論　文

熊本高等専門学校八代キャンパスにおける侵入検知システムの構築と運用

小島　俊輔[1,*]　藤本　洋一[1]　岩本　舞[2]

Construction and Operation of Intrusion Detection System at National Institute of Technology, Kumamoto College, Yatsushiro Campus

Shunsuke Oshima[1,*], Yoichi Fujimoto[1], Mai Iwamoto[2]

In order to detect cyber security threat at NIT Kumamoto, Yatsushiro Campus, we construct and operate an Intrusion Detection System (IDS) in our computer network since December, 2017. IDS has based on a predetermined signature and it can detect intrusions, attacks and other signs by monitoring packet-flow through the computer network. In this paper, we report introduction, construction and operation of this system.

キーワード：侵入検知、ネットワークインフラ、サイバーセキュリティ、Snort、Security Onion
Keywords：Intrusion Detection, Network Infrastructure, Cyber Security, Snort, Security Onion

１．まえがき

熊本高等専門学校八代キャンパスでは、学内ネットワークにおけるサイバーセキュリティの脅威を検出するため、2017 年度より侵入検知システム（Intrusion Detection System、以後 IDS と略）の 1 種である Snort(1)を導入している。また、2018 年度より Snort を基幹として、セキュリティイベントの可視化やログの収集・検索機能など多くのネットワークセキュリティ関係ツールを集約した Linux ディストリビューションである Security Onion(2)を導入し実績を挙げている。本稿では、熊本高専内のネットワークにおけるこれら侵入検知システムの具体的な導入と構築について紹介するとともに、実際に IDS を運用したことで得られた知見についても併せて記載しておく。

２．IDS

2.1 IDS の分類

IDS とは、ネットワークを流れるパケットやホストのログなどを監視し、あらかじめ決められたあるパターンにマッチする特徴を検出することで不正侵入を検知するシステムである。有名な IDS としては Snort や Suricata(3)が挙げられる。

IDS は、検知する場所によって大きく 2 つに分類することができる。一つは、ホストコンピュータなどに常駐し、リソースやログを監視するホストベース型、もう一つは、ネットワークを流れるパケットを監視し、異常を検知するネットワーク型である。

今回導入する IDS にはネットワーク型を選択する。ネットワーク型とする理由は、1) OS タイプが Windows, Linux, MacOS, Android の場合はマルウェア対策専用のソフトウェアが存在するが、プリンタ、スキャナ、マイコンや各種 IoT 機器では専用のマルウェア対策ソフトウェアがなく、これら機器への対策が急務であること、2) BYOD (Bring Your Own Device) によるデバイスの持ち込みや個人所有の USB メモリの接続により、マルウェア対策が十分ではない機器から不正なコードが侵入してくる可能性があること、が挙げられる。

ネットワーク型 IDS であれば、OS やデバイスに依存せず、デバイスが不正アクセスを受けたり、デバイスから他のネットワーク機器に対する異常な動きがあった場合に、早期に検知できる可能性が高い。

2.2 Snort の概要

Snort はオープンソースのネットワーク型 IDS である。1998 年に開発され Sourcefire 社からオープンソースとして配布されていたが、2013 年に Cisco Systems 社が買収、その後もオープンソースとして Snort 3 が配布されている。

[1] 拠点化プロジェクト系（情報セキュリティセンター）
〒866-8501　熊本県八代市平山新町 2627
Center for Information Security,
2627 Hirayama-Shinmachi, Yatsushiro-shi, Kumamoto, Japan 866-8501
[2] 技術・教育支援センター
〒866-8501　熊本県八代市平山新町 2627
Center for Technical and Educational Support,
2627 Hirayama-Shinmachi, Yatsushiro-shi, Kumamoto, Japan 866-8501
* Corresponding author:
E-mail address: oshima@kumamoto-nct.ac.jp (S. Oshima).

```
書式：action proto srcIP srcPort -> dstIP dstPort
      (keyword:arg; [keyword:arg; ...])
例：  alert tcp any any -> 10.1.2.3 22
      (flags: S; msg: "SSH Connection attempt";)
```

図 1 Snort ルールの書式と例

Snort はパケットを常時監視し、Snort ルールと呼ばれる、あらかじめ定義したパターンