HTTP 消息报尾。`http_cookie.with_trailer` 参数是可选的。

语法： `http_cookie:with_trailer;`

示例： `http_cookie:with_trailer;`

http_header

将检测光标设置为规范化 HTTP 信头。您可以使用 `Field` 选项指定单个信头名称。

`http_header` 规则选项包括以下参数： `http_header.field`、 `http_header.request`、`http_header.with_header`、 `http_header.with_body`和 `http_header.with_trailer`。

语法： `http_header: field<field_name> 、<parameter> 、<parameter>`

示例： `http_header: Field Content-Type, with_trailer;`

http_header.field

将指定信头名称与规范化 HTTP 信头进行匹配。信头名称不区分大小写。如果不指定信头名称，HTTP 检查器将检查除 HTTP Cookie 信头（`Cookie` 和 `Set-Cookie`）之外的所有信头。

类型： 字符串

语法： `http_header: field <field_namer>;`

有效值： HTTP 信头名称。

示例： `http_header: Field Content-Type;`

http_header.request

匹配 HTTP 请求中找到的信头。检查 HTTP 响应时使用 HTTP 请求信头。`http_header.request` 参数是可选参数。

语法： `http_header: request;`

示例： `http_header: request;`

http_header.with_header

指定规则只能检查 HTTP 消息信头。 `http_header.with_header` 参数是可选的。

语法： `http_header:with_header;`

示例： `http_header: with_header;`

http_header.with_body

指定规则的其他部分而不是 `http_header` 规则选项检查 HTTP 消息正文。 `http_header.with_body` 参数是可选的。

语法： `http_header:with_body;`

示例： `http_header:with_body;`

http_header.with_trailer

指定由规则的另一部分而不是 `http_header` 规则选项检查 HTTP 消息报尾。`http_header.with_trailer` 参数是可选的。

语法： `http_header:with_trailer;`

示例： `http_header: with_trailer;`

http_method

将检测光标设置为 HTTP 请求的方法。常见的 HTTP 请求方法值为 `GET`、 `POST`、 `OPTIONS`、 `HEAD`、 `DELETE`、 `PUT`、 `TRACE`和 `CONNECT`。

`http_method` 规则选项包括参数： `http_method.with_header`、 `http_method.with_body`和 `http_method.with_trailer`。

语法： `http_method: <parameter> 、<parameter>;`

示例： `http_method; content:"GET";`

http_method.with_header

指定规则只能检查 HTTP 消息信头。 `http_method.with_header` 参数是可选的。

语法： `http_method:with_header;`

示例： `http_method:with_header;`

http_method.with_body

指定规则的其他部分而不是 http_header 规则选项检查 HTTP 消息正文。http_method.with_body 参数是可选的。

语法： http_method:with_body;

示例： http_method:with_body;

http_method.with_trailer

指定由规则的另一部分而不是http_header 规则选项检查HTTP 消息报尾。 http_method.with_trailer 参数是可选的。

语法： http_method:with_trailer;

示例： http_method:with_trailer;

http_param

将检测光标设置为指定的 HTTP 参数键。HTTP 参数密钥可能会显示在查询或请求正文中。

http_param 规则选项包括参数： http_param.param 和 http_method.no Case。

语法： http_param:<parameter_key> 、no Case;

示例： http_param: offset, noase;

http_param.param

匹配指定的参数。

类型： 字符串

语法： http_param:<http_parameter>;

有效值： 请求查询参数或请求正文字段。

示例： http_param: offset;

http_param.nocase

匹配指定参数，但不考虑大小写。http_param.nocase 参数是可选的。

语法： http_param: no Case;

示例： http_param: noase;

http_raw_body

将检测光标设置为未规范化请求或响应消息正文。

语法： http_raw_body;

示例： http_raw_body;

http_raw_cookie

将检测光标设置为非规范化 HTTP `Cookie` 信头。 `http_raw_cookie` 规则选项包括以下参数：`http_raw_cookie.request`、 `http_raw_cookie.with_header`、 `http_raw_cookie.with_body`和 `http_raw_cookie.with_trailer`。

语法： `http_raw_cookie: <parameter>;`

示例： `http_raw_cookie: request;`

http_raw_cookie.request

匹配在 HTTP 请求中找到的 cookie。检查响应消息时，请使用 HTTP 请求 cookie。`http_raw_cookie.request` 参数是可选参数。

语法： `http_raw_cookie: request;`

示例： `http_raw_cookie: request;`

http_raw_cookie.with_header

指定规则只能检查 HTTP 消息信头。 `http_raw_cookie.with_header` 参数是可选的。

语法： `http_raw_cookie:with_header;`

示例： `http_raw_cookie:with_header;`

http_raw_cookie.with_body

指定规则的其他部分而不是 `http_raw_cookie` 规则选项检查 HTTP 消息正文。`http_raw_cookie.with_body` 参数是可选的。

语法： `http_raw_cookie:with_body;`

示例： `http_raw_cookie:with_body;`

http_raw_cookie.with_trailer

指定由规则的另一部分而不是 `http_raw_cookie` 规则选项检查 HTTP 消息报尾。`http_raw_cookie.with_trailer` 参数是可选的。

语法： `http_raw_cookie:with_trailer;`

示例： `http_raw_cookie:with_trailer;`

http_raw_header

将检测光标设置为非规范化信头。 `http_raw_header` 包含原始消息中所有未经修改的信头名称和值。

`http_raw_header` 规则选项包括以下参数： `http_raw_header.field`、 `http_raw_header.request`、`http_raw_header.with_header`、 `http_raw_header.with_body`和 `http_raw_header.with_trailer`。

语法： `http_raw_header:field<field_name> 、<parameter> 、<parameter>;`

示例： `http_raw_header: Field Content-Type, with_trailer;`

http_raw_header.field

将指定信头名称与非规范化 HTTP 信头进行匹配。信头名称不区分大小写。如果不指定信头名称，HTTP 检查器将检查除 HTTP Cookie 信头（Cookie 和 Set-Cookie）以外的所有信头。

类型： 字符串

语法： `http_raw_header: field <field_name>`

有效值： HTTP 信头名称。

示例： `http_raw_header: Field Content-Type;`

http_raw_header.request

匹配 HTTP 请求消息中的找到的信头。检查响应消息时，请使用 HTTP 请求信头。`http_raw_header.request` 参数是可选参数。

语法： `http_raw_header: request;`

示例： `http_raw_header: request;`

http_raw_header.with_header

指定规则只能检查 HTTP 消息信头。`http_raw_header.with_header` 参数是可选的。

语法： `http_raw_header:with_header;`

示例： `http_raw_header: with_header;`

http_raw_header.with_body

指定规则的其他部分而不是 `http_raw_header` 规则选项检查 HTTP 消息正文。`http_raw_header.with_body` 参数是可选的。

语法： `http_raw_header:with_body;`

示例： `http_raw_header:with_body;`

http_raw_header.with_trailer

指定由规则的另一部分而不是 `http_raw_header` 规则选项检查 HTTP 消息报尾。`http_raw_header.with_trailer` 参数是可选的。

语法： `http_raw_header:with_trailer;`

示例： `http_raw_header: with_trailer;`

http_raw_request

将检测光标设置为非规范化请求行。要检查第一个信头行的特定部分，请使用以下规则选项之一：`http_method`、 `http_raw_uri`或 `http_version`。

`http_raw_request` 规则选项包括参数： `http_raw_request.with_header`、 `http_raw_request.with_body` 和 `http_raw_request.with_trailer`。

语法： `http_raw_request: <parameter>, <parameter>;`

示例： `http_raw_request:with_header;`

http_raw_request.with_header

指定规则只能检查 HTTP 消息信头。`http_raw_request.with_header` 参数是可选的。

语法： `http_raw_request:with_header;`

示例： `http_raw_request:with_header;`

http_raw_request.with_body

指定规则的其他部分而不是 `http_raw_request` 规则选项检查 HTTP 消息正文。`http_raw_request.with_body` 参数是可选的。

语法： `http_raw_request:with_body;`

示例： `http_raw_request:with_body;`

http_raw_request.with_trailer

指定由规则的其他部分而不是 `http_raw_request` 规则选项检查 HTTP 消息报尾。`http_raw_request.with_trailer` 参数是可选的。

语法： `http_raw_request:with_trailer;`

示例： `http_raw_request: with_trailer;`

http_raw_status

将检测光标设置为非规范化状态行。要检查状态行的特定部分，请使用以下规则选项之一：`http_version`、`http_stat_code`或 `http_stat_msg`。

`http_raw_status` 规则选项包括以下参数：`http_raw_status.with_body` 和 `http_raw_status.with_trailer`。

语法： `http_raw_status: <parameter>, <parameter>;`

示例： `http_raw_status:with_body;`

http_raw_status.with_body

指定由规则的其他部分而不是 `http_raw_status` 规则选项检查 HTTP 消息正文。`http_raw_status.with_body` 参数是可选的。

语法： `http_raw_status:with_body;`

示例： `http_raw_status:with_body;`

http_raw_status.with_trailer

指定由规则的另一部分而不是 `http_raw_status` 规则选项检查 HTTP 消息报尾。`http_raw_status.with_trailer` 参数是可选的。

语法： `http_raw_status:with_trailer;`

示例： `http_raw_status: with_trailer;`

http_raw_trailer

将检测光标设置为非规范化 HTTP 报尾。报尾包含有关消息内容的信息。当客户端请求创建 HTTP 报头时，报尾不可用。

`http_raw_trailer` 与 `http_raw_header`相同，但前者适用于结束信头。您必须创建单独的规则来检测 HTTP 信头和报尾。

`http_raw_trailer` 规则选项包括以下参数： `http_raw_trailer.field`、 `http_raw_trailer.request`、 `http_raw_trailer.with_header`、 `http_raw_trailer.with_body`。

语法： `http_raw_trailer: field <field_name>, <parameter>, <parameter>;`

示例： `http_raw_trailer: field <field_name>, request;`

http_raw_trailer.field

将指定报尾名称与非规范化 HTTP 报尾进行匹配。报尾名称不区分大小写。

类型： 字符串

语法： `http_raw_trailer: field <field_name>;`

有效值： HTTP 报尾名称。

示例： `http_raw_trailer:field trail-timestamp;`

http_raw_trailer.request

匹配 HTTP 请求消息中找到的报尾。在检查响应消息时，请使用 HTTP 请求报尾。 `http_raw_trailer.request` 参数是可选参数。

语法： `http_raw_trailer: request;`

示例： `http_raw_trailer: request;`

http_raw_trailer.with_header

指定规则只能检查 HTTP 响应信头。 `http_raw_trailer.with_header` 参数是可选的。

语法： `http_raw_trailer:with_header;`

示例： `http_raw_trailer: with_header;`

http_raw_trailer.with_body

指定由规则的其他部分（而不是 `http_raw_trailer` 规则选项）检查 HTTP 响应消息正文。 `http_raw_trailer.with_body` 参数是可选的。

语法： `http_raw_trailer:with_body;`

示例： `http_raw_trailer:with_body;`

http_raw_uri

将检测光标设置为非规范化 URI。

`http_raw_uri` 规则选项包括：

- `http_raw_uri.with_header`
- `http_raw_uri.with_body`
- `http_raw_uri.with_trailer`
- `http_raw_uri.scheme`
- `http_raw_uri.host`
- `http_raw_uri.port`
- `http_raw_uri.path`
- `http_raw_uri.query`
- `http_raw_uri.fragment`

语法： `http_raw_uri: <parameter>, <parameter>;`

示例： `http_raw_uri: with_header, path, query;`

http_raw_uri.with_header

指定规则只能检查 HTTP 消息信头。 `http_raw_uri.with_header` 参数是可选的。

语法： `http_raw_uri:with_header;`

示例： `http_raw_uri:with_header;`

http_raw_uri.with_body

指定规则的其他部分检查 HTTP 消息正文，而不是由 `http_raw_uri` 规则选项检查。 `http_raw_uri.with_body` 参数是可选的。

语法： `http_raw_uri:with_body;`

示例： `http_raw_uri:with_body;`

http_raw_uri.with_trailer

指定由规则的另一部分而不是 `http_raw_uri` 规则选项检查 HTTP 消息报尾。 `http_raw_uri.with_trailer` 参数是可选的。

语法： `http_raw_uri:with_trailer;`

示例： `http_raw_uri:with_trailer;`

http_raw_uri.scheme

仅根据 URI 方案进行匹配。 `http_raw_uri.scheme` 参数是可选的。

语法： `http_raw_uri:scheme;`

示例： `http_raw_uri:scheme;`

http_raw_uri.host

仅根据 URI 的主机（域名）匹配。`http_raw_uri.host` 参数是可选的。

语法： `http_raw_uri: host;`

示例： `http_raw_uri: host;`

http_raw_uri.port

仅根据 URI 的端口（TCP 端口）匹配。`http_raw_uri.port` 参数是可选的。

语法： `http_raw_uri: port;`

示例： `http_raw_uri: port;`

http_raw_uri.path

仅匹配 URI 的路径部分（目录和文件）。`http_raw_uri.path` 参数是可选的。

语法： `http_raw_uri: path;`

示例： `http_raw_uri: path;`

http_raw_uri.query

仅匹配 URI 中的查询参数。`http_raw_uri.query` 参数是可选的。

语法： `http_raw_uri: query;`

示例： `http_raw_uri: query;`

http_raw_uri.fragment

仅匹配 URI 的分片部分。分段是所请求文件的一部分，通常仅在浏览器内找到，而不通过网络进行传输。`http_raw_uri.fragment` 参数是可选的。

语法： `http_raw_uri: fragment;`

示例： `http_raw_uri: fragment;`

http_stat_code

将检测光标设置为 HTTP 状态代码。HTTP 状态代码是一个介于 100 - 599 之间的三位数字。

`http_stat_code` 规则选项包括以下参数：`http_stat_code.with_body` 和 `http_stat_code.with_trailer`。

语法： `http_stat_code: <parameter>, <parameter>;`

示例： `http_stat_code: with_trailer;`

http_stat_code.with_body

指定由规则的其他部分而不是 http_stat_code 规则选项检查 HTTP 消息正文。http_stat_code.with_body 参数是可选的。

语法： http_stat_code:with_body;

示例： http_stat_code:with_body;

http_stat_code.with_trailer

指定由规则的另一部分而不是 http_stat_code 规则选项检查 HTTP 消息报尾。http_stat_code.with_trailer 参数是可选的。

语法： http_stat_code:with_trailer;

示例： http_stat_code: with_trailer;

http_stat_msg

将检测光标设置至 HTTP 状态消息。HTTP 状态消息以纯文本描述 HTTP 状态代码，例如： 正常。

http_stat_msg 规则选项包括参数： http_stat_msg.with_body 和 http_stat_msg.with_trailer。

语法： http_stat_msg: <parameter>, <parameter>;

示例： http_stat_msg: with_body;

http_stat_msg.with_body

指定由规则的其他部分而不是 http_stat_msg 规则选项检查 HTTP 消息正文。 http_stat_msg.with_body 参数是可选的。

语法： http_stat_msg:with_body;

示例： http_stat_msg: with_body;

http_stat_msg.with_trailer

指定由规则的另一部分而不是 http_stat_msg 规则选项检查 HTTP 消息报尾。http_stat_msg.with_trailer 参数是可选的。

语法： http_stat_msg: with_trailer;

示例： http_stat_msg: with_trailer;

http_trailer

将检测光标设置为规范化报尾。报尾包含有关消息内容的信息。当客户端请求创建 HTTP 报头时，报尾不可用。

http_trailer 与 http_header相同，不同之处在于 http_trailer 适用于结束信头。您必须创建单独的规则来检测 HTTP 信头和报尾。

http_trailer 规则选项包括以下参数： http_trailer.field、 http_trailer.request、 http_trailer.with_header、 http_trailer.with_body。

语法： `http_trailer: field <field_name>, <parameter>, <parameter>;`

示例： `http_trailer: Field trail-timestamp, with_body;`

http_trailer.field

将指定报尾名称与规范化 HTTP 报尾进行匹配。报尾名称不区分大小写。

类型： 字符串

语法： `http_trailer: field <field_name>;`

有效值： HTTP 报尾名称。

示例： `http_trailer:field trail-timestamp;`

http_trailer.request

匹配 HTTP 请求消息中找到的报尾。在检查响应消息时，请使用 HTTP 请求报尾。`http_trailer.request` 参数是可选参数。

语法： `http_trailer: request;`

示例： `http_trailer: request;`

http_trailer.with_header

指定由规则的其他部分检查 HTTP 消息信头，而不是由 `http_trailer` 规则选项检查。`http_trailer.with_header` 参数是可选的。

语法： `http_trailer:with_header;`

示例： `http_trailer: with_header;`

http_trailer.with_body

指定规则的其他部分而不是 `http_trailer` 规则选项检查 HTTP 消息正文。 `http_trailer.with_body` 参数是可选的。

语法： `http_trailer:with_body;`

示例： `http_trailer:with_body;`

http_true_ip

将检测光标设置为最终客户端 IP 地址。当客户端发送请求时，代理服务器将存储最终的客户端 IP 地址。客户端 IP 地址是 `X-Forwarded-For`、 `True-Client-IP`或任何其他自定义 `X-Forwarded-For` 类型报头中列出的最后一个 IP 地址。如果存在多个信头，Snort 会考虑 `xff_headers`中定义的信头。

`http_true_ip` 规则选项包括以下参数： `http_true_ip.with_header`、 `http_true_ip.with_body`和 `http_true_ip.with_trailer`。

语法： `http_true_ip: <parameter>, <parameter>;`

示例： `http_true_ip:with_header;`

http_true_ip.with_header

指定规则只能检查 HTTP 消息信头。 `http_true_ip.with_header` 参数是可选的。

语法： `http_true_ip:with_header;`

示例： `http_true_ip:with_header;`

http_true_ip.with_body

指定规则的其他部分而不是 `http_true_ip` 规则选项检查 HTTP 消息正文。 `http_true_ip.with_body` 参数是可选的。

语法： `http_true_ip:with_body;`

示例： `http_true_ip:with_body;`

http_true_ip.with_trailer

指定由规则的另一部分而不是 `http_true_ip` 规则选项检查 HTTP 消息报尾。 `http_true_ip.with_trailer` 参数是可选的。

