部局単位でトラブル発生原因の特定を容易にする簡易 SOC の構築

井田 敦也 [1], 奥田 剛 [1]

1) 大阪大学大学院工学研究科

ida@eng.osaka-u.ac.jp

Constructing Security Operation Center to Easily Identify the Cause of Network Troubles at a Department Level

Atsuya Ida [1], Takeshi Okuda[1]

1) Graduate School of Engineering, Osaka University

概要

大阪大学大学院工学研究科で管理しているグローバル IPv4 アドレスの総数は大阪大学全体の約 5 分の 1 を占める。これらの利用と管理は専攻等の教職員に一任されており、研究室によっては NAT ルータを利用しているところがある。この場合、通信障害やセキュリティインシデントなどのトラブル発生時にログが利用できず、原因の特定が困難である。そこで、研究室のルータから取得した NAT ログや ARP ログなどを収集し、必要な情報に素早くアクセスするためのダッシュボードを備えた簡易 SOC を構築した。この結果、トラブル発生時の原因の特定が容易になった。

1 はじめに

大阪大学大学院工学研究科（以下、工学研究科）の組織は、9 専攻、5 学科、6 附属教育研究施設、12 共通施設、事務部、技術部からなる[1]。構成員として教職員 613 名、学生約 6,000 名が工学研究科に所属している[2]。

これら多数の構成員の研究教育活動を支える工学研究科のネットワークは、全学のネットワークである大阪大学総合情報通信システム（ODINS: Osaka University Information Network System）の一部であり、2021 年時点で 159 台のスイッチで構成されている[3]。工学研究科で管理しているグローバル IPv4 アドレスの第 3 オクテットは 50 におよぶ。専攻や施設毎に選出される情報ネットワーク運用委員と情報ネットワーク部門が、教職員と連携することで工学研究科のネットワークを運営している。

工学研究科のグローバル IPv4 アドレスを付与した端末や機器の管理は情報ネットワーク運用委員と教職員に一任されており、研究室によってはルータに割り当てて NAT を利用しているケースがある。しかし、研究室独自でルータを運用している場合、その通信ログが適切に保存されておらず、保存されていても保存期間が短い事や見方が分からない事があり、ネットワークトラブル発生時などに活かせないという問題がある。解決策として、RFC6888[4]に基づくシェアドアドレススペースを使ったキャリアグレード NAT の導入を進めている。この環境では通信ログを一元管理しているが、研究室内のアドレス構成を変更する必要があるため、普及には時間がかかる。

そこで、既に研究室で運用されているルータから NAT ログや ARP ログなどを当部門で集約し、ネットワークトラブルやセキュリティインシデント発生時の原因特定に役立つ簡易 SOC サービスを提供する。

2 工学研究科ネットワークでの問題と解決策

工学研究科のネットワークは、キャンパスネットワークの ODINS ルータを中心に、スター型に構成されており、各棟・各階に設置された L2 スイッチで約 170 の研究室の機器を収容している。スイッチに接続されている機器の多くは PC であるが、研究室によっては複数のルータを接続し、内部ネットワークに PC などを接続している。さらに、ルータの通信ログは保存されていないことが多い。そのため、NAT 内部でのトラブルは研究室内でのみ観測可能であり、そのことが障害原因の特定や

セキュリティインシデント発生時の個体特定に時間がかかる要因となっている。研究室によってはNAT 内部に 100 台近くの PC が存在し、トラブル発生時には解決までに教員の時間を消費することになる。

これらの問題を解決するため、研究室で運用されているネットワーク機器のログを集約し、ネットワークのトラブル発生時に迅速に原因特定が行えるように統合ログ管理の機能を提供する。これを簡易 SOC と呼称している。簡易 SOC では、オープンソースの Elastic Stack を利用した。Elastic Stack は、全文検索エンジンの Elasticsearch、web インターフェイスの Kibana、ログ処理機能の Logstash などからなる、可視化・分析プラットフォームで、無料で利用することが可能である [5,6,7,8]。

3 簡易 SOC の概要

簡易 SOC では、各専攻や研究室で運用されているネットワーク機器のログを集約し、Elastic Stack による検索機能や web インターフェイスを提供する。従前であれば、ログのフォーマットを熟知し、コマンドラインに習熟する必要があったが、簡易 SOC を用いることで、マウス等によるポイントアンドクリックで検索条件の固定や検索範囲の指定ができ、目的のログ情報を容易に抽出することが可能となる。

簡易 SOC を構築するにあたり、まずは安価な PC 3 台を用いて Elastic Stack の各機能を実装し、利用状況に合わせて PC の台数を増やすように構成している。