语法： `http_true_ip:with_trailer;`

示例： `http_true_ip:with_trailer;`

http_uri

将检测光标设置为规范化 URI 缓冲区。

- `http_uri.with_header`
- `http_uri.with_body`
- `http_uri.with_trailer`
- `http_uri.scheme`
- `http_uri.host`
- `http_uri.port`
- `http_uri.path`
- `http_uri.query`
- `http_uri.fragment`

语法： `http_uri: <parameter>, <parameter>;`

示例： `http_uri:with_trailer,path,query;`

http_uri.with_header

指定规则只能检查 HTTP 消息信头。 `http_uri.with_header` 参数是可选的。

语法： `http_uri:with_header;`

示例： `http_uri:with_header;`

http_uri.with_body

指定规则的其他部分检查 HTTP 消息正文，而不是检查 `http_uri` 规则选项。 `http_uri.with_body` 参数是可选的。

语法： `http_uri:with_body;`

示例： `http_uri:with_body;`

http_uri.with_trailer

指定由规则的另一部分而不是 `http_uri` 规则选项检查 HTTP 消息报尾。 `http_uri.with_trailer` 参数是可选的。

语法： `http_uri:with_trailer;`

示例： `http_uri:with_trailer;`

http_uri.scheme

仅根据 URI 方案进行匹配。 `http_uri.scheme` 参数是可选的。

语法： `http_uri:scheme;`

示例： `http_uri:scheme;`

http_uri.host

仅根据 URI 的主机（域名）匹配。 `http_uri.host` 参数是可选的。

语法： `http_uri: host;`

示例： `http_uri: host;`

http_uri.port

仅根据 URI 的端口（TCP 端口）匹配。 `http_uri.port` 参数是可选的。

语法： `http_uri: port;`

示例： `http_uri: port;`

http_uri.path

仅匹配 URI 的路径（目录和文件）。 `http_uri.path` 参数是可选的。

语法： `http_uri: path;`

示例： `http_uri: path;`

http_uri.query

仅匹配 URI 中的查询参数。`http_uri.query` 参数是可选的。

语法： `http_uri: uri;`

示例： `http_uri: query;`

http_uri.fragment

仅匹配 URI 的分片部分。分段是所请求文件的一部分，通常仅在浏览器内找到，而不通过网络进行传输。`http_uri.fragment` 参数是可选的。

语法： `http_uri: fragment;`

示例： `http_uri: fragment;`

http_version

将检测光标设置为 HTTP 版本缓冲区的开头。`http_version` 接受各种 HTTP 版本。最常见的版本为：`HTTP/1.0` 和 `HTTP/1.1`。`http_version` 规则选项包括参数：`http_version.request`、`http_version.with_header`、`http_version.with_body`和 `http_version.with_trailer`。

语法： `http_version: <parameter>, <parameter>;`

示例： `http_version; content:"HTTP/1.1";`

http_version.request

与 HTTP 请求中的版本相匹配。检查响应消息时，请使用请求版本。`http_version.request` 参数是可选参数。

语法： `http_version: request;`

示例： `http_version: request;`

http_version.with_header

指定规则只能检查 HTTP 消息信头。`http_version.with_header` 参数是可选的。

语法： `http_version:with_header;`

示例： `http_version:with_header;`

http_version.with_body

指定规则的其他部分检查 HTTP 消息正文，而不是由 `http_version` 规则选项检查。`http_version.with_body` 参数是可选的。

语法： `http_version:with_body;`

示例： `http_version:with_body;`

http_version.with_trailer

指定由规则的另一部分检查 HTTP 消息报尾，而不是由 `http_version` 规则选项检查。`http_version.with_trailer` 参数是可选的。

语法： `http_version:with_trailer;`

示例： `http_version:with_trailer;`

http_version_match

指定要针对标准 HTTP 版本进行匹配的 HTTP 版本列表。使用空格字符分隔多个版本。HTTP 请求或状态行可能包含版本。如果版本存在，Snort 会将此版本与 `http_version_match`中指定的列表进行比较。

如果版本不是 `[0-9].[0-9]` 格式，则会被视为格式错误。 采用 `[0-9].[0-9]` 格式（非 `1.0` 或 `1.1`）的版本被视为 其他。

类型： 字符串

语法： `http_version_match: <version_list>`

有效值： `1.0`、`1.1`、`2.0`、`0.9`、其他、格式错误

示例： `http_version_match: "1.0 1.1";`

js_data

将检测光标设置为规范化的 JavaScript 数据。此选项特定于增强型 JavaScript 规范器。

语法： `js_data;`

示例： `js_data;`

vba_data

将检测光标设置为 Microsoft Office Visual Basic for Applications 宏缓冲区。

语法： `vba_data;`

示例： `vba_data;`

第 12 章

IEC104 检查器

IEC104 检查器概述

类型	检查器（服务）
使用方式	检测
实例类型	多实例
所需其他检查器	`stream_tcp`
已启用	`false`

IEC 60870-5-104 (IEC104) 协议描述了一种在电力系统之间交换远程控制消息的通信标准。IEC104 协议使用 TCP 端口 2404。

`iec104` 检查器会检测网络流量中的 IEC104 消息。 `iec104` 检查器通过组合分布在多个框架中的消息或拆分一个框架内的多条消息来分析和规范化 IEC104 消息。

启用后，入侵规则选项将提供对 IEC104 应用协议控制信息 (APCI) 类型和应用服务数据单元 (ASDU) 功能代码的访问。

IEC104 检查器参数

IEC104 TCP 端口配置

绑定程序 检查器定义 IEC104 TCP 端口配置。有关详细信息，请参阅绑定程序检查器概述， 第 13 页。

示例：

```
[
    {
        "when": {
            "role": "server",
            "proto": "tcp",
            "ports": "2404"
         },
        "use": {
            "type": "iec104"
        }
    }
]
```

注释 `iec104` 检查器不提供任何参数。

IEC104 检查器规则

启用 `iec104` 检查器规则以 生成事件并在内联部署中丢弃攻击性数据包。

表 *15: IEC104* 检查器规则

GID:SID	Rule Message
151:1	IEC104 APCI 信头中的长度与给定的 IEC104 ASDU 类型 ID 所需的长度不匹配
151:2	IEC104 起始字节与 0x68 不匹配
151:3	保留的 IEC104 ASDU 类型 ID 正在使用中
151:4	IEC104 APCI U 保留字段包含非默认值
151:5	IEC104 APCI U 消息类型设置为无效值
151:6	IEC104 APCI S 保留字段包含非默认值
151:7	IEC104 APCI I 元素数量设置为零
151:8	在不支持该功能的 ASDU 上设置了 IEC104 APCI I SQ 位
151:9	IEC104 APCI I 在不支持功能的 ASDU 上设置的元素数量大于一个
151:10	IEC104 APCI I 初始化原因设置为保留值
151:11	IEC104 APCI I 询问命令限定符设置为保留值
151:12	IEC104 APCI I Qualifier of Counter Interrogation Command 请求参数设置为保留值
151:13	设置为保留值的测量值参数类型的 IEC104 APCI I 参数限定符

GID:SID	Rule Message
151:14	测量值本地参数更改为 IEC104 APCI I 参数限定符设置为技术上有效但未使用的值
151:15	测量值参数的 IEC104 APCI I 限定符参数选项设置为技术上有效但未使用的值
151:16	参数激活的 IEC104 APCI I 限定符设置为保留值
151:17	命令的 IEC104 APCI I 限定符设置为保留值
151:18	重置过程的 IEC104 APCI I 限定符设置为保留值
151:19	IEC104 APCI I 文件就绪限定符设置为保留值
151:20	IEC104 APCI I 区域就绪限定符设置为保留值
151:21	IEC104 APCI I 选择和设置为保留值的呼叫限定符
151:22	IEC104 APCI I 最后的分段或网段限定符设置为保留值
151:23	IEC104 APCI I 确认文件或部分限定符设置为保留值
151:24	邮件上设置的 IEC104 APCI I 结构限定符原本不应产生任何影响
151:25	IEC104 APCI I 单点信息保留字段包含非默认值
151:26	IEC104 APCI I 双点信息保留字段包含非默认值
151:27	IEC104 APCI I 传输原因设置为保留值
151:28	设置为 ASDU 不允许的值的 IEC104 APCI I 传输原因设置的值
151:29	IEC104 APCI I 检测到两个八位组通用地址值无效
151:30	IEC104 APCI I 质量描述符结构保留字段包含非默认值
151:31	保护设备结构保留字段的 IEC104 APCI I 质量描述符包含非默认值
151:32	IEC104 APCI I IEEE STD 754 值导致 NaN
151:33	IEC104 APCI I IEEE STD 754 值导致无限大
151:34	IEC104 APCI I 保护设备结构单一事件保留字段包含非默认值
151:35	IEC104 APCI I 保护设备结构开始事件的保留字段包含非默认值
151:36	IEC104 APCI I 输出电路信息结构保留字段包含非默认值
151:37	检测到 IEC104 APCI I 异常固定测试位模式
151:38	IEC104 APCI I 单个命令结构保留字段包含非默认值

GID:SID	Rule Message
151:39	IEC104 APCI I 双命令结构包含无效值
151:40	IEC104 APCI I 管制步骤命令结构保留字段包含非默认值
151:41	设置的 IEC104 APCI I Time2a 毫秒超出允许范围
151:42	IEC104 APCI I Time2a 分钟设置超出允许范围
151:43	IEC104 APCI I Time2a “保留的分钟”字段包含非默认值
151:44	设置的 IEC104 APCI I Time2a 小时数超出允许的范围
151:45	IEC104 APCI I Time2a “保留小时”字段包含非默认值
151:46	IEC104 APCI I 设置的 Time2a 月份日期超出允许范围
151:47	设置的 IEC104 APCI I 月 Time2a 超出允许的范围
151:48	IEC104 APCI I Time2a “保留月份”字段包含非默认值
151:49	设置的 IEC104 APCI I Time2a 年份超出允许范围
151:50	IEC104 APCI I Time2a “保留年份”字段包含非默认值
151:51	IEC104 APCI I 检测到空的网段长度值
151:52	IEC104 APCI I 检测到无效的网段长度值
151:53	IEC104 APCI I 文件状态设置为保留值
151:54	IEC104 APCI I 设定值命令限定符设置为保留值

IEC104 Inspector 入侵规则选项

iec104_apci_type

验证 IEC104 消息是否与选项中设置的 IEC104 应用协议信息控制 (APIC) 类型匹配。

`iec104_apci_type` 入侵规则选项接受使用完整 APIC 类型名称或者大写或小写 APIC 类型缩写指定的字符串。

类型： 字符串

语法： `iec104_apci_type: <apic_type>;`

示例：

```
iec104_apci_type: unnumbered_control_function;
iec104_apci_type: S;
```

```
iec104_apci_type: I;
iec104_apci_type: i;
```

iec104_asdu_func

验证 IEC104 消息是否与选项中设置的 IEC104 应用服务数据单元 (ASDU) 功能代码相匹配。

`iec104_asdu_fuc` 入侵规则选项接受以大写或小写字母 ASDU 功能代码指定的字符串。

类型： 字符串

语法： `iec104_asdu_fuc: <asdu_func>;`

示例：

```
iec104_asdu_func: M_SP_NA_1;
iec104_asdu_func: m_sp_na_1;
```

第 13 章

IMAP 检查器

IMAP 检查器概述

类型	检查器（服务）
使用方式	检测
实例类型	多实例
所需其他检查器	`stream_tcp`
已启用	`true`

互联网邮件应用协议 (IMAP) 使电子邮件客户端能够从远程 IMAP3 服务器检索邮件。IMAP3 服务器将 TCP 端口 143 用于不安全的会话，或将 TCP 端口 993 用于基于 SSL/TLS 的 IMAP。

`imap` 检查器检测 IMAP 流量并分析 IMAP 命令和响应。

`imap` 检查器可以识别 IMAP 邮件的命令、信头和正文部分，并提取和解码多用途互联网邮件扩展 (MIME) 附件。MIME 附件可能包括多个附件和跨越多个数据包的大型附件。

`imap` 检查器可识别 IMAP 流量并将其添加到 Snort 允许列表。启用后，入侵规则会针对异常 IMAP 流量生成事件。

IMAP 检查器参数

IMAP 服务配置

`绑定程序` 检查器定义 IMAP `服务` 配置。有关详细信息，请参阅绑定程序检查器概述，第 13 页。

示例：

```
[
    {
        "when": {
            "service": "imap",
            "role": any
        },
        "use": {
            "type": "imap"
        }
    }
]
```

b_64_decode_depth

指定要从每个 Base64 编码的 MIME 邮件附件中提取和解码的最大字节数。可以指定小于 65535 的整数，或指定 0 以禁用解码。指定 -1 以对要解码的字节数不设限制。

您可以启用规则 141:4 以生成此参数的事件，并且在内联部署中，当解码失败（由于编码不正确或数据损坏）时，丢弃违规的数据包。

类型： 整数

有效范围： `-1` 至 `65535`

默认值： `-1`

bitenc_decode_depth

指定要从每个非编码的 MIME 邮件附件中提取的最大字节数。可以指定一个小于 65535 的整数，或指定 0 以禁用未编码 MIME 附件的提取。指定 -1 对要提取的字节数不设限制。这些附件类型包括 7 位、8 位、二进制以及各种多部分内容类型（例如，纯文本、JPEG 和 PNG 图像、MP4 文件等）。

类型： 整数

有效范围： `-1` 至 `65535`

默认值： `-1`

decompress_pdf

指定是否解压缩 MIME 附件中的 `application/pdf` (PDF) 文件。

您可以启用规则 141:8 以生成此参数的事件，并在内联部署中，丢弃违规数据包。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

decompress_swf

指定是否解压缩 MIME 附件中的 `application/vnd.adobe.flash-movie` (SWF) 文件。

您可以启用规则 141:8 以生成此参数的事件，并在内联部署中，丢弃违规数据包。

类型： 整数

有效值： `true`、`false`

默认值： `false`

decompress_vba

指定是否解压缩 MIME 附件中的 Microsoft Office Visual Basic for Applications 宏文件。

类型： boolean

有效值： `true`、`false`

默认值： `false`

decompress_zip

指定是否解压缩 MIME 附件中的 `application/zip` (ZIP) 文件。

您可以启用规则 141:8 以生成此参数的事件，并在内联部署中，丢弃违规数据包。

类型： boolean

有效值： `true`、`false`

默认值： `false`

qp_decode_depth

指定要从每个 Quoted-Printable (QP) 编码的 MIME 邮件附件中提取和解码的最大字节数。可以指定小于 65535 的整数，或指定 0 以禁用解码。指定 -1 以对要解码的字节数不设限制。

您可以启用规则 141:5 以生成此参数的事件，并且在内联部署中，当解码失败（由于编码不正确或数据损坏）时，丢弃违规的数据包。

类型： 整数

有效范围： `-1` 至 `65535`

默认值： `-1`

uu_decode_depth

指定要从每个 Unix-to-Unix 编码（UuEncode 编码）的 MIME 邮件附件中提取和解码的最大字节数。可以指定小于 65535 的整数，或指定 0 以禁用解码。指定 -1 以对要解码的字节数不设限制。

您可以启用规则 141:7 以生成此参数的事件，并且在内联部署中，当解码失败（由于编码不正确或数据损坏）时，丢弃违规的数据包。

类型： 整数

有效范围： `-1` 至 `65535`

默认值： `-1`

IMAP 检查器规则

启用 imap 检查器规则以 生成事件并在内联部署中丢弃攻击性数据包。

表 *16: IMAP* 检查器规则

GID:SID	Rule Message
141:1	未知 IMAP3 命令
141:2	未知 IMAP3 响应
141:4	base64 解码失败
141:5	Quoted-Printable 解码失败
141:7	Unix-to-Unix 解码失败
141:8	文件解压缩失败

IMAP 检查器入侵规则选项

vba_data

将检测光标设置为 Microsoft Office Visual Basic for Applications 宏缓冲区。

语法： `vba_data;`

示例： `vba_data;`

第 14 章

MMS 检查器

MMS 检查器概述

类型	检查器（服务）
使用方式	检测
实例类型	多实例
所需其他检查器	`stream_tcp`
已启用	`false`

IEC 61850 是定义电力系统通信协议的国际标准。制造消息规范 (MMS) 协议是 IEC 61850 协议之一。MMS 支持在各种制造和过程控制设备之间实时传输监控和数据采集 (SCADA) 数据。MMS 协议使用 TCP 端口 102 在客户端和服务器设备之间交换消息。

`MMS` 检查器检测并分析 MMS 流量。MMS 消息可以在一个 TCP 数据包中包含多个协议数据单元 (PDU)，可以在多个 TCP 数据包中包含一个 PDU，也可以包含这两种消息配置。`MMS` 检查器规范化 MMS 流量，以向设备显示完整的 MMS 消息。

您可以为 MMS 消息编写 Snort 3 规则，而无需解码 MMS 协议。`MMS` 检查器会分析封装 MMS 协议的 OSI 层，并通过规则选项提供对某些 MMS 协议字段和数据内容的访问权限。有关 MMS 规则选项的信息，请参阅 MMS 检查器入侵规则选项， 第 120 页

MMS 检查器参数

MMS 服务配置

绑定程序 检查器定义 MMS 服务 配置。有关详细信息，请参阅绑定程序检查器概述 ， 第 13 页。

示例：

```
[
    {
        "when": {
            "service": "mms"
        },
        "use": {
            "type": "mms"
        }
    }
]
```

MMS 检查器规则

mms 检查器没有任何关联规则。

MMS 检查器入侵规则选项

mms_data

将检测光标位置设置到 MMS 协议数据单元 (PDU) 的开头，绕过所有 OSI 封装层。当入侵规则包括 mms_data时，规则中的后续规则选项从 MMS PDU 开始处理。

语法： `mms_data;`

示例：

以下入侵规则示例设置 mms_data 规则选项。 mms_data 规则选项将检测光标定位到 MMS PDU 的开头，并检查该位置的字节是否具有 发起-请求 消息的值。

```
alert tcp ( \
msg: "PROTOCOL-SCADA MMS Initiate-Request"; \
flow: to_server, established; \
mms_data; \
content:"|A8|", depth 1; \
sid:1000000; \
)
```

mms_func

将提供的函数名称或编号与 MMS 请求或响应中的 已确认服务 字段进行比较。当 MMS 功能名称或数量与 已确认的服务匹配时发出警报。

类型： 字符串

语法： mms_func <function>;

示例：

以下入侵规则示例设置 mms_fuc 规则选项，并在 已确认的服务请求 服务与提供的函数名称匹配时发出警报。此外， mms_fuc 启用快速模式匹配功能，以匹配 已确认的服务请求 (0xA0) 消息。

```
alert tcp ( \
msg: "PROTOCOL-SCADA MMS svc get_name_list"; \
flow: to_server, established; \
content:"|A0|"; \
mms_func: get_name_list; \
sid:1000000; \
)
```

以下入侵规则示例设置 mms_fuc 规则选项，并在 GetNameList 消息与功能编号匹配时发出警报。

```
alert tcp ( \
msg: "PROTOCOL-SCADA MMS svc get_name_list"; \
flow: to_server, established; \
content:"|A0|"; \
mms_func:1; \
sid:1000001; \
)
```

第 15 章

Modbus 检查器

Modbus 检查器概述

类型	检查器（服务）
使用方式	检测
实例类型	多实例
所需其他检查器	`stream_tcp`
已启用	`false`

Modbus 协议定义了一种通信标准，用于在监控和数据采集 (SCADA) 系统与可编程自动化控制器 (plc) 之间交换消息。Modbus 协议使用 TCP 端口 502。

`Modbus` 检查器检测并分析网络流量中的 Modbus 消息。启用 Modbus 入侵规则选项后，即可访问某些 Modbus 协议字段。

配置 Modbus 检查器的最佳实践

如果您的网络中没有开启的 Modbus 设备，请不要在应用于流量的网络分析策略中开启 `Modbus` 巡检器。

Modbus 检查器参数

Modbus TCP 端口配置

绑定程序 检查器定义 Modbus TCP 端口配置。有关详细信息，请参阅绑定程序检查器概述 ， 第 13 页。

示例：

```
[
    {
        "when": {
            "role": "server",
            "proto": "tcp",
            "ports": "502"
        },
        "use": {
            "type": "modbus"
        }
    },
    {
        "when": {
            "role": "any",
            "service:" "modbus"
        },
        "use": {
            "type":"modbus"
        }
    }
]
```

注释 Modbus 检查器不提供任何参数。

Modbus 检查器规则

启用 Modbus 检查器规则以 生成事件并在内联部署中丢弃攻击性数据包。

表 *17: Modbus* 检查器规则

GID:SID	Rule Message
144:1	Modbus MBAP 信头中的长度与给定功能所需的长度不匹配
144:2	Modbus 协议 ID 非零
144:3	保留的 Modbus 功能代码正在使用中

Modbus Inspector 入侵规则选项

您可以单独使用 Modbus 选项，也可以将其与 content 和 byte_jump 入侵规则选项结合使用。

modbus_data

将数据光标设置到 Modbus 数据 字段的开头。

语法： modbus_data;

示例： modbus_data;

modbus_func

验证 Modbus 功能 字段是否与指定的 Modbus 功能代码匹配。您可以设置一个正整数或字符串文字来表示 Modbus 功能代码。

类型： 字符串

语法： modbus_fuc: <function>中所述;

有效值：

表 *18: Modbus* 函数代码值

代码	字符串
1	read_coils
2	read_discrete_inputs
3	read_holding_registers
4	read_input_registers
5	write_single_coil
6	write_single_register
7	read_exception_status
8	diagnostics
11	get_comm_event_counter
12	get_comm_event_log
15	write_multiple_coils
16	write_multiple_registers
17	report_slave_id
20	read_file_record

代码	字符串
21	write_file_record
22	mask_write_register
23	read_write_multiple_registers
24	read_fifo_queue
43	encapsulated_interface_transport

示例：

```
modbus_func: read_coils;
modbus_func: 8;
```

modbus_unit

验证消息中的 Modbus 设备 ID 是否与指定的设备 ID 匹配。您可以设置一个数字来表示 Modbus 设备 ID。

类型： 整数

语法： `modbus_unit: <unit_id>;`

有效范围： 0 至 255

示例：

```
modbus_unit: 1;
```

第 16 章

规范器检查器

规范器检查器概述

类型	检查器（数据包）
使用方式	情景
实例类型	网络
所需其他检查器	无
已启用	`true`

规范器检查器检测并删除数据包中的协议异常。规范器检查器可以在内联部署中最大限度地减少攻击者创建数据包以规避检测的可能性。

注释 在从网络发送流量前，必须使用路由接口、交换接口或透明接口或者内联接口对向受管设备部署相关配置。

您可以指定数据包中的 IPv4、IPv6、ICMPv4、ICMPv6 和 TCP 协议的任意组合的规范化。规范器检查器执行每个数据包的规范化操作并处理大多数规范化操作。`stream_tcp` 检查器处理 TCP 状态相关的数据包和流规范化，包括 TCP 负载规范化。

在进行解码后会立即执行内联规范化，直至其他检查器进行处理。规范化从内数据包层继续执行到外数据包层。

规范器检查器不会生成事件。规范器检查器会准备数据包，以供其他检查器和在内联部署中使用。检查器有助于确保系统处理的数据包与网络中主机接收的数据包相同。

规范器检查器参数

找到配置中的 规范器 范围，以设置 规范器 检查器参数。

ip6

清除 IPv6 流量中的 Reserved 标志。

类型： boolean

有效值： true、 false

默认值： false

icmp4

清除 ICMPv4 流量中的 Reserved 标志。

类型： boolean

有效值： true、 false

默认值： false

icmp6

清除 ICMPv6 流量中的 Reserved 标志。

类型： boolean

有效值： true、 false

默认值： false

ip4.base

清除“IPv4 标志”信头字段及传送参数的一位 保留 子字段。修复紧急指针/标志问题。我们建议您启用 ip4.base。

类型： boolean

有效值： true、 false

默认值： false

ip4.df

清除“IPv4 标志”信头字段的一位 不分片 子字段。启用 ip4.df，以允许下游路由器对数据包进行分片，而不是丢弃它们。 ip4.df 参数可以防止造成数据包丢弃的规避。

类型： boolean

有效值： true、 false

默认值： false

ip4.rf

清除传入数据包上的 保留 位。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

ip4.tos

清除一个字节的 差分服务 字段（以前称为 服务类型）。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

ip4.trim

将具有多余负载的数据包截断至 IP 报头中指定的数据报长度加上第 2 层（例如以太网）报头，但是不截断为小于最小帧长度。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

tcp.base

清除 TCP 信头的单位 保留 子字段以及选项填充字节。修复紧急指针或标志问题。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

tcp.block

指定在 TCP 规范化期间是否丢弃数据包。

启用此选项时，Snort 阻止异常 TCP 数据包，这些数据包在规范化的情况下会无效，并可能受到接收主机的阻止。例如，Snort 阻止后续传输到已建立的会话上的任何 SYN 数据包。

无论是否启用规则，Snort 都会丢弃与以下任何 TCP 数据流检查器规则匹配的任何数据包：

- 129:1
- 129:3
- 129:4
- 129:6

- 129:8
- 129:11
- 129:14 至 129:19

类型： boolean

有效值： `true`、 `false`

默认值： `false`

tcp.ecn

对 显式堵塞通知 (ECN) 标志启用逐个数据包或逐条数据流规范化。

- 指定 数据包 以逐个数据包清除 ECN 标志（无论协商与否）。
- 选择 流 以逐条数据流清除 ECN 标志（如果未协商 ECN 的使用）。如果指定 流，则必须在 TCP 数据流检查器中启用 `tcp.require_3whs` ，才能进行规范化。
- 指定 `off` 以禁用 `tcp.ecn` 参数。

类型： enum

有效值： `off`、 `packet`、 `stream`

默认值： `off`

tcp.ips

启用“TCP 数据”(TCP Data) 字段的规范化以确保重传数据的一致性。无法正确重组的所有数据段都会被丢弃。

类型： boolean

有效值： `true`、 `false`

默认值： `true`

tcp.opts

指定是否规范化流量中允许的特定 TCP 选项。Snort 不对您明确允许的选项进行规范化。Snort 不对您明确允许的选项进行规范化。

Snort 总是允许以下 TCP 选项，因为它们通常用于优化 TCP 性能：

- 最大分片大小 (MSS)
- 窗口比例
- 时间戳 TCP

Snort 不会自动允许其他不太常用的选项。

启用 `tcp.opts` 后，TCP 流量规范化包括以下操作：

- 除 MSS、“窗口比例”、“时间戳”及任何明确允许的选项以外，所有选项字节都设置为“无操作”（TCP 选项 1）。
- 如果时间戳存在但无效，或者有效但未协商，则将时间戳八位元设置为“无操作”。
- 如果时间戳已协商但不存在，则阻止数据包
- 如果未设置 Acknowledgment (ACK) 控制位，则清除“时间戳回应答复 (TSecr)”选项字段。
- 如果未设置 SYN 控制位，则将 MSS 和 Window Scale 选项设置为“无操作” (TCP 选项 1)。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

tcp.pad

清除任何选项填充字节。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

tcp.req_pay

如果没有负载，则清除 TCP 信头 紧急指针 字段和紧急 URG 控制位。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

tcp.req_urg

如果未设置 TCP 信头紧急 (URG) 控制位，则清除 16 位 TCP 信头 紧急指针 字段。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

tcp.req_urp

如果未设置 TCP 信头紧急指针字段，则清除 TCP 信头紧急 (URG) 控制字段。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

tcp.resv

清除 TCP 信头中的 保留 位。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

tcp.trim_mss

如果负载长度大于 MSS，则将 TCP 数据 字段修剪为“最大分片大小”(MSS)。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

tcp.tim_rst

清除 RST 数据包中的数据。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

tcp.trun_syn

删除 TCP 同步 (SYN) 数据包中的数据。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

tcp.trim_win

将 TCP 数据 字段修剪为在 窗口 字段中指定的大小。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

tcp.urp

如果指针大于负载长度，则将两字节的 TCP 信头 紧急指针 字段设置为负载长度。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

规范器检查器规则

规范器 检查器没有任何关联的规则。

规范器检查器入侵规则选项

规范器 检查器没有任何入侵规则选项。

第 17 章

POP 检查器

POP 检查器概述

类型	检查器（服务）
使用方式	检测
实例类型	多实例
所需其他检查器	`stream_tcp`
已启用	`true`

邮局协议版本 3 (POP3) 使邮件客户端能够从远程 POP3 服务器检索邮件。POP3 服务器使用 TCP 端口 110 进行不安全的会话，或将 TCP 端口 995 用于基于 SSL/TLS 的 POP。

`POP` 检查器可检测 POP 流量并分析 POP 命令和响应。

`POP` 检查器可以识别 POP 邮件的命令、信头和正文部分，并提取和解码多用途互联网邮件扩展 (MIME) 附件。 `POP` 检查器处理 MIME 附件，包括多个附件和跨越多个数据包的大型附件。

`POP` 检查器识别 POP 消息并将其添加到 Snort 允许列表。启用后，入侵规则将针对异常 POP 流量生成事件。

POP 检查器参数

注释 如果 MIME 邮件附件不要求解码，解码或提取涵盖多个附件（如果有）以及存在于多个数据包中的大型附件。

当 `b_64_decode_detection`、 `bitencen_decode_length`、 `qp_decode_length`或 `uu_decode_length` 参数的值在以下情况下不同时，将使用最大值：

- 默认网络分析策略
- 由同一访问控制策略中的网络分析规则调用的任何其他自定义网络分析策略

POP 服务配置

`绑定程序` 检查器定义 POP `服务` 配置。有关详细信息，请参阅绑定程序检查器概述，第 13 页。

示例：

```
[
    {
        "when": {
           "service": "pop",
           "role": any
        },
        "use": {
           "type": "pop"
        }
    }
]
```

b_64_decode_depth

指定要从每个 Base64 编码的 MIME 邮件附件中提取和解码的最大字节数。可以指定小于 65535 的整数，或指定 0 以禁用解码。指定 -1 以对要解码的字节数不设限制。

您可以启用规则 142:4 以生成此参数的事件，并在内联部署中当解码失败时丢弃违规数据包。

类型： 整数

有效范围： `-1` 至 `65535`

默认值： `-1`

bitenc_decode_depth

指定要从每个非编码的 MIME 邮件附件中提取的最大字节数。可以指定一个小于 65535 的整数，或指定 0 以禁用未编码 MIME 附件的提取。指定 -1 对要提取的字节数不设限制。这些附件类型包括 7 位、8 位、二进制以及各种多部分内容类型（例如，纯文本、JPEG 和 PNG 图像、MP4 文件等）。

类型： 整数

有效范围： `-1` 至 `65535`

默认值： `-1`

decompress_pdf

指定是否解压缩 MIME 附件中的 `application/pdf` (PDF) 文件。

您可以启用规则 142:8 以生成此参数的事件，并在内联部署中，丢弃违规数据包。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

decompress_swf

指定是否解压缩 MIME 附件中的 `application/vnd.adobe.flash-movie` (SWF) 文件。

您可以启用规则 142:8 以生成此参数的事件，并在内联部署中，丢弃违规数据包。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

decompress_vba

指定是否解压缩 MIME 附件中的 Microsoft Office Visual Basic for Applications 宏文件。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

decompress_zip

指定是否解压缩 MIME 附件中的 `application/zip` (ZIP) 文件。

您可以启用规则 142:8 以生成此参数的事件，并在内联部署中，丢弃违规数据包。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

qp_decode_depth

指定要从每个 Quoted-Printable (QP) 编码的 MIME 邮件附件中提取和解码的最大字节数。可以指定小于 65535 的整数，或指定 0 以禁用解码。指定 -1 以对要解码的字节数不设限制。

您可以启用规则 142:5 以生成此参数的事件，并且在内联部署中，当解码失败（由于编码不正确或数据损坏）时，丢弃违规的数据包。

类型： 整数

有效范围： -1 至 65535

默认值： -1

uu_decode_depth

指定要从每个 Unix-to-Unix 编码（UuEncode 编码）的 MIME 邮件附件中提取和解码的最大字节数。可以指定小于 65535 的整数，或指定 0 以禁用解码。指定 -1 以对要解码的字节数不设限制。

您可以启用规则 142:7 以生成此参数的事件，并且在内联部署中，当解码失败（由于编码不正确或数据损坏）时，丢弃违规的数据包。

类型： 整数

有效范围： -1 至 65535

默认值： -1

POP 检查器规则

对 生成事件并在内联部署中丢弃攻击性数据包启用 pop 检查器规则。

表 *19: POP* 检查器规则

GID:SID	Rule Message
142:1	未知 POP3 命令
142:2	未知 POP3 响应
142:4	base64 解码失败
142:5	Quoted-Printable 解码失败
142:7	Unix-to-Unix 解码失败
142:8	文件解压缩失败

POP 检查器入侵规则选项

vba_data

将检测光标设置为 Microsoft Office Visual Basic for Applications 宏缓冲区。

语法： vba_data;

示例： vba_data;

第 18 章

端口扫描检查器

端口扫描检查器概述

类型	检查器（探测器）
使用方式	全局
实例类型	全局
所需其他检查器	无
已启用	false

端口扫描是一种通常被攻击者用作攻击前奏的网络侦察形式。在端口扫描中，攻击者发送旨在探测目标主机上的网络协议和服务的数据包。通过检查主机响应时发送的数据包，攻击者通常可以直接或通过推理确定主机上的哪些端口是开放的，以及哪种应用协议正在这些端口上运行。

端口扫描本身不算是攻击。网络上的合法用户可能会采用与攻击者类似的端口扫描技术。

`port_scan` 检查器可检测四种类型的端口扫描，并监控 TCP、UDP、ICMP 和 IP 协议上的连接尝试。通过检测活动模式， `port_scan` 检查器帮助确定哪些端口扫描可能是恶意的。

表 *20:* 端口扫描协议类型

协议 (Protocol)	说明
TCP	检测 TCP 探针，例如 SYN 扫描、ACK 扫描、TCP connect() 扫描和带异常标志组合（如 Xmas tree、FIN 和 NULL）的扫描。
UDP	检测 UDP 探针，如零字节 UDP 数据包。

协议 (Protocol)	说明
ICMP	检测 ICMP 回应请求 (ping)。
IP	检测 IP 协议扫描。Snort 不会查找开放端口，而是搜索目标主机上支持的 IP 协议。

根据目标主机的数量、扫描主机的数量和扫描的端口数量，端口扫描通常分为四种类型。

表 *21:* 端口扫描类型

类型	说明
端口扫描	一对一端口扫描，在这种扫描中，攻击者使用一个或几个主机扫描单个目标主机上的多个端口。 一对一端口扫描具有如下特征： • 扫描主机的数量少 • 扫描单个主机 • 扫描的端口数量多 端口扫描选项检测 TCP、UDP 和 IP 端口扫描。
端口清扫	一对多端口清扫，在这种扫描中，攻击者使用一个或几个主机扫描多个目标主机上的单个端口。 端口扫描具有以下特征： • 扫描主机的数量少 • 扫描的主机数量多 • 扫描的唯一端口数量少 端口清扫检测 TCP、UDP、ICMP 和 IP 端口清扫。
诱骗端口扫描	一对一端口扫描，在这种攻击中，攻击者将伪造的源 IP 地址与真实的扫描 IP 地址混合在一起。 诱骗端口扫描具有如下特征： • 扫描主机的数量多 • 一次扫描的端口数量少 • 扫描的主机为一个（或数量少） 诱骗端口扫描选项检测 TCP、UDP 和 IP 协议端口扫描。

类型	说明
分布式端口扫描	多对一端口扫描，在这种攻击中，多个主机查询单个主机是否有开放端口。 分布式端口扫描具有如下特征： • 扫描主机的数量多 • 一次扫描的端口数量多 • 扫描的主机为一个（或数量少） 分布式端口扫描检测 TCP、UDP 和 IP 协议端口扫描。