3.1 簡易 SOC 用 PC の用意

簡易 SOC を実現する PC の要件として、CPU やメモリ、SSD の性能が良く、省スペースかつ取り替え可能で、1 台 10 万円未満であることを設定した。検討の結果、HP ProDesk 400 G5 DM/CT を 3 台用意した。PC にインストールされている Windows10 は Elastic Stack のサポート OS に含まれていないため、CentOS8 に入れ替えている[9]。CentOS8 のインストール作業は手動で行なっている。

これら 3 台に Java と Elasticsearch をインストールし、クラスタを構成している。3 台のうち 1 台に Logstash、別の 1 台に Kibana をインストールしている。

3.2 Ansible の利用

OS のインストール作業以降は、Elastic Stack やリポジトリといった設定ファイルの準備や編集、パッケージのインストールといった作業の簡略化、設定内容の統一化を行うために、Ansible を利用している [10]。Ansible が使える環境であれば、作業内容をプロジェクト毎にフォルダとして用意し引き継ぐ事が可能である。また、一度 playbook を用意すれば、簡易 SOC への PC の追加も容易に行える。

```
---
- name: Just force systemd to reload configs
  systemd:
    daemon_reload: yes
- name: Enable service elasticsearch and ensure it
is not masked
  systemd:
    name: elasticsearch
    enabled: yes
    masked: no
- name: Make sure a service is running
  systemd:
    state: started
    name: elasticsearch
```

図 1 ansible playbook の例

図 1 ではデーモンの再読み込みを行い、Elasticsearch のサービスを登録し起動するように記述している。

Java や Elastic Stack のインストール、Elastic Stack のリポジトリ設定、systemctl への登録、yml ファイルの記述を Ansible で対応できるようにした。

3.3 簡易 SOC 用 PC の追加

簡易 SOC の容量を拡張するため、3.1 の要件を満たす、HP ProDesk 400 G6 DM を 3 台追加した。クラスタ化するためには、Elasticsearch の設定ファイルで cluster.name を統一し、discovery.seed_hosts に既存の簡易 SOC 用 PC の IP アドレスを入力する必要がある。そして、ELK(Elasticsearch, Logstash, Kibana) はいずれの PC においても同じバージョンに統一しなければクラスタとして機能しないことに注意が必要である。

4　ログ情報の収集と解析、抽出

4.1 Syslog 転送機能の利用

研究室で運用されているネットワーク機器からのログは、syslog 機能を用いて、転送用ルータに転送され、そこからさらに簡易 SOC 内の Logstash に転送される（図２）。

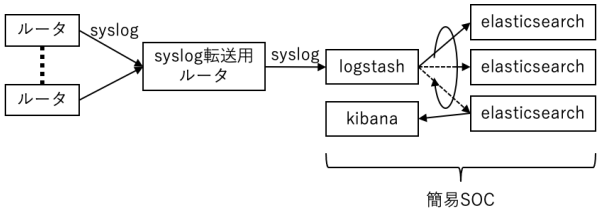

図２ 簡易 SOC への Syslog 転送

4.2 Logstash での Syslog 解析、抽出

Logstash は、転送されてきた Syslog メッセージを Elasticsearch に格納する。何も加工しない状態であれば１つのフィールド「Message」に収められ、ID やタイムスタンプなどのフィールドが付与される。Elasticsearch で検索する際には「Message」に収められたデータの中を検索するため、格納されたデータ数だけ検索時間がかかる。検索時間短縮と、範囲検索などのために、転送されてきた syslog メッセージを解析して IP アドレスやポート番号などを抽出し、個別のフィールドに代入する処理パイプラインを設定した（図３）。

```
   input {
     syslog {
       host => Logstash のIP アドレス
       port => Logstash のポート番号
     }
   }
   filter {
     if [program] == "ARP" {
       grok {
         match => { "message" =>
"%{IP:inside_local_address}
*%{MAC:inside_local_mac_address}" }
       }
     } else if [message] =~ "¥[NAT" {
       mutate { add_field => { "program" =>
"NAT" } }
       grok {
         match => { "message" =>
"%{IP:inside_global_address}¥.%{POSINT:inside_
global_port}
<-> %{IP:inside_local_address}¥.%{POSINT:inside
_local_port}
==> %{IP:outside_global_address}¥.%{POSINT:out
side_global_port}" }
       }
     } else if [message] =~ "¥[DHCPD¥]" {
       mutate { add_field => { "program" =>
"DHCP" } }
       grok {
         match => { "message" =>
"%{IP:inside_local_address}:
*%{MAC:inside_local_mac_address}" }
       }
     }

     date {
       match => [ "syslog_timestamp", "MMM  d
HH:mm:ss", "MMM dd HH:mm:ss" ]
     }
   }
   output {
     elasticsearch {
       hosts => elasticsearch のIP アドレス(台数
分記入)
       index => "syslog%{+YYYY.MM.dd}"
     }
   }
```