端口扫描敏感度级别

port_scan 检查器提供三个默认扫描灵敏度级别。

- default_low_port_scan
- default_med_port_scan
- default_high_port_scan

可以使用各种过滤器配置其他扫描灵敏度级别：

- 扫描
- 拒绝
- nets
- 端口

port_scan 检查器通过收集被探测主机的否定响应来了解探测。例如，当 web 客户端使用 TCP 连接到 web 服务器时，客户端可以假设 web 服务器监听 80 端口。但是，当攻击者探测服务器时，攻击者事先并不知道其是否提供Web服务。当port_scan检测器看到否定响应（ICMP不可达或TCP RST数据包）时，它会将该响应记录为潜在的端口扫描。当目标主机位于设备（例如，过滤否定响应的防火墙或路由器）的另一端时，这个过程更难以执行。在这种情况下，port_scan检测器可以根据选择的灵敏度级别生成已过滤端口扫描事件。

配置端口扫描检查器的最佳实践

要优化端口扫描检测，我们建议您调整 port_scan 检查器以匹配您的网络。

- 确保仔细配置 watch_ip 参数。 watch_ip 参数可帮助 port_scan 检查器过滤网络中非常活跃的合法主机。最常见的示例包括 NAT IP、DNS 缓存服务器、系统日志服务器和 NFS 服务器。
- port_scan 检查器可能生成的大多数误报都属于过滤后的扫描 警报 类型。 警报 类型可能会指示主机在特定时间段内过度活跃。如果主机持续生成过滤式扫描 警报 类型，请将主机添加到 ignore_scanners 列表或使用较低的扫描灵敏度级别。

- 利用 Priority Count、Connection Count、IP Count、Port Count、IP 范围和 Port 范围确定误报。确定误报的最简单方法是通过简单的比率估计。下面列出了需要估计的比率以及表示合法扫描和误报的相关值。
  - 连接计数/IP 计数 - 此比率表示每个 IP 的估计平均连接数。对于端口扫描，此比率应较高。对于端口扫描，此比率应较低。
  - 端口计数/IP 计数 - 此比率表示每个 IP 连接到的端口的估计平均值。对于端口扫描，此比率应较高，表示通过较少的 IP 连接到的被扫描主机的端口。对于端口扫描，此比率应较低，表示扫描主机连接到的端口较少，但主机数量较多。
  - 连接计数/端口计数 - 此比率表示每个端口的估计平均连接数。对于端口扫描，此比率应较低。这表示每个连接都连接到不同的端口。对于端口扫描，此比率应较高。这表示有许多连接到同一端口。

  优先级计数越高，实际端口扫描或端口扫描的可能性越大（除非主机由防火墙管理）。
- 如果无法检测端口扫描，可以降低扫描灵敏度级别。越高的扫描灵敏度级别越能获得最好的保护。低扫描灵敏度级别仅根据错误响应生成警报，而不会捕获过滤的扫描。低扫描灵敏度级别错误响应可指示端口扫描，而低灵敏度级别生成的警报非常准确，并且需要最少的调整。过滤后或高灵敏度级别的扫描容易出现误报。

端口扫描检查器参数

memcap

指定最大跟踪器内存（以字节为单位）。

类型： 整数

有效范围： 1024 到 9,007,199,254,740,992 (maxSZ)

默认值： 10,485,760

原型

指定要监控的协议。提供协议缩写字符串。要指定多个协议，请使用空格分隔各协议缩写。

类型： 字符串

有效值： tcp、 udp、 icmp、 ip、 all

默认值： all

scan_types

指定要检查的端口扫描的类型。提供协议缩写字符串。要指定多个协议，请使用空格分隔各个协议字符串。

类型： 字符串

有效值： `portscan`、 `portscan`、 `decay_portscan`、 `distributed_portscan`、 `all`

默认值： `all`

watch_ip

指定 CIDR 块和 IP 的列表，以及要监控的可选端口。

如果未定义 `watch_ip` ， `port_scan` 检查器会检查所有网络流量。

类型： 字符串

有效值： CIDR 或 IP 地址、CIDR 或 IP 地址列表

默认值： None

alert_all

指定是否对已建立窗口内超过阈值的所有事件发出警报。如果 `alert_all` 设置为 `false`，则 `port_scan` 检查器仅对窗口内的第一个超过阈值的事件发出警报。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

include_midstream

指定是否列出 CIDR 带可选端口。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

tcp_decoy.rejects

指定给出否定响应的扫描尝试次数。

类型： 整数

有效范围： `0` 至 `65535`

默认值： `15`

tcp_decoy.ports

指定端口（或协议）较先前尝试被更改的次数。

类型： 整数

有效范围： `0` 至 `65535`

默认值： `25`

tcp_decoy.scan

指定扫描尝试的次数。

类型： 整数

有效范围： 0 至 65535

默认值： 100

tcp_decoy.nets

指定自先前尝试起已更改的地址次数。

类型： 整数

有效范围： 0 至 65535

默认值： 25

tcp_dist.rejects

指定给出否定响应的扫描尝试次数。

类型： 整数

有效范围： 0 至 65535

默认值： 15

tcp_dist.ports

指定端口（或协议）较先前尝试被更改的次数。

类型： 整数

有效范围： 0 至 65535

默认值： 25

tcp_dist.scans

指定扫描尝试的次数。

类型： 整数

有效范围： 0 至 65535

默认值： 100

tcp_dist.nets

指定自先前尝试起已更改的地址次数。

类型： 整数

有效范围： 0 至 65535

默认值： 25

tcp_ports.rejects

指定给出否定响应的扫描尝试次数。

类型： 整数

有效范围： 0 至 65535

默认值： 15

tcp_ports.ports

指定端口（或协议）较先前尝试被更改的次数。

类型： 整数

有效范围： 0 至 65535

默认值： 25

tcp_ports.scans

指定扫描尝试的次数。

类型： 整数

有效范围： 0 至 65535

默认值： 100

tcp_ports.nets

指定自先前尝试起已更改的地址次数。

类型： 整数

有效范围： 0 至 65535

默认值： 25

tcp_sweep.rejects

指定给出否定响应的扫描尝试次数。

类型： 整数

有效范围： 0 至 65535

默认值： 15

tcp_swave.ports

指定端口（或协议）较先前尝试被更改的次数。

类型： 整数

有效范围： 0 至 65535

默认值： 25

tcp_swave.scans

指定扫描尝试的次数。

类型： 整数

有效范围： 0 至 65535

默认值： 100

tcp_sweep.nets

指定自先前尝试起已更改的地址次数。

类型： 整数

有效范围： 0 至 65535

默认值： 25

udp_decoy.rejects

指定给出否定响应的扫描尝试次数。

类型： 整数

有效范围： 0 至 65535

默认值： 15

udp_decoy.ports

指定端口（或协议）较先前尝试被更改的次数。

类型： 整数

有效范围： 0 至 65535

默认值： 25

udp_decoy.scans

指定扫描尝试的次数。

类型： 整数

有效范围： 0 至 65535

默认值： 100

udp_decoy.nets

指定自先前尝试起已更改的地址次数。

类型： 整数

有效范围： 0 至 65535

默认值： 25

udp_dist.rejects

指定给出否定响应的扫描尝试次数。

类型： 整数

有效范围： 0 至 65535

默认值： 15

udp_dist.ports

指定端口（或协议）较先前尝试被更改的次数。

类型： 整数

有效范围： 0 至 65535

默认值： 25

udp_dist.scans

指定扫描尝试的次数。

类型： 整数

有效范围： 0 至 65535

默认值： 100

udp_dist.nets

指定自先前尝试起已更改的地址次数。

类型： 整数

有效范围： 0 至 65535

默认值： 25

udp_ports.rejects

指定给出否定响应的扫描尝试次数。

类型： 整数

有效范围： 0 至 65535

默认值： 15

udp_ports.ports

指定端口（或协议）较先前尝试被更改的次数。

类型： 整数

有效范围： 0 至 65535

默认值： 25

udp_ports.scans

指定扫描尝试的次数。

类型： 整数

有效范围： 0 至 65535

默认值： 100

udp_ports.nets

指定自先前尝试起已更改的地址次数。

类型： 整数

有效范围： 0 至 65535

默认值： 25

udp_swave.rejects

指定给出否定响应的扫描尝试次数。

类型： 整数

有效范围： 0 至 65535

默认值： 15

udp_sweep.ports

指定端口（或协议）较先前尝试被更改的次数。

类型： 整数

有效范围： 0 至 65535

默认值： 25

udp_sweep.scans

指定扫描尝试的次数。

类型： 整数

有效范围： 0 至 65535

默认值： 100

udp_sweep.nets

指定自先前尝试起已更改的地址次数。

类型： 整数

有效范围： 0 至 65535

默认值： 25

ip_decoy.rejects

指定给出否定响应的扫描尝试次数。

类型： 整数

有效范围： 0 至 65535

默认值： 15

ip_decoy.ports

指定端口（或协议）较先前尝试被更改的次数。

类型： 整数

有效范围： 0 至 65535

默认值： 25

ip_decoy.scans

指定扫描尝试的次数。

类型： 整数

有效范围： 0 至 65535

默认值： 100

ip_decoy.net

指定自先前尝试起已更改的地址次数。

类型： 整数

有效范围： 0 至 65535

默认值： 25

ip_dist.rejects

指定给出否定响应的扫描尝试次数。

类型： 整数

有效范围： 0 至 65535

默认值： 15

ip_dist.ports

指定端口（或协议）较先前尝试被更改的次数。

类型： 整数

有效范围： 0 至 65535

默认值： 25

ip_dist.scans

指定扫描尝试的次数。

类型： 整数

有效范围： 0 至 65535

默认值： 100

ip_dist.nets

指定自先前尝试起已更改的地址次数。

类型： 整数

有效范围： 0 至 65535

默认值： 25

ip_sweep.rejects

指定给出否定响应的扫描尝试次数。

类型： 整数

有效范围： 0 至 65535

默认值： 15

ip_sweep.ports

指定端口（或协议）较先前尝试被更改的次数。

类型： 整数

有效范围： 0 至 65535

默认值： 25

ip_sweep.scans

指定扫描尝试的次数。

类型： 整数

有效范围： 0 至 65535

默认值： 100

ip_sweep.nets

指定自先前尝试起已更改的地址次数。

类型： 整数

有效范围： 0 至 65535

默认值： 25

ip_proto.rejects

指定给出否定响应的扫描尝试次数。

类型： 整数

有效范围： 0 至 65535

默认值： 15

ip_proto.ports

指定端口（或协议）较先前尝试被更改的次数。

类型： 整数

有效范围： 0 至 65535

默认值： 25

ip_proto.scans

指定扫描尝试的次数。

类型： 整数

有效范围： 0 至 65535

默认值： 100

ip_proto.nets

指定自先前尝试起已更改的地址次数。

类型： 整数

有效范围： 0 至 65535

默认值： 25

icmp_sweep.rejects

指定给出否定响应的扫描尝试次数。

类型： 整数

有效范围： 0 至 65535

默认值： 15

icmp_sweep.ports

指定端口（或协议）较先前尝试被更改的次数。

类型： 整数

有效范围： 0 至 65535

默认值： 25

icmp_sweep.scans

指定扫描尝试的次数。

类型： 整数

有效范围： 0 至 65535

默认值： 100

icmp_sweep.nets

指定自先前尝试起已更改的地址次数。

类型： 整数

有效范围： 0 至 65535

默认值： 25

tcp_window

指定传输控制协议 (TCP) 扫描的检测间隔。

类型： 整数

有效范围： 0 到 4,294,967,295（最大 32）

默认值： 0

udp_window

指定用户数据报协议 (UDP) 扫描的检测间隔。

类型： 整数

有效范围： 0 到 4,294,967,295（最大 32）

默认值： 0

ip_window

指定互联网协议 (IP) 扫描的检测间隔。

类型： 整数

有效范围： 0 到 4,294,967,295（最大 32）

默认值： 0

icmp_window

指定互联网控制消息协议 (ICMP) 扫描的检测间隔。

类型： 整数

有效范围： 0 到 4,294,967,295（最大 32）

默认值： 0

端口扫描检查器规则

启用 port_scan 检查器规则，以 生成事件并在内联部署中丢弃攻击性数据包。

表 *22:* 端口扫描检查器规则

GID:SID	Rule Message
122:1	TCP 端口扫描
122:2	TCP 诱骗端口扫描
122:3	TCP 端口清扫
122:4	TCP 分布式端口扫描
122:5	TCP 过滤的端口扫描
122:6	TCP 过滤的诱骗端口扫描
122:7	TCP 过滤的端口清扫
122:8	TCP 过滤的分布式端口扫描
122:9	IP 协议扫描
122:10	IP 诱骗协议扫描
122:11	IP 协议清扫
122:12	IP 分布式协议扫描
122:13	IP 过滤的协议扫描
122:14	IP 过滤的诱骗协议扫描
122:15	IP 过滤的协议清扫
122:16	IP 过滤的分布式协议扫描
122:17	UDP 端口扫描
122:18	UDP 诱骗端口扫描
122:19	UDP 端口清扫
122:20	UDP 分布式端口扫描
122:21	UDP 过滤的端口扫描
122:22	UDP 过滤的诱骗端口扫描

GID:SID	Rule Message
122:23	UDP 过滤的端口清扫
122:24	UDP 过滤的分布式端口扫描
122:25	ICMP 清扫
122:26	ICMP 过滤的清扫
122:27	开放端口

端口扫描检查器入侵规则选项

`port_scan` 检查器没有任何入侵规则选项。

第 19 章

速率过滤器

速率过滤器概述

类型	模块（基本）
使用方式	情景
实例类型	单例对象
已启用	false

基于速率的攻击通过向网络或主机发送过多的流量，企图让网络或主机不堪重负，导致其速度下降或拒绝合法请求。您可以使用基于速率的防御来更改入侵规则的动作，以响应对该规则的过多匹配。

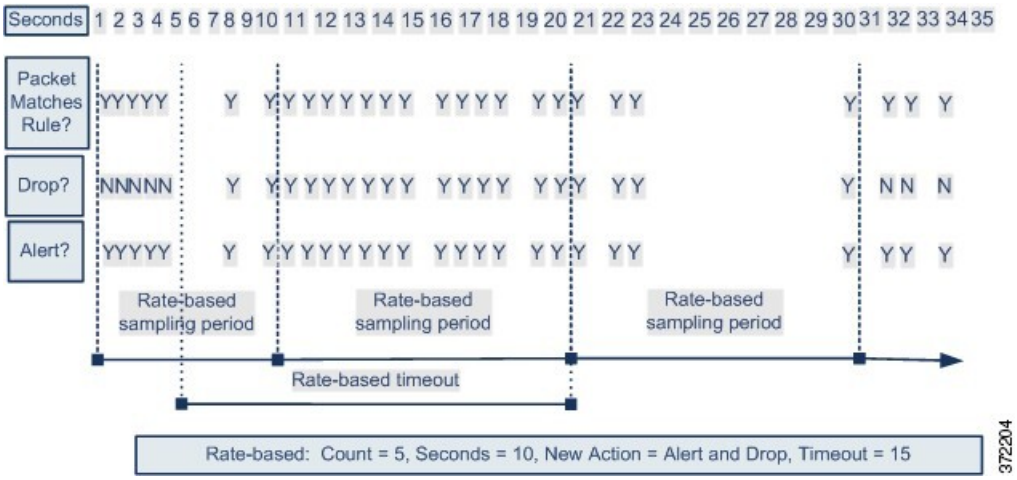
rate_filter 检测在给定间隔内何时发生规则匹配过多。此功能可以用于内联部署的受管设备上，先在指定时间内拦截基于速率的攻击，然后恢复为规则匹配项仅生成事件而不丢弃流量的规则状态。

您可以配置 rate_filter 以响应任何入侵规则，但必须为 rate_filter 启用您指定的规则才能检测攻击和响应。例如，要建立针对 DDOS/SYN 泛洪攻击的防御，请启用规则 135:1（已接收 TCP SYN），并将 rate_filter 配置为对规则 135:1 的触发次数过多发出警报。

基于速率的攻击防御可确定异常流量模式，并可将这些流量对合法请求的影响降至最低。您可以识别出发往一个或多个特定目标 IP 地址或者由一个或多个特定源 IP 地址发出的流量中存在的过多规则匹配项。也可以对检测的所有流量中符合特定规则的过多匹配项作出响应。

下图显示的例子中，攻击者正在尝试访问主机。反复尝试查找密码触发了配置有基于速率的攻击防御的规则。当在 10 秒的时间跨度内发生五次规则匹配之后，基于速率的设置会将规则属性更改为“丢弃并生成事件”(Drop and Generate Events)。新的规则属性在 15 秒之后超时。

请注意，到达超时时间后，在接下来的基于速率的采样周期内，系统仍然丢弃数据包。如果采样速率高于当前或前一个采样周期的阈值，新操作将继续。只有在采样周期完毕而采样速率低于阈值速率之后，新操作才会恢复为“生成事件”(Generate Events)。



可以对同一规则以及不同规则定义多个基于速率的过滤器。在定义了多个基于速率的过滤器的入侵策略中，策略中列出的第一个过滤器具有最高优先级。当两个基于速率的过滤器的操作相冲突时，系统将执行第一个基于速率的过滤器的操作。

为 `rate_filter` 设置的配置参数适用于部署中的所有流量。但是，系统为系统监控的每个唯一连接维护一个单独的计数器，用于记录采样期间内的匹配项数量。系统还会针对每个连接将更改应用到操作。

注释 基于速率的操作无法启用禁用的规则，也无法丢弃与禁用的规则匹配的流量。

速率过滤器参数

rate_filter[]

指定 `rate_filter` 信息的数组。每个 `rate_filter` 包括一组字段，如果流量包含基于速率的攻击，这些字段可以更改规则操作。

类型： 数组（对象）

示例：

```
{
    "rate_filter": {
        "data": [
            {
                "apply_to": "[10.1.2.100, 10.1.2.101]",
                "count": 5,
                "gid": 135,
                "new_action": "alert",
                "seconds": 1,
```

```
                "sid": 1,
                "timeout": 5,
                "track": "by_src"
            }
        ],
        "enabled": true,
        "type": "singleton"
    }
}
```

rate_filter[].gid

指定标识要匹配的规则的生成器 ID (GID)。

类型： 整数

有效范围： 0 到 4,294,967,295（最大 32）

默认值： 1

rate_filter[].sid

指定标识要匹配的规则的签名 ID (SID)。

类型： 整数

有效范围： 0 到 4,294,967,295（最大 32）

默认值： 1

rate_filter[].track

指定过滤器以匹配源地址或目标地址。

类型： enum

有效值：

- by_src：仅过滤与 rate_filter[].gid 和 rate_filter[].sid指定的规则匹配的流量，且源地址与 rate_filter[].apply_to匹配。
- by_dst：仅过滤与 gid 和 sid指定的规则匹配且目标地址与 rate_filter[].apply_to匹配的流量。
- by_rule：过滤与 rate_filter[].gid 和 rate_filter[].sid指定的规则匹配的所有流量。

默认值： by_src

rate_filter[].count

指定在应用替代操作 (rate_filter[].new_action) 之前，在采样期 (rate_filter[].seconds) 内允许的规则匹配数。

类型： 整数

有效范围： 0 到 4,294,967,295（最大 32）

默认值： 1

rate_filter[].seconds

指定与流量匹配的采样周期中的秒数。 `rate_filter[].seconds` 表示将匹配项的内部计数器重置为零之前经过的时间量。

类型： 整数

有效范围： `0` 到 `4,294,967,295`（最大 32）

默认值： `1`

rate_filter[].new_action

指定为响应超过 `rate_filter[].seconds` 和 `rate_filter[].count`指定的限制的流量中的匹配项而采取的操作。

类型： 字符串

有效值：以下字符串之一： `alert`、 `block`、 `drop`、 `log`、 `pass`、 `react`、 `reject`、 `rewrite`。

默认值： `alert`

rate_filter[].timeout

指定为响应匹配流量而执行由 `rate_filter[].new_action` 指定的操作的秒数。

类型： 整数

有效范围： `0` 到 `4,294,967,295`（最大 32）

默认值： `0`

rate_filter[].apply_to

指定要根据流量源或目标地址匹配的网络地址列表，具体取决于 `rate_filter[].track`的值。

类型： 字符串

有效值： 有效的 IPv4 地址或 CIDR 格式的 IPv4 地址块。

默认值： None

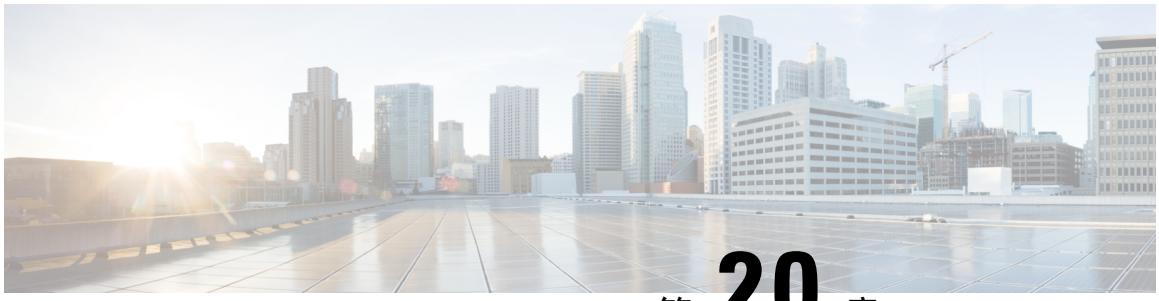
速率过滤器规则

`rate_filter` 没有任何关联规则。

您可以配置 `rate_filter` 以响应任何入侵规则。启用规则的 `rate_filter` 以检测攻击和响应。

速率过滤器入侵规则选项

`rate_filter` 没有任何入侵规则选项。



第 20 章

S7CommPlus 检查器

S7CommPlus 检查器概述

类型	检查器（服务）
使用方式	检测
实例类型	多实例
所需其他检查器	`stream_tcp`
已启用	`false`

S7CommPlus 是由Siemens 开发的专有协议。S7CommPlus 支持Siemens S7 产品系列的可编程逻辑控制器之间的通信。

`S7Commplus` 检查器可检查并分析 S7Commplus 流量。可以设置入侵规则选项以对指定的 S7CommPlus 函数和操作代码报头字段发出警报，并检测 S7CommPlus 流量中的攻击。

配置 S7CommPlus 检查器的最佳实践

如果您的网络中没有开启的 S7CommPlus 设备，请不要在应用于流量的网络分析策略中开启 `S7CommPlus` 巡检器。

S7CommPlus 检查器参数

S7CommPlus TCP 端口配置

绑定程序 检查器定义 S7CommPlus TCP 端口配置。有关详细信息，请参阅绑定程序检查器概述，第 13 页。

示例：

```
[
    {
        "when": {
            "role": "server",
            "proto": "tcp",
            "ports": "102"
        },
        "use": {
            "type": "s7commplus"
        }
    },
    {
        "when": {
            "role": "any",
            "service": "s7commplus"
        },
        "use": {
            "type": "s7commplus"
        }
    }
]
```

注释 `s7commplus` 检查器不提供任何参数。

S7CommPlus 检查器规则

启用 `s7commplus` 检查器规则以 生成事件并在内联部署中丢弃攻击性数据包。

表 *23: S7CommPlus* 检查器规则

GID:SID	Rule Message
149:1	S7commplus MBAP 信头中的长度与给定 S7commplus 功能所需的长度不匹配
149:2	S7commplus 协议 ID 为非零
149:3	保留的 S7commplus 功能代码正在使用中

S7CommPlus 检查器入侵规则选项

可以单独使用 S7Commplus 关键字或组合来创建自定义入侵规则，这些规则根据 S7Commplus 检查器检测的流量识别攻击。对于可配置的关键字，指定在允许的范围内的单个已知值或单个整数。

请注意以下提示：

- 同一规则中的多个 S7commplus 关键字都使用 AND 运算。
- 在同一规则中使用多个 s7commplus_func 或 s7commplus_opcode 关键字会否定该规则。被否定的规则无法匹配流量。要使用这些关键字搜索多个值，请创建多个规则。

s7commplus_content

使用 s7commplus_content 关键字将光标定位到 S7CommPlus 数据包负载的开头。我们建议您在 S7CommPlus 入侵规则中使用 content 或 protected_content 关键字之前设置此关键字。

语法： s7commplus_content;

示例： s7commplus_content;

s7commplus_func

使用 s7commplus_func 关键字匹配 S7Commplus 信头中的以下值之一。可以指定 S7CommPlus 参数名称或相应的十六进制代码。

类型： 字符串

语法： s7commplus_fuc:<header_parameter>;

有效值：

名称	代码
explore	0x04BB
createobject	0x04CA
deleteobject	0x04D4
setvariable	0x04F2
getlink	0x0524
setmultivar	0x0542
getmultivar	0x054C
beginsequence	0x0556
endsequence	0x0560
invoke	0x056B

名称	代码
getvarsubstr	0x0586
0x0 至 0xFF	请注意，数字表达式允许使用其他值。

示例： `s7commplus_fuc: createobject;`

s7commplus_opcode

使用 `s7commplus_opcode` 关键字匹配 S7Commplus 信头中的以下值之一。可以指定 S7CommPlus 参数名称或相应的十六进制代码。

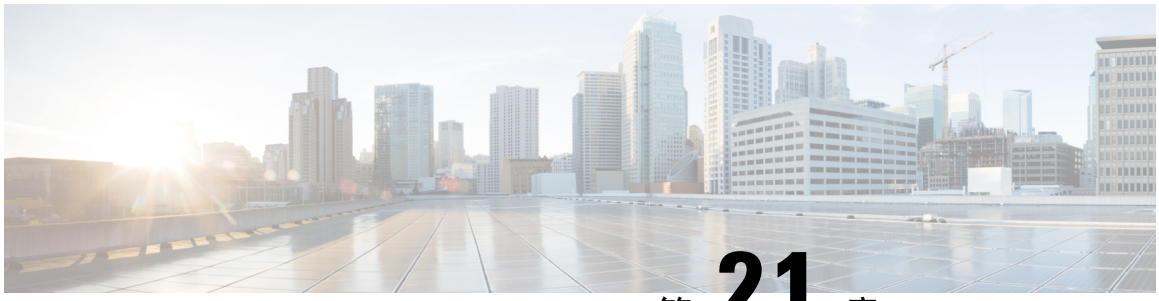
类型： 字符串

语法： `s7commplus_opcode:<header_parameter>`

有效值：

名称	代码
请求	0x31
效率低下	0x32
通知	0x33
response2	0x02
0x0 至 0xFF	请注意，数字表达式允许使用其他值。

示例： `s7commplus_opcode:0x31;`



第 21 章

SIP 检查器

SIP 检查器概述

类型	检查器（服务）
使用方式	检测
实例类型	多实例
所需其他检查器	`stream_udp`
已启用	`true`

会话发起协议 (SIP) 管理包括一个或多个参与者的实时呼叫会话的创建、修改和断开。SIP 可以控制的应用包括：互联网电话、多媒体会议、即时消息、在线游戏和文件传输。SIP 协议是基于文本的请求和响应协议。

每个 SIP 请求中的 `方法` 字段识别请求的目的，`请求 URI` 则指定发送请求的目的地。每个 SIP 响应中的状态代码指明请求操作的结果。SIP 协议使用 TCP（端口 5060）或 UDP（端口 5061）。

在 SIP 创建呼叫会话后，SIP 可以通过实时传输协议 (RTP) 传输音频和视频流。对于数据通道参数协商、会话公告和会话邀请，RTP 在 SIP 消息正文中使用会话描述协议 (SDP)。

`SIP` 检查器可检测并分析网络流量中的 SIP 消息。`SIP` 检查器会提取 SIP 报头和消息正文，并将 SIP 消息正文中的所有数据传递到检测引擎。

`SIP` 检查器可检测 SIP 流量中的异常和已知漏洞，包括无序和无效的呼叫序列。

注释

- SIP 检查器不会解码 RTP 消息。 SIP 检查器会根据 SDP 数据中定义的端口识别 RTP 信道。
- UDP 通常传输 SIP 支持的媒体会话。 SIP 检查器从已解码的 UDP 流中获取会话跟踪信息。
- SIP 规则选项允许您将检测光标定位到 SIP 数据包信头或消息体，并将检测限制为针对特定 SIP 方法或状态码的数据包。

SIP 检查器参数

SIP 服务配置

绑定程序 检查器定义 SIP 服务配置。有关详细信息，请参阅绑定程序检查器概述 ， 第 13 页。

示例：

```
[
    {
        "when": {
            "role": "any",
            "service": "sip"
        },
        "use": {
            "type": "sip"
        }
    }
]
```

ignore_call_channel

指定是否检测音频/视频数据通道流量。启用后， SIP 检查器会解码所有非数据 SIP 通道流量，并忽略音频/视频 SIP 数据通道流量。

类型： boolean

有效值： true、 false

默认值： false

max_call_id_len

指定 Call-ID 信头字段中允许的最大字节数。Call-ID 字段唯一地识别请求和响应中的 SIP 会话。当 max_call_id_len 为 0 时， sip 检查器不会生成警报。

您可以启用规则 140:5 以生成事件，并在内联部署中，丢弃违规数据包。当 Call-ID 报头长度大于 max_call_id_len 的值时， sip检查器会生成事件。

类型： 整数

有效范围： 0 至 65535

默认值： 256

max_contact_len

指定 联系人 信头字段中允许的最大字节数。联系人 字段提供用以指定与后续消息进行联系的位置的 URI。值为 0 时， sip 检查器不会生成警报。

您可以启用规则 140:15 以生成事件，并在内联部署中，丢弃违规数据包。当 联系人 信头字段长度大于 max_contact_len 的值时， sip检查器会生成事件。

类型： 整数

有效范围： 0 至 65535

默认值： 256

max_content_len

指定在消息正文的内容中允许的最大字节数。值为 0 时， sip 检查器不会生成警报。

您可以启用规则 140:16 以生成事件，并在内联部署中，丢弃违规数据包。当内容长度大于 max_content_len 的值时， sip检查器会生成事件。

类型： 整数

有效范围： 0 至 65535

默认值： 1024

max_dialogs

指定数据流会话中允许的最大对话数量。如果对话框数量超过该数量，则 sip 检查器将丢弃最早的对话框，直至对话框数量不超过指定的最大数量。

您可以启用规则 140:27 以生成事件，并在内联部署中，丢弃违规数据包。

类型： 整数

有效范围： 1 到 4,294,967,295（最大 32）

默认值： 4

max_from_len

指定 发件人 信头字段中允许的最大字节数。发件人 字段识别消息发起方。值为 0 时， sip 检查器不会生成警报。

您可以启用规则 140:9 以生成事件，并在内联部署中丢弃违规数据包。当 发件人 字段长度大于 max_from_len 的值时， sip检查器会生成事件。

类型： 整数

有效范围： 0 至 65535

默认值： 256

max_request_name_len

指定请求名称中允许的最大字节数。SIP 请求名称是指在 SIP CSeq 事务标识符中指定的方法的名称。值为 0 时， sip 检查器不会生成警报。

您可以启用规则 140:7 以生成事件，并在内联部署中丢弃违规数据包。当请求名称长度大于 max_request_name_len 的值时， sip检查器会生成事件。

类型： 整数

有效范围： 0 至 65535

默认值： 20

max_requestName_len

max_requestName_len 参数已弃用。改用 max_request_name_len 参数。

max_to_len

指定 收件人 信头字段中允许的最大字节数。收件人 字段识别消息收件人。值为 0 时， sip 检查器不会生成警报。

您可以启用规则 140:11 以生成事件，并在内联部署中，丢弃违规数据包。当 收件人 字段长度大于 max_to_len 的值时， sip检查器会生成事件。

类型： 整数

有效范围： 0 至 65535

默认值： 256

max_uri_len

指定 Request-URI 中允许的最大字节数。 Request-URI 指示所请求资源的目的路径。值为 0 时， sip 检查器不会生成警报。

您可以启用规则 140:3 以生成事件，并在内联部署中丢弃违规数据包。Request-URI 字段长度大于 max_uri_len 的值时， sip检查器会生成事件。

类型： 整数

有效范围： 0 至 65535

默认值： 256

max_via_len

指定 通过 信头字段中允许的最大字节数。通过 字段标识要在请求中使用的传输和接收方的位置。值为 0 时， sip 检查器不会生成警报。

您可以启用规则 140:13 来生成事件，并在内联部署中丢弃违规数据包。当 通过 字段长度大于 max_via_len 的值时， sip检查器会生成事件。

类型： 整数

有效范围： 0 至 65535

默认值： 1024

方法

指定 SIP 检测方法列表。方法名称不区分大小写。使用逗号或空格分隔列表中的方法名称。方法名称仅可以包含字母字符、数字和下划线字符。

类型： 字符串

有效值： ack、benotify、bye、cancel、do、info、invite、join、message、notify、options、prack、publish、quarh、refer、register、service、srack、subscribe、unsubscribe、update

默认值： invite cancel ack bye register options

SIP 检查器规则

启用 sip 检查器规则以 生成事件并在内联部署中丢弃攻击性数据包。

表 *24: SIP* 检查器规则

GID:SID	Rule Message
140:2	URI 请求为空
140:3	URI 过长
140:4	空 Call-Id
140:5	Call-Id 过长
140:6	CSeq 编号过大或为负数
140:7	CSeq 中的请求名称过长
140:8	空“发件人”信头
140:9	“发件人”信头过长
140:10	空“收件人”信头
140:11	“收件人”报头过长
140:12	空“通过”信头
140:13	“通过”信头过长
140:14	联系人为空
140:15	联系人过长

GID:SID	Rule Message
140:16	内容长度太大或为负数
140:17	一个数据包包含多条 SIP 消息
140:18	内容长度不匹配
140:19	请求名称无效
140:20	邀请重放攻击
140:21	非法修改会话信息
140:22	响应状态代码不是三位数字
140:23	空 Content-Type 信头
140:24	SIP 版本无效
140:25	请求的 METHOD 和 CSEQ 信头不匹配
140:26	方法未知
140:27	会话中达到的最大对话数

SIP 检查器入侵规则选项

sip_method

SIP 请求识别请求的目的。使用 `sip_method` 关键字来匹配 SIP 请求中的方法。方法名称不区分大小写。使用逗号隔开多种方法。

类型： 字符串

语法： `sip_method:`<methods>中所述：

有效值： `ack`、`benotify`、`bye`、`cancel`、`do`、`info`、`invite`、`join`、`message`、`notify`、`options`、`prack`、`publish`、`quarh`、`refer`、`register`、`service`、`srack`、`subscribe`、`unsubscribe`、`update`

示例： `sip_method: "ack,service,info,bye";`

sip_stat_code

SIP 响应包括三位数字状态代码。SIP 状态代码指明请求操作的结果。使用 `sip_stat_code` 关键字可将 SIP 响应与指定的状态代码进行匹配。

您可以使用任一数字组合指定一个来表示三位数状态代码的第一位数字、一个三位数或一个以逗号分隔的数字列表。如果列表中的任何一个数字与 SIP 响应中的代码相匹配，则列表匹配。