図３ 処理用パイプライン定義ファイル

図３の Input は入力先の IP アドレスとポート番号の指定、Filter は入力されてきたデータの加工条件とフィールドへの代入、Output は出力先の指定とインデックスの命名を行っている。Filter の箇所では、例として NAT ログから IP アドレスやポート番号をフィールドに代入している。インデックスを検索する時に IP アドレスやポート番号を指定できることで検索しやすくしている。

5　インデックスの可視化

インデックスの中にあるログ情報の可視化や検索のための web インターフェイスとして Kibana を用いる。標準の Discover でフィールドを指定しドキュメントを解析することは可能だが、高度な検索を行うためには構文を知る必要がある。送信元や送信先の IP アドレスやポート番号のような情報を可視化するための方法として、図４のようなダッシュボードを作成した。ダッシュボードには、Kibana に搭載されているビジュアライゼーションを複数組み合わせている。

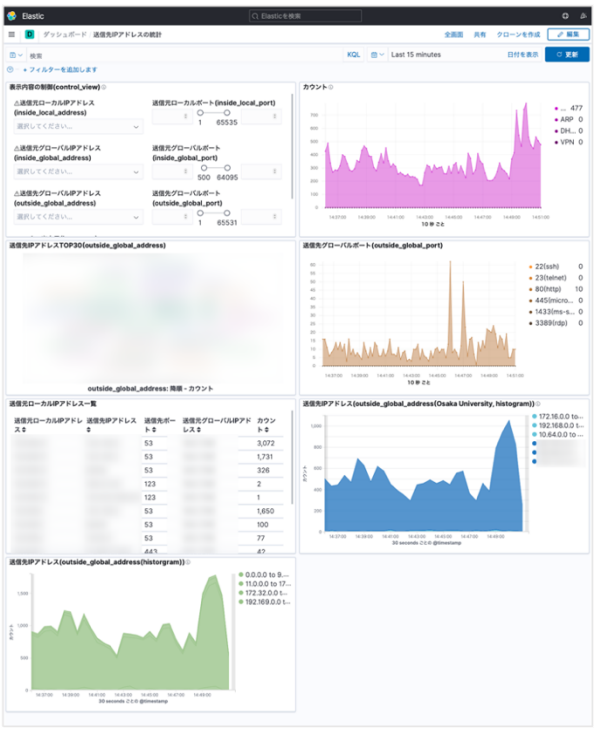

図 4　送信先 IP アドレスの統計

図 4 の内容は、送信元や送信先の IPv4 アドレスや ARP、DHCP などのプロトコル、大阪大学内外の IPv4 アドレスといった全体の統計情報を表している。図中左上の「表示内容の制御」で特定の IPv4 アドレスやポート番号を指定すると、内容に応じて他の統計情報も変更される。例えば、トラブル発生時に IP アドレスと時刻を指定して検索することで、外部の IP アドレスに対し SSH やリモートデスクトップを行っている端末の特定や、MAC アドレスの特定といったことも行える。これまでに、IP アドレスが重複した端末の MAC アドレスの特定や、セキュリティインシデントに関係する端末の特定に有用であった。

6　おわりに

本稿では、大阪大学大学院工学研究科での簡易 SOC の構築に関し、安価な PC と Elastic Stack を組み合わせた方法について紹介した。

今後は、Elasticsearch で蓄積したインデックスの IPv4 アドレスと大阪大学で通信遮断中の IPv4 アドレスを組み合わせて、マルウェアや C2 サーバとの通信を検出する機能の追加などを考えている。

参考文献

[1] 大阪大学　大学院工学研究科・工学部　要覧、https://www.eng.osaka-u.ac.jp/wp-content/uploads/pdf/outline/publicity/outline2021.pdf、p.09-10、2021.

[2] 大阪大学プロフィール、https://www.osaka-u.ac.jp/ja/guide/about/profile/profile2021、p.20-21、2021.

[3] ODINS とは、https://www.odins.osaka-u.ac.jp/contact-us/、20210908 にアクセス確認

[4] RFC6888、https://datatracker.ietf.org/doc/html/rfc6888、20210928 にアクセス確認

[5] Elastic Stack and Product Documentation、https://www.elastic.co/guide/en/elastic-stack/current/overview.html、20210928

[6] What is Elasticsearch?、https://www.elastic.co/guide/en/elasticsearch/reference/7.x/elasticsearch-intro.html

[7] Kibana—your window into Elastic、https://www.elastic.co/guide/en/kibana/7.x/introduction.html

[8] Logstash Introduction、https://www.elastic.co/guide/en/logstash/7.x/introduction.html

[9] Support Matrix、https://www.elastic.co/jp/support/matrix、20210928 にアクセス確認

[10] Ansible Documentation、https://docs.ansible.com/ansible/latest/index.htm、20210928 にアクセス確認