类型： 整数

语法： `sip_stat_code:`<codes>中所述：

有效范围：

- `1` 到 `9`
- `100` 到 `999`

示例： `sip_stat_code: "1";`

表 *25: SIP* 参数值和状态代码

参数值	检测到的状态代码	说明
189	189	设置特定状态代码
1	100 - 199	设置单个数字。
222, 3	222; 300 - 399	设置以逗号分隔的三位数或单位数字列表。

sip_header

使用 `sip_header` 关键字将检测光标定位到提取的 SIP 信头缓冲区的开头。限制对信头字段进行检测。

语法： `sip_header;`

示例： `sip_header;`

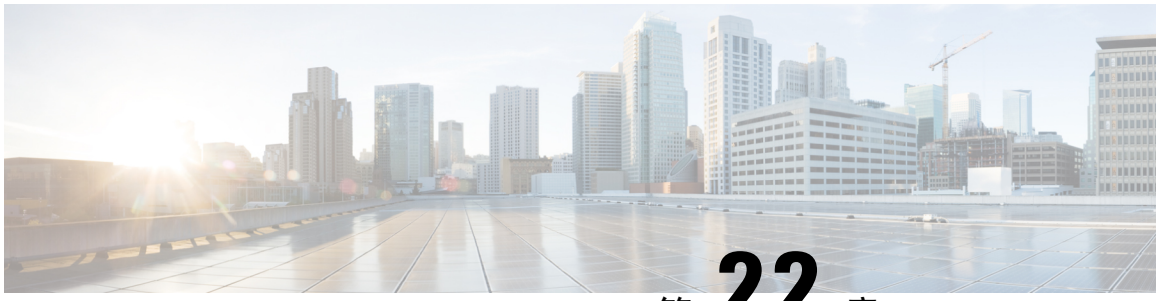
sip_body

使用 `sip_body` 关键字将检测光标定位到提取的 SIP 消息正文的开头。限制对邮件正文进行检查。

语法： `sip_body;`

示例： `sip_body;`

注释 `sip`检查器将提取整个消息正文并使其可供规则引擎使用。规则引擎不仅限于搜索会话描述协议(SDP)内容。



第 22 章

SMTP 检查器

SMTP 检查器概述

类型	检查器（服务）
使用方式	检测
实例类型	多实例
所需其他检查器	`stream_tcp`
已启用	`true`

简单邮件传输协议 SMTP 使邮件客户端能够向邮件服务器发送消息。SMTP 发出命令以将邮件传送给收件人。SMTP 服务器将 TCP 端口 25 用于不安全的会话，或将 TCP 端口 587 用于基于 SSL/TLS 的 SMTP。

`smtp` 检查器会检测 SMTP 流量并分析 SMTP 命令和响应。

`smtp` 检查器识别 SMTP 邮件的命令、信头和正文部分，并提取和解码多用途互联网邮件扩展 (MIME) 附件。MIME 附件可能包括多个附件和跨越多个数据包的大型附件。

`smtp` 检查器识别 SMTP 邮件并将其添加到 Snort 允许列表。启用后，入侵规则将针对异常 SMTP 流量生成事件。

您可以将 `smtp` 检查器配置为：

- 记录发件人邮件 ID、收件人邮件 ID、邮件标题和附件文件名，以及生成的所有事件。

- 删除无关的空格字符，使 SMTP 命令行规范化。smtp 检查器会将空格 (ASCII 0x20) 或制表符 (ASCII 0x09) 规范化。
- 忽略 TLS 加密流量以提高性能。
- 忽略纯文本邮件数据以提高性能。

配置 SMTP 检查器的最佳实践

我们建议您按照 RFC 2821 中的准则来配置 smtp 检查器的核心配置参数：

- max_command_line_len：512 个字符
- max_header_line_len：1024 个字符
- max_response_line_len：512 个字符

SMTP 检查器参数

SMTP 服务配置

绑定程序 检查器定义 SMTP 服务 配置。有关详细信息，请参阅绑定程序检查器概述，第 13 页。

示例：

```
[
    {
        "when": {
            "service": "smtp",
            "role": any
        },
        "use": {
            "type": "smtp"
        }
    }
]
```

alt_max_command_line_len[]

指定 SMTP 命令数组和该命令的备用最大行长度。备用最大行长度将覆盖 SMTP 命令的 max_command_line_len 的值。您可以启用规则 124:4 为此参数生成事件。

类型： 数组

示例：

```
{
    "alt_max_command_line_len": [
        {
            "command": "AUTH",
            "length": 240
        }
```

```
    ]
}
```

alt_max_command_line_len[].command

指定命令字符串。

类型： 字符串

有效值： SMTP 命令

默认值： 请参阅 表 26: SMTP 命令和默认备用命令长度。

alt_max_command_line_len[].length

指定备用最大命令行长度。指定 0 将禁用命令的命令行长度检测。

类型： 整数

有效范围： 0 到 4,294,967,295（最大 32）

默认值： 请参阅 表 26: SMTP 命令和默认备用命令长度。

表 *26: SMTP* 命令和默认备用命令长度

命令	长度
ATRN	255
AUTH	246
BDAT	255
数据	246
DEBUG	255
EHLO	500
EMAL	255
ESAM	255
ESND	255
ESOM	255
ETRN	500
EVFY	255
EXPN	255
HELO	500
帮助	500

命令	长度
IDENT	255
邮件	260
NOOP	255
ONEX	246
QUEU	246
QUIT	246
RCPT	300
RSET	255
SAML	246
SEND	246
大小	255
SOML	246
STARTTLS	246
TICK	246
TIME	246
TURN	246
TURNME	246
VERB	246
VRFY	255
XADR	246
XAUTH	246
XCIR	246
XEXCH50	246
X-EXPS	246
XGEN	246
XLICENSE	246
X-LINK2STATE	246
XQUE	246

命令	长度
XSTA	246
XTRN	246
XUSR	246

auth_cmds

指定发起身份验证交换的 SMTP 命令列表。使用空格分隔多个 SMTP 命令。

类型： 字符串

有效值： SMTP 身份验证交换启动命令

默认值： AUTH XAUTH X-EXPS

b64_decode_depth

指定要从每个 Base64 编码的 MIME 邮件附件中提取和解码的最大字节数。可以指定小于 65535 的整数，或指定 0 以禁用解码。指定 -1 以对要解码的字节数不设限制。

您可以启用规则 124:10 来生成此参数的事件，并且在内联部署中，当解码失败时丢弃违规数据包。

类型： 整数

有效范围： -1 至 65535

默认值： -1

binary_data_cmds

指定一系列 SMTP 命令，这些命令启动发送数据，并在命令后使用长度值（以八位组为单位）以指示要发送的数据量。使用空格分隔多个 SMTP 命令。

类型： 字符串

有效值： 使用数据长度参数的有效 SMTP 数据发送启动命令

默认值： BDATA XEXCH50

bitenc_decode_depth

指定要从每个非编码的 MIME 邮件附件中提取的最大字节数。可以指定一个小于 65535 的整数，或指定 0 以禁用未编码 MIME 附件的提取。指定 -1 对要提取的字节数不设限制。这些附件类型包括 7 位、8 位、二进制以及各种多部分内容类型（例如，纯文本、JPEG 和 PNG 图像、MP4 文件等）。

类型： 整数

有效范围： -1 至 65535

默认值： -1

data_cmds

指定发起发送数据并使用数据结束符 (<CRLF>.<CRLF>) 的 SMTP 命令列表。

类型： 字符串

有效值： 使用数据结束分隔符的 SMTP 数据发送启动命令。

默认值： `DATA`

decompress_pdf

指定是否解压缩 MIME 附件中的 `application/pdf` (PDF) 文件。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

decompress_swf

指定是否解压缩 MIME 附件中的 `application/vnd.adobe.flash-movie` (SWF) 文件。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

decompress_vba

指定是否解压缩 MIME 附件中的 Microsoft Office Visual Basic for Applications 宏文件。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

decompress_zip

指定是否解压缩 MIME 附件中的 `application/zip` (ZIP) 文件。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

email_hdrs_log_depth

指定要从 SMTP 数据中提取的电邮信头的字节数。指定 0 以禁用电邮信头提取。

类型： 整数

有效范围： `0` 到 `20480`

默认值： `1464`

ignore_data

指定是否对邮件数据部分进行解码（MIME 邮件信头除外）。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

ignore_tls_data

指定是否解码 TLS 加密数据。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

log_email_hdrs

指定是否解码和记录 SMTP 邮件信头以及所有已生成的会话事件。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

log_filename

指定是否解码和记录从 MIME 正文内的 `Content-Disposition` 报头提取的 MIME 附件文件名，以及为会话生成的所有事件。如果邮件包含多个 MIME 附件，则 SMTP 检查器会记录用逗号分隔的文件名。SMTP 检查器记录的内容不超过 1024 字节。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

log_mailfrom

指定是否解码和记录从SMTP `MAIL FROM` 命令提取的发件人邮件地址，以及生成的所有会话事件。如果邮件包含多个发件人，则 SMTP 检查器将记录用逗号分隔的发件人。SMTP 检查器记录的内容不超过 1024 字节。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

log_rcptto

指定是否解码和记录来自 SMTP `RCPT TO` 命令的收件人邮件地址，以及生成的所有事件事件。如果邮件包含多个收件人，则 SMTP 检查器将记录用逗号分隔的收件人。SMTP 检查器记录的内容不超过 1024 字节。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

max_auth_command_line_len

指定 SMTP 身份验证命令行接受的最大字节数。

您可以启用规则 124:15 以生成事件，并在内联部署中丢弃违规数据包。指定 0 以禁用对 SMTP `AUTH` 命令的警报，或从 Snort 配置中省略 `max_auth_command_line_len` 参数。

类型： 整数

有效范围： `0` 至 `65535`

默认值： `1000`

max_command_line_len

指定 SMTP 命令行接受的最大字节数。

RFC 2821（网络工作组关于 SMTP 的规范）建议的最大命令行长度为 512 字节。指定 0 以禁用对 SMTP 命令行长度的警报，或从 Snort 配置中省略 `max_command_line_len` 参数。

您可以启用规则 124:1 来生成事件，并在内联部署中丢弃违规数据包。

类型： 整数

有效范围： `0` 至 `65535`

默认值： `512`

max_header_line_len

指定 SMTP 数据信头行接受的最大字节数。

RFC 2821（网络工作组关于 SMTP 的规范）建议的最大数据信头行长度为 1024 字节。指定 0 以禁用 SMTP 数据信头行长度警报，或从 Snort 配置中省略 `max_header_line_len` 参数。

您可以启用规则 124:2 和 124:7 以生成事件，并在内联部署中丢弃违规数据包。

类型： 整数

有效范围： `0` 至 `65535`

默认值： `1000`

max_response_line_len

指定 SMTP 响应行接受的最大字节数。

RFC 2821（网络工作组关于 SMTP 的规范）建议的最大响应行长度为 512 字节。指定 0 以禁用有关 SMTP 响应行长度的警报，或从 Snort 配置中省略 `max_response_line_len` 参数。

您可以启用规则 124:3 以生成事件，并在内联部署中丢弃违规数据包。

类型： 整数

有效范围： `0` 至 `65535`

默认值： `512`

规范化

指定规范化所有命令、无命令还是命令列表。您可以在 `normalize_cmds` 参数中指定命令列表。检查器会检查命令后是否存在多个空格 (ASCII 0x20) 或制表符 (ASCII 0x09) 字符。

类型： enum

有效值：

- `none`
- `cmds`
- `all`

默认值： `none`

normalize_cmds

指定要规范化的 SMTP 命令列表。使用空格分隔多个 SMTP 命令。

类型： 字符串

有效值： SMTP 命令

默认值： None

qp_decode_depth

指定要从每个 Quoted-Printable (QP) 编码的 MIME 邮件附件中提取和解码的最大字节数。可以指定小于 65535 的整数，或指定 0 以禁用解码。指定 -1 以对要解码的字节数不设限制。

您可以启用规则 124:11 来生成事件，并在内联部署中丢弃违规数据包。

类型： 整数

有效范围： `-1` 至 `65535`

默认值： `-1`

uu_decode_depth

指定要从每个 Unix-to-Unix 编码（UuEncode 编码）的 MIME 邮件附件中提取和解码的最大字节数。可以指定小于 65535 的整数，或指定 0 以禁用解码。指定 -1 以对要解码的字节数不设限制。

您可以启用规则 124:13 来为此参数生成事件，并且在内联部署中，当解码失败（例如，由于编码不正确或数据损坏）时，丢弃违规的数据包。

类型： 整数

有效范围： -1 至 65535

默认值： -1

valid_cmds

指定 SMTP 检查器认为有效的 SMTP 命令的其他列表。

即使此列表为空，预处理器仍允许下列有效命令：ATRN AUTH BDAT DATA DEBUG EHLO EMAL ESAM ESND ESOM ETRN EVFY EXPN HELO HELP IDENT MAIL NOOP ONEX QUEU QUIT RCPT RSET SAML SEND SIZE STARTTLS SOML TICK TIME TURN TURNME VERB VRFY X-EXPS X-LINK2STATE XADR XAUTH XCIR XEXCH50 XGEN XLICENSE XQUE XSTA XTRN XUSR.

您可以启用规则 124:5 以生成事件，并在内联部署中丢弃违规数据包。

类型： 字符串

有效值： SMTP 命令

默认值： None

xlink2state

指定 SMTP 检查器如何处理属于 X-Link2State Microsoft Exchange 缓冲区数据溢出攻击的数据包（有关漏洞的说明，请参阅 CVE-2005-0560）。您可以禁用检测 (disable)，启用检测并生成警报 (alert)，也可以启用检测并丢弃违规数据包 (drop)。

您可以启用规则 124:8 来生成此参数的事件，并且在内联部署中丢弃违规数据包。

类型： enum

有效值：

- disable
- 警报
- 丢弃

默认值： alert

SMTP 检查器规则

对 生成事件并在内联部署中丢弃攻击性数据包启用 smtp 检查器规则。

表 *27: SMTP* 检查器规则

GID:SID	Rule Message
124:1	尝试的命令缓冲区溢出
124:2	数据报头缓冲区溢出尝试次数
124:3	尝试的响应缓冲区溢出
124:4	特定命令缓冲区溢出尝试次数
124:5	未知命令
124:6	非法命令
124:7	已尝试报头名称缓冲区溢出
124:8	尝试的 X-Link2State 命令缓冲区溢出
124:10	base64 解码失败
124:11	Quoted-Printable 解码失败
124:13	Unix-to-Unix 解码失败
124:14	Cyrus SASL身份验证攻击
124:15	命令缓冲区溢出所尝试的身份验证
124:16	文件解压缩失败

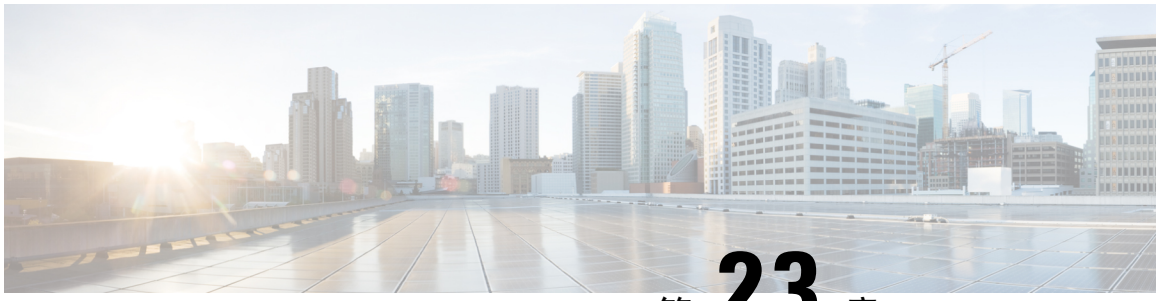
SMTP 检查器入侵规则选项

vba_data

将检测光标设置为 Microsoft Office Visual Basic for Applications 宏缓冲区。

语法： `vba_data;`

示例： `vba_data;`



第 23 章

SnortML

类型	检查器 (被动)
使用方式	检测
实例类型	单例对象
所需其他检查器	`snort_ml_engine`、`http_inspect`
已启用	最大检测

每天都有新的漏洞出现在对现代社会至关重要的软件中。安全分析师会分解这些新漏洞，隔离触发这些漏洞的必要因素，并写入签名来检测针对这些漏洞的攻击。大多数签名实际上只能针对特定漏洞编写。

SnortML 是用于 Snort 入侵防御系统的基于人工网络的漏洞攻击检测。它不仅可以从训练数据中学习检测已知攻击，还可以学习检测以前从未见过的攻击。

`snort_ml` 检查器主要搜索通过 HTTP 的 SQL 注入攻击。由于此检查器可能会影响性能，因此默认情况下仅在处于 最大检测 模式下时启用。

- SnortML 规则， 第 183 页
- SnortML 参数， 第 184 页

SnortML 规则

启用 `snort_ml` 检查器规则以生成事件并在内联部署中丢弃攻击性数据包。默认情况下，仅在 Maximum Detection NAP 策略下启用 `snort_ml` 检查器规则。

表 *28: Snort ML* 检查器规则

GID:SID	Rule Message
411:1	通过基于网络的漏洞攻击检测在 HTTP 参数中发现 (snort_ml) 潜在威胁。

SnortML 参数

uri_depth

指定要从 HTTP URI 扫描的字节数。值 -1 表示无限制。

类型： 整数

有效范围： -1 到 2147483648

默认值： -1

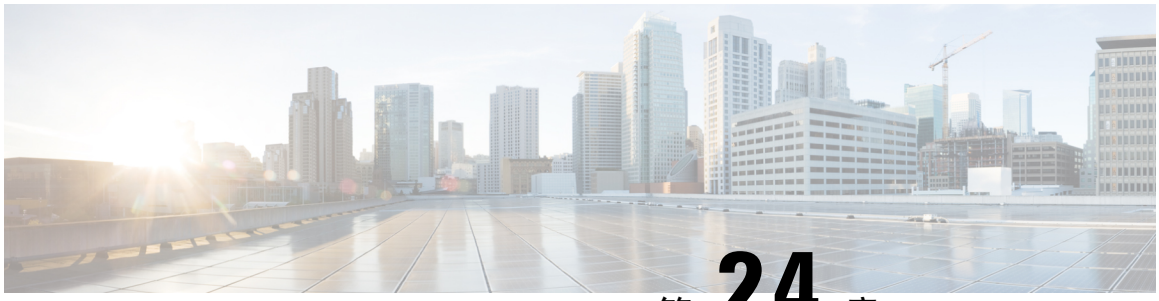
client_body_depth

指定要从 HTTP 客户端主题扫描的字节数。值 -1 表示无限制。

类型： 整数

有效范围： -1 到 2147483648

默认值： 0



第 24 章

SSH 检查器

SSH 检查器概述

类型	检查器（服务）
使用方式	检测
实例类型	多实例
所需其他检查器	无
已启用	`true`

安全外壳协议 (SSH) 是一种网络协议，支持客户端和服务器通过不安全网络进行安全通信。SSH 支持建立隧道，并使用公共密钥加密对远程主机进行身份验证。

您可以使用 SSH 安全地传输文件，或登录远程主机并与命令行交互。SSH 协议相对于 TCP、UDP 或 SCTP 使用端口 22。

`ssh` 检查器解码流数据包并检测以下 SSH 漏洞：

- 质询-响应缓冲区溢出攻击
- CRC-32 攻击
- SecureCRT SSH 客户端缓冲区溢出攻击
- SSH 消息方向不正确

如果主机之间的网络连接已加密，则身份验证后会发生质询响应缓冲区溢出和 CRC-32 攻击。这两种类型的攻击都在身份验证挑战之后立即向服务器发送超过 20 KB 的反常态大量负载。

ssh 检查器通过计算传输到服务器的字节数来检测质询响应缓冲区溢出和 CRC-32 攻击。如果字节超过在预定义数据包数量内定义的限制，则 ssh 检查器会生成警报。CRC-32 攻击仅适用于 SSH 版本 1；质询-响应缓冲区溢出攻击仅适用于 SSH 版本 2。ssh 检查器在会话开始时读取 SSH 版本字符串，以识别攻击类型。

密钥交换前，如果主机试图保护连接，会发生 SecureCRT SSH 客户端缓冲溢出和协议不匹配攻击。SecureCRT SSH 客户端缓冲溢出攻击会向客户端发送过长的协议标识符字符串，从而导致缓冲区溢出。如果非 SSH 客户端应用试图连接到安全 SSH 服务器或者服务器和客户端的版本号不匹配，会出现协议不匹配攻击。

注释 ssh 检查器不处理暴力攻击。

配置 SSH 检查器的最佳实践

我们建议您使用默认的 ssh 检查器配置配置设置。如果超过 max_encrypted_packets 参数中定义的会话加密数据包的最大数量，则 ssh 检查器会停止处理该会话的流量以提高性能。 ssh 检查器仅检测在 SSH 会话开始时出现的 SSH 漏洞。

注释 如果 ssh 检查器在质询-响应溢出或 CRC 32 上生成误报，您可以使用 max_client_bytes 参数增加所需的客户端字节数。

SSH 检查器参数

SSH 服务配置

绑定程序 检查器定义 SSH 服务 配置。有关详细信息，请参阅绑定程序检查器概述 ， 第 13 页。

示例：

```
[
    {
        "when": {
            "service": "ssh",
            "role": any
        },
        "use": {
            "type": "ssh"
        }
    }
]
```

max_encrypted_packets

指定在 ssh 检查器忽略 SSH 会话之前要检查的最大加密数据包数。如果超过会话的加密数据包的最大数量，则 ssh 检查器会停止处理该会话的流量以提高性能。

类型： 整数

有效范围： -1 至 65535

默认值： 25

max_client_bytes

指定在 ssh 检查器对质询-响应溢出或 CRC 32 发出警报之前，要传输到服务器的最大无应答字节数。如果在发送 max_encrypted_packets 之前超出 max_client_bytes 限制，则检查器会假设已发生攻击并忽略流量。

您可以启用规则 128:1 以在检查器检测到质询-响应溢出时生成警报，或启用规则 128:2 以在检查器检测到 CRC 32 漏洞攻击时生成警报。

对于客户端从服务器接收的每个有效响应， ssh 检查器会重置 max_client 字节数的数据包计数。

注释 我们不建议您将 max_client_bytes 设置为 0 或 1。如果将 max_client_bytes 设置为 0 或 1，则 ssh 检查器始终发出警报。

类型： 整数

有效范围： 0 至 65535

默认值： 19600

max_server_version_len

指定 SSH 服务器版本字符串的最大长度。如果 SSH 服务器版本字符串的长度超过 max_server_version_len，则 ssh 检查器会生成警报。您可以启用规则 128:3，以在 Secure CRT 服务器版本字符串溢出时发出警报。

类型： 整数

有效范围： 0 至 255

默认值： 80

注释 ssh 检查器默认配置不启用任何警报。

SSH 检查器规则

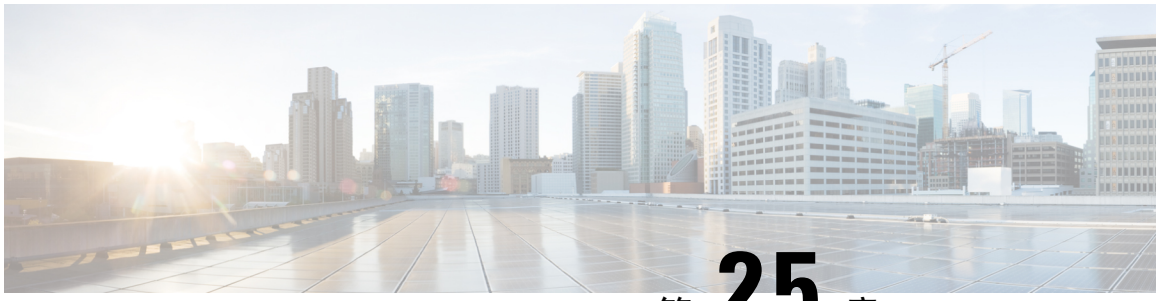
启用 ssh inspector rules to 生成事件并在内联部署中丢弃攻击性数据包。

表 *29: SSH* 检查器规则

GID:SID	Rule Message
128:1	质询-响应溢出攻击
128:2	SSH1 CRC32 攻击
128:3	服务器版本字符串溢出
128:5	错误消息方向
128:6	给定负载的负载大小不正确
128:7	SSH 版本字符串检测失败

SSH 检查器入侵规则选项

`ssh` 检查器没有任何入侵规则选项。



第 25 章

流 ICMP 检查器

流 ICMP 检查器概述

类型	检查器（流）
使用方式	检测
实例类型	多实例
所需其他检查器	无
已启用	`true`

互联网控制消息协议 (ICMP) 是网络实用程序应用程序和网络设备使用的网络层协议。ICMP 可发送诊断和错误信息，以确定 IP 主机之间的通信成功还是失败。ICMP 消息包括报头和数据部分。

ICMP 可传输有关其他数据流的信息。它不传输需要重组的数据，也不需要基于目标的绑定。

`stream_icmp` 检查器定义 ICMP 流跟踪。对于 ping，检查器通过 ICMP 信头中的源和目标 IP 地址字段以及端口字段提供基本流跟踪。对于无法访问的目标，检查器会分析原始 IP 地址和传输端口，然后更新会话的状态。`port_scan` 检查器可以使用无法访问的主机和端口（如果可用）。

配置流 ICMP 检查器的最佳实践

配置 `stream_icmp` 检查器时，请考虑以下最佳实践：

- 为要应用于主机或网络的每个会话超时创建 `stream_icmp` 检查器。 `stream_icmp` 检查器将 `session_timeout` 与 `绑定程序` 检查器中定义的 ICMP 主机或网络相关联。

您可以在同一网络分析策略 (NAP) 中拥有多个版本的 stream_icmp 检查器。

流 ICMP 检查器参数

session_timeout

指定 stream_icmp 检查器在状态表中保持非活动 ICMP 流的秒数。下次 Snort 检测到具有相同流密钥的 ICMP 数据报时，它会检查较早流上的会话超时是否已到期。如果超时已到期，Snort 将关闭流并启动新的流。Snort 检查与基本流配置关联的过时流。

类型： 整数

有效范围： 0 至 2,147,483,647（最大 31）

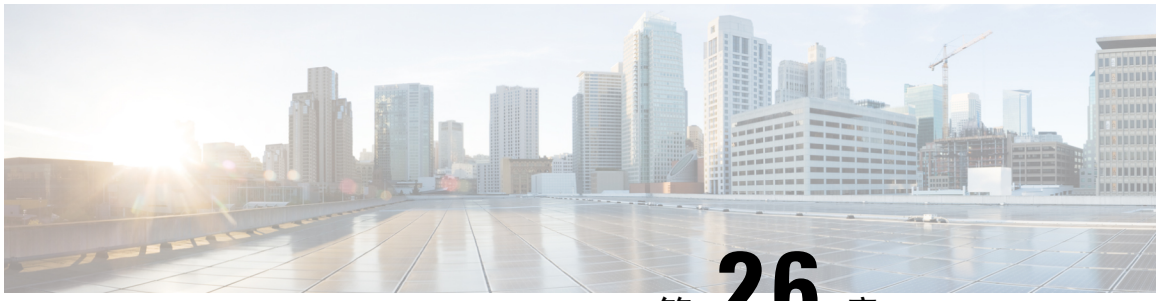
默认值： 60

流 ICMP 检查器规则

stream_icmp 检查器没有任何关联的规则。

流 ICMP 检查器入侵规则选项

stream_icmp 检查器没有任何入侵规则选项。



第 26 章

流 IP 检查器

流 IP 检查器概述

类型	检查器（流）
使用方式	检测
实例类型	多实例
所需其他检查器	无
已启用	`true`

互联网协议 (IP) 是构成互联网基础的无连接网络层协议。IP 使用主机地址将消息跨 IP 网络从源主机路由到目标主机。除其他传输协议外，IP 还可以路由 TCP 和 UDP 数据包。

IP 消息包含信头和数据部分。IP 信头包含用于将消息路由至其目的地的 IP 地址。IP 数据部分封装消息负载。IP 处理消息的重组和分段。

`stream_ip` 检查器可检测 IP 网络流并检查流中的数据包。 `stream_ip` 检查器定义 IP 会话和流跟踪、操作系统策略以及数据报重叠配置参数。根据模式， `stream_ip` 检查器或 Snort 数据平面会处理分片重组。

配置流 IP 检查器的最佳实践

配置 `stream_ip` 检查器时，请考虑以下最佳实践：

- 为要应用于主机、终端或网络的每个 IP 配置创建一个 stream_ip 检查器。流 IP 检查器将 IP 配置与 绑定程序 检查器中定义的 IP 主机、终端或网络相关联。

  您可以在同一网络分析策略中设置多个版本的 stream_ip 检查器。

流 IP 检查器参数

max_overlaps

指定每个数据报允许的最大重叠数。指定 0 以允许无限数量的重叠。

您可以启用规则 123:12 以触发分段重叠过多的警报。

类型： 整数

有效范围： 0 到 4,294,967,295（最大 32）

默认值： 0

min_frag_length

指定 IP 分段中预期的最小字节数。指定 0 以允许 IP 分段中包含无限数量的字节。

您可以启用规则 123:13，以便为短于 min_frag_length的分段触发警报。

类型： 整数

有效范围： 0 至 65535

默认值： 0

min_ttl

指定最小生存时间 (TTL) 或跳数。丢弃低于指定最小 TTL 的分片。

您可以启用规则 123:11，以便为 TTL 低于此值的分片触发警报。

类型： 整数

有效范围： 1 至 255

默认值： 1

策略

指定目标主机或主机的操作系统。操作系统可确定适当的 IP 分段重组策略和操作系统特征。只能为每个流 IP 检查器定义一个 策略 参数。

注释 如果将 策略 参数设置为 第一个，Snort 可能会提供一些保护，但会错过攻击。您应编辑 IP 流检查器的 policy 参数指定正确的操作系统。

类型： enum

有效值： 为 策略 参数设置一种操作系统类型。

表 *30:* 策略的有效值

策略	操作系统
第一	未知的操作系统
linux	Linux
bsd	AIX FreeBSD OpenBSD
bsd_right	HP JetDirect（打印机）
最后一个	思科 IOS
windows	Windows 98 Windows NT Windows 2000 Windows XP
solaris	Solaris OS SunOS

默认值： Linux

session_timeout

指定 stream_ip 检查器在状态表中保持非活动 IP 流的秒数。下次 Snort 检测到具有相同流密钥的 IP 数据报时，它会检查较早流上的会话超时是否已到期。如果超时已到期，Snort将关闭流并启动新的流。Snort 检查与基本流配置关联的过时流。

类型： 整数

有效范围： 0 至 2,147,483,647（最大 31）

默认值： 60

流 IP 检查器规则

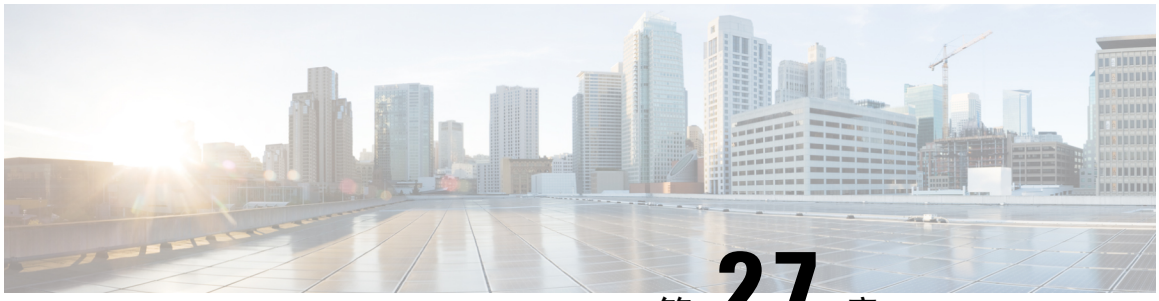
启用 stream_ip 检查器规则，以 生成事件并在内联部署中丢弃攻击性数据包。

表 *31:* 流 *IP* 检查器规则

GID:SID	Rule Message
123:1	分片数据包上的 IP 选项不一致
123:2	teardrop 攻击
123:3	短分段，可能的 DOS 尝试
123:4	分片数据包在分片重组数据包之后结束
123:5	零字节分片数据包
123:6	分片大小错误，数据包大小为负数
123:7	分片大小错误，数据包大小大于 65536
123:8	分段重叠
123:11	TTL 值小于配置的最小值，未用于重组
123:12	分段重叠过多
123:13	微小分片

流 IP 检查器入侵规则选项

`stream_ip` 检查器没有任何入侵规则选项。



第 27 章

流 TCP 检查器

流 TCP 检查器概述

类型	检查器（流）
使用方式	检测
实例类型	多实例
所需其他检查器	无
已启用	`true`

传输控制协议 (TCP) 是一种面向连接的状态传输层协议。TCP 能够通过 IP 网络在客户端和服务器之间可靠地传输有序的字节流。TCP 一次仅允许存在一个具有相同连接参数值的连接。主机操作系统管理 TCP 连接的状态。

`stream_tcp` 检查器提供 TCP 流跟踪、数据流规范化和数据流重组。每个流 TCP 检查器都可以处理网络中的一个或多个主机的 TCP 流量。此外，如果您有足够的有关向您的网络发送 TCP 流量的主机的信息，则可以为这些主机配置 `stream_tcp` 检查器。

在网络分析策略 (NAP) 中，Snort 将每个配置的 `stream_tcp` 检查器应用于 `绑定程序` 检查器配置中定义的 TCP 服务。

您可以配置多个流 TCP 检查器来处理各种操作系统和 TCP 流量。

`stream_tcp` 检查器配置包括：

- TCP 主机上的操作系统

- 操作系统选项：在重组期间如何处理重叠
- 流量处理选项：会话或方向中的最大字节数或分段数
- TCP 数据流重组选项：重组的最大 PDU 大小

注释 在内联 IPS 模式下， `stream_tcp` 检查器会规范化负载流，以便始终将重叠解析为所见的第一个副本。每个流 TCP 检查器处理重复的 SYN、RST 验证和时间戳检查。

配置流 TCP 检查器的最佳实践

在配置 `stream_tcp` 检查器时，请考虑以下最佳实践：

- 不要在设备上部署感应接口，以便 Snort 只能检查流量的一端。您可以启用 `stream_tcp` 检查器中的 `reassembly_async` 参数以处理非对称流量。但是，流 TCP 检查器不能在所有情况下处理非对称流量。例如，对 HTTP HEAD 请求的响应可能会导致 HTTP 检查器不同步。在 IDS 模式下，由于缺少 TCP 确认，因此更容易规避。

  对于 IPS 模式，我们建议您仅在 Snort 可以检查流量两端时才部署 设备。

- 为您希望发送或接收 TCP 流量的每个 TCP 主机操作系统创建一个 `stream_tcp` 检查器。您可以在同一网络分析策略中使用多个版本的 `stream_tcp` 检查器。每个 `stream_tcp` 检查器中定义的 TCP 策略都会应用于 `绑定程序` 检查器中指定的 TCP 主机。
- 要启用 IPS 模式，请将 `规范器` 检查器中的 `normalizer.tcp.ips` 参数设置为 `true`。
- 在网络分析策略 (NAP) 的高级设置中，确认要在自定义的基于目标的 `stream_tcp` 检查器中标识的网络是否匹配或是其父 NAP 处理的网络、区域和 VLAN 的子集。
- 系统会为每个枝叶域构建单独的网络映射。在多域部署中，使用文字 IP 地址限制此配置可能会出现意外结果。 通过使用支持覆盖的对象，后代域管理员可为其本地环境自定义全局配置。
- 要 生成事件并在内联部署中丢弃攻击性数据包，启用 `stream_tcp` 检查器规则 (GID 129)。

TCP 数据流重组最佳实践

`stream_tcp` 检查器收集和重组属于 TCP 会话的服务器到客户端通信数据流和/或客户端到服务器通信数据流的一部分的所有数据包。TCP 流重新组装允许 Snort 将流作为一个单独的、重新组装的实体、一个协议数据单元（PDU）来检查，而不是只检查作为给定流一部分的单个数据包。如果 PDU 很大，规则引擎会将其拆分为多个部分。

数据流重组允许 Snort 识别基于数据流的攻击，在检查个别数据包时它可能无法检测此类攻击。您可以根据网络需要指定要重新组装的通信流。例如，在监控网络服务器上的流量时，您可能只希望检查客户端流量，因为您不太可能从自己的网络服务器接收到恶意流量。

对于每个 stream_tcp 检查器，您可以在 绑定程序 配置中指定 TCP 端口列表。TCP 数据流检查器会自动且透明地包含已配置的端口，以识别和重组流量。启用自适应文件更新后，您可以列出用于识别要重组的流量的服务（以替代端口或端口组合的形式）。

在以下 Snort 检查器的 绑定程序 配置中指定 TCP 端口：

- dnp3
- ftp_server
- gtp_inspect （默认提供端口）
- http_inspect
- imap
- iec104 （默认情况下提供端口）
- mms （默认情况下提供的端口）
- Modbus （默认提供端口）
- 弹出
- sip
- smtp
- ssh
- ssl
- telnet

注释 重组多种流量类型（客户端、服务器或两者）时，Snort 资源需求可能会增加。

流 TCP 检查器参数

流 TCP 重组配置

绑定程序 检查器会为网络分析策略 (NAP) 定义 TCP 数据流重组配置。指定要对其应用 TCP 数据流重组策略的主机 IP 地址。流 TCP 检查器会自动与在 NAP 的 绑定程序 中配置的端口相关联。有关详细信息，请参阅绑定程序检查器概述， 第 13 页。

注释 系统会为每个枝叶域构建单独的网络映射。在多域部署中，使用文字 IP 地址限制此配置可能会出现意外结果。 通过使用支持覆盖的对象，后代域管理员可为其本地环境自定义全局配置。

默认策略中的 default 设置指定受监控网段上其他基于目标的策略未涵盖的所有 IP 地址。因此，不能且不需要为默认策略指定 IP 地址或 CIDR 块/前缀长度，并且不能在其他策略中将此设置留空或使用地址记法来表示 any（例如，0.0.0.0/0 或 ::/0）。

策略

指定目标主机或主机的操作系统。操作系统可确定适当的 TCP 重组策略和操作系统特征。只能为每个流 TCP 检查器定义一个 策略 参数。

注释 如果将 策略 参数设置为 第一个，Snort 可能会提供一些保护，但会错过攻击。您应编辑 TCP 数据流检查器的 策略 参数指定相应的操作系统。

类型： enum

有效值： 为 策略 参数设置一种操作系统类型。

表 *32: TCP* 操作系统策略

策略	操作系统
第一	未知的操作系统
最后一个	思科 IOS
bsd	AIX FreeBSD OpenBSD
hpux_10	HP-UX 10.2 及更高版本
hpux_11	HP-UX 11.0 及更高版本
irix	SGI Irix
linux	Linux 2.4 内核 Linux 2.6 内核
macos	Mac OS 10 (Mac OS X)
old_linux	Linux 2.2 及更低版本的内核
solaris	Solaris OS SunOS
vista	Windows Vista

策略	操作系统
windows	Windows 98 Windows NT Windows 2000 Windows XP
win_2003	Windows 2003

默认值： bsd

max_window

指定接收主机允许的最大 TCP 窗口大小。可以指定小于 65535 的整数，或指定 0 以禁用 TCP 窗口大小检测。

注意 max_window 的上限是 RFC 1323 允许的最大窗口大小。您可以设置上限以防止攻击者逃避检测，但是，如果将最大 TCP 窗口大小设置得过大，则可能会导致自身造成的拒绝服务。

类型： 整数

有效范围： 0 至 1,073,725,440

默认值： 0

overlap_limit

指定每个 TCP 会话中允许的重叠网段的最大数量。指定 0 以允许无限数量的分段重叠。如果设置介于 0 和 255之间的数字，则会话的网段重组将会停止。

对 生成事件并在内联部署中丢弃攻击性数据包启用规则 129:7。

类型： 整数

有效范围： 0 到 4,294,967,295（最大 32）

默认值： 0

max_pdu

指定最大重组协议数据单元 (PDU) 大小。

类型： 整数

有效范围： 1460 至 32768

默认值： 16384

reassemble_async

确保在两个方向上看到流量之前，数据已排队等待重组。如果受监控的网络为异步网络，则必须启用 `reassembly_async` 参数。异步网络一次仅允许单个方向和一个流上的流量。如果启用了 `reassembly_async` 参数，则 Snort 在提高性能时不会重组 TCP 流。

注释 流 TCP 检查器在所有情况下都无法正确处理非对称流量。例如，对 HTTP HEAD 请求的响应可能会导致 HTTP 检查器不同步。在 IDS 模式下，由于缺少 TCP 确认，因此更容易规避。对于 IPS 模式，我们建议您仅在规则引擎可以检查流量的两端时才部署 设备。

Cisco Secure Firewall Threat Defense 路由和透明接口的 `reassembly_async` 参数将被忽略。

类型： boolean

有效值： `true`、 `false`

默认值： `true`

require_3whs

指定流 TCP 检查器停止跟踪中间会话之前从启动经过的秒数。指定 `-1` 以跟踪所有中间 TCP 会话，无论这些会话何时发生。

Snort 不会同步大多数协议流。如果 Snort 需要任何握手选项（时间戳、窗口缩放或 MSS），则始终接收 SYN。通常，允许中流代扣器不会提高 IPS 效率。

类型： 整数

有效范围： `-1` 到 `2,147,483,647`（最大 31）

默认值： `-1`

queue_limit.max_bytes

指定 TCP 连接一端的会话排队的最大字节数。指定 `0` 以允许无限个字节。

注意 我们建议您联系思科 TAC，然后再更改 `queue_limit.max_bytes` 参数的默认设置。

类型： 整数

有效范围： `0` 到 `4,294,967,295`（最大 32）

默认值： `4,194,304`

queue_limit.max_segments

指定在 TCP 连接的一端等待会话排队的数据分段的最大数。指定 `0` 以允许无限数量的数据段。

注意 我们建议您联系思科 TAC，然后再更改 `queue_limit.max_segments` 参数的默认设置。

类型： 整数

有效范围： `0` 到 `4,294,967,295`（最大 `32`）

默认值： `3072`

small_segments.count

指定大于连续小 TCP 分段的预期数量的数字。指定 `0` 以忽略连续的小 TCP 分段的计数。

您必须使用相同类型的值设置 `small_segments.count` 和 `small_segments.maximum_size` 参数。为两个参数指定 `0` 或将每个参数设置为非零值。

注释 即使每个网段长度为 `1` 字节，Snort 也会考虑 2000 个连续网段，这超出了连续 TCP 网段的正常数量。

Snort 会忽略 威胁防御 路由和透明接口的 `small_segments.count` 参数。

您可以对 生成事件并在内联部署中丢弃攻击性数据包启用规则 129:12。

类型： 整数

有效范围： `0` 至 `2048`

默认值： `0`

small_segments.maximum_size

指定将 TCP 分段标识为大于小 TCP 分段的字节数。小 TCP 分段大小的范围为 `1` 至 `2048` 字节。指定 `0` 以忽略小数据段的最大大小。

Snort 会忽略 威胁防御 路由和透明接口的 `small_segments.maximum_size` 参数。

您必须为 `small_segments.maximum_size` 和 `small_segments.count` 参数设置相同类型的值。为两个参数指定 `0` 或将每个参数设置为非零值。

注释 `2048` 字节的 TCP 分段大于普通的 `1500` 字节的以太网帧。

您可以对 生成事件并在内联部署中丢弃攻击性数据包启用规则 129:12。

类型： 整数

有效范围： `0` 至 `2048`

默认值： `0`

session_timeout

指定 Snort 在其状态表中保持非活动 TCP 流的秒数。如果数据流在指定时间内未重组，则 Snort 会将其从状态表中删除。如果会话仍处于活动状态并且显示更多数据包，Snort 会将该流作为中游流处理。

我们建议将 `session_timeout` 参数设置为大于或等于主机 TCP 会话超时。

类型： 整数

有效范围： `0` 至 `2,147,483,647`（最大 `31`）

默认值： `180`

流 TCP 检查器规则

启用 `stream_tcp` 检查器规则，以 生成事件并在内联部署中丢弃攻击性数据包。

表 *33*: 流 *TCP* 检查器规则

GID:SID	Rule Message
129:1	已建立会话的 SYN
129:2	SYN 数据包上的数据
129:3	在流上发送的数据不接受数据
129:4	TCP 时间戳位于 PAWS 窗口之外
129:5	错误的分段，调整后的大小小于或等于 0（已弃用）
129:6	窗口大小（扩展后）大于策略允许的大小
129:7	达到重叠 TCP 数据包数量限制
129:8	TCP 重置发送后，在流中发送的数据
129:9	TCP 客户端可能被劫持，以太网地址不同
129:10	TCP 服务器可能被劫持，不同的以太网地址
129:11	未设置 TCP 标志的 TCP 数据
129:12	超过阈值的连续 TCP 小分段
129:13	检测到 4 次握手
129:14	缺少 TCP 时间戳
129:15	在窗口外部重置

GID:SID	Rule Message
129:16	FIN 编号大于上一个 FIN
129:17	ACK 编号大于先前 FIN
129:18	收到 TCP 重置后在流中发送的数据
129:19	TCP 窗口在接收数据前已关闭
129:20	不带三次握手的 TCP 会话

流 TCP 检查器入侵规则选项

stream_reassemble

指定是否对匹配流量启用 TCP 数据流重组。`stream_reassemble` 规则选项包含四个参数：`stream_reassemble.action`、`stream_reassemble.direction`、`stream_reassemble.noalert`和`stream_reassemble.fastpath`。

语法： `stream_reassemble: <enable|disable>, <server|client|both>, noalert, fastpath;`

示例： `stream_reassembly: disable,client,noalert;`

stream_reassemble.action

停止或开始流重组。

类型： enum

语法： `stream_reassemble: <action>;`

有效值： 禁用 或 启用

示例： `stream_reassembly: enable;`

stream_reassembly.direction

操作适用于给定的方向。

类型： enum

语法： `stream_reassemble: <direction>`

有效值： `client`、`server`、`both`

示例： `stream_reassembly: Both;`

stream_reassembly.noalert

规则匹配时不提醒。`stream_reassembly.noalert` 参数是可选的。

语法： `stream_reassembly: noalert;`

示例： `stream_reassembly: noalert;`

stream_reassembly.fastpath

（可选）信任会话的其余部分。`stream_reassembly.fastpath` 参数是可选的。

语法： `stream_reassembly: fastpath;`

示例： `stream_reassembly: fastpath;`

stream_size

用于检查流大小的检测选项。允许规则根据观察到的字节数（由 TCP 序列号确定）匹配流量。`stream_size` 规则选项包括两个参数：`stream_size.direction` 和 `stream_size.range`。

语法： `stream_size: <server|client|both|either>, <operator><number>;`

示例： `stream_size: client, <6;`

stream_size.direction

比较适用于流的方向。

类型： enum

语法： `stream_size: <direction>;`

有效值：

- `either`
- `to_server`
- `to_client`
- `both`

示例： `stream_size: to_client;`

stream_size.range

检查数据流大小是否在指定范围内。指定 `范围` 运算符和一个或多个正整数。

类型： 间隔

语法： `stream_size: <range_operator><positive integer>;` or `stream_size: <positive integer><range_operator><positive integer>;`

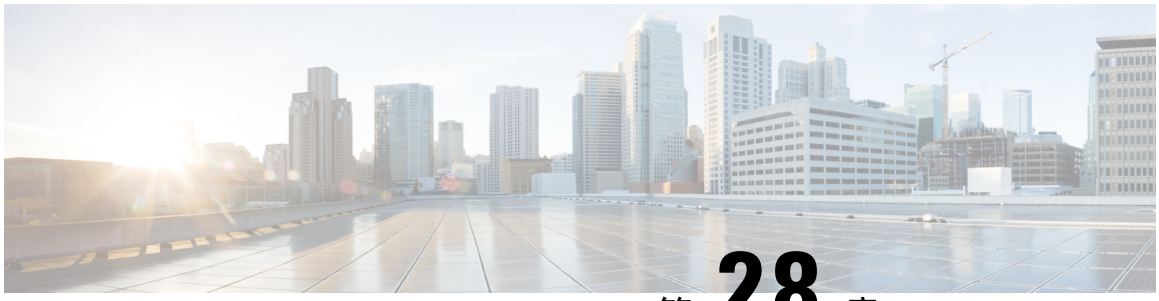
有效值： 一组一个或多个正整数以及 表 34: 范围格式 中指定的一个 `range_operator`。

示例： `stream_size: +6;`

表 *34:* 范围格式

范围格式	Operator	说明
operator i		

范围格式	Operator	说明
	<	少于
	>	大于
	=	平分
	≠	不等于
	⩽	小于或等于
	⩾	大于或等于
j operator k		
	<>	大于 j 且小于 k
	<=>	大于或等于 j 且小于或等于 k



第 28 章

流 UDP 检查器

流 UDP 检查器概述

类型	检查器（流）
使用方式	检测
实例类型	多实例
所需其他检查器	无
已启用	`true`

用户数据报协议 (UDP) 是一种无连接、低延迟的传输层协议。在接收方提供协议之前，UDP 可实现两个网络终端之间的无状态通信。为评估消息报头和数据的完整性，UDP 使用校验和。

`stream_udp` 检查器检查 IP 数据报报头中的源和目标 IP 地址字段及 UDP 报头中的端口字段来确定流动方向并识别会话。当超过可配置的计时器，或者当任何一个端点接收到另一个端点不可达的ICMP消息时，会话结束。

UDP 数据流检查器不会生成事件。您可以启用数据包解码器规则 (GID 116) 来检测 UDP 信头异常。

配置流 UDP 检查器的最佳实践

配置 `stream_udp` 检查器时，请考虑以下最佳实践：

- 为要应用于主机或终端的每个会话超时创建 `stream_udp` 检查器。数据流 UDP 检查器将 `session_timeout` 与 绑定程序 检查器中定义的 UDP 主机相关联。

在同一网络分析策略中，您可以有多个版本的 stream_udp 检查器。

- 启用数据包解码器规则 (GID 116) 以检测 UDP 报头异常。

流 UDP 检查器参数

session_timeout

指定 UDP 检查器在状态表中保持非活动 UDP 流的秒数。下次 Snort 检测到具有相同流密钥的 UDP 数据报时，它会检查较早流上的会话超时是否已到期。如果超时已到期，Snort将关闭流并启动新的流。Snort 检查与基本流配置关联的过时流。

类型： 整数

有效范围： 0 至 2,147,483,647（最大 31）

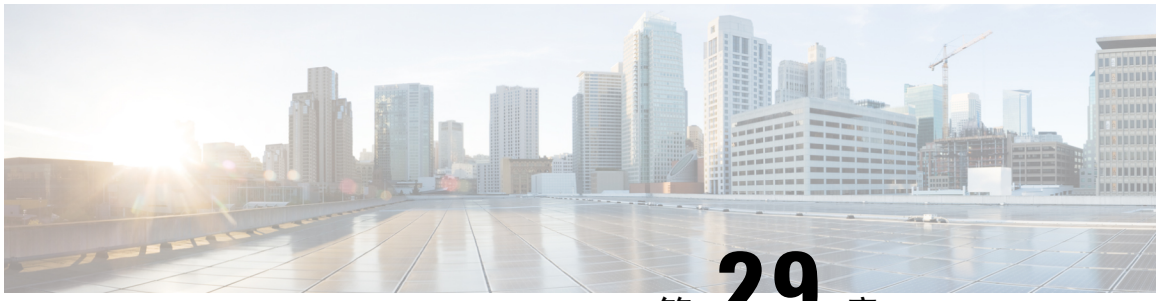
默认值： 30

流 UDP 检查器规则

stream_udp 检查器没有任何关联规则。

流 UDP 检查器入侵规则选项

stream_udp 检查器没有任何入侵规则选项。



第 29 章

Telnet 检查器

Telnet 检查器概述

类型	检查器（服务）
使用方式	检测
实例类型	多实例
所需其他检查器	`stream_tcp`
已启用	`false`

Telnet 是一种应用层协议，可在 TCP 上创建 8 位字节通信通道。Telnet 使用网络虚拟终端在客户端和远程主机之间进行通信。Telnet 服务器使用 TCP 端口 23.

`telnet` 检查器通过检测 Telnet 命令序列和选项协商将 Telnet 数据缓冲区规范化。`telnet` 检查器会消除数据包中的 Telnet 命令序列 (RFC 854)。`Telnet` 检查器通过检查 Telnet 加密选项 (RFC 2946) 来检测加密 Telnet 连接。

Telnet 检查器参数

TeInt 服务配置

绑定程序 检查器定义 Telnet 服务 配置。有关详细信息，请参阅绑定程序检查器概述， 第 13 页。

示例：

```
[
    {
        "when": {
            "service": "telnet",
            "role": any
        },
        "use": {
            "type": "telnet"
        }
    }
]
```

ayt_attack_thresh

指定Are You There (AYT) telnet 命令的最大连续次数。 `Telnet` 检查器检测到超过 `ayt_attack_thresh` 值的连续 AYT 命令数，并发出警报。 `ayt_attack_thresh` 参数修复与 telnet 的 BSD 实施相关的特定漏洞。指定 `-1` 以禁用 `ayt_attack_thresh` 参数。您可以为此参数启用规则 126:1 到 生成事件并在内联部署中丢弃攻击性数据包 。

类型： 整数

有效范围： `-1` 到 `2,147,483,647`（最大 31）

默认值： `-1`

encrypted_traffic

指定是否检测加密的 Telnet 流量。您可以为此参数启用规则 126:2 更改为 生成事件并在内联部署中丢弃攻击性数据包 。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

规范化

指定是否规范 Telnet 流量。`Telnet` 检查器通过消除 Telnet 转义序列来规范化 Telnet 流量。如果已启用的入侵规则指定 `原始` 内容参数，则该规则将忽略 `telnet` 检查器创建的规范化 telnet 缓冲区。

类型： boolean

有效值： `true`、 `false`

默认值： `false`

Telnet 检查器规则

启用 `telnet` 检查器 生成事件并在内联部署中丢弃攻击性数据包。

表 *35: Telnet* 检查器规则

GID:SID	Rule Message
126:1	连续 Telnet AYT 命令超出阈值
126:2	Telnet 流量已加密
126:3	不带子协商结束的 Telnet 子协商开始命令

Telnet 检查器入侵规则选项

`Telnet` 检查器没有任何入侵规则选项。

当地语言翻译版本说明

思科可能会在某些地方提供本内容的当地语言翻译版本。请注意，翻译版本仅供参考，如有任何不一致之处，以本内容的英文版本为准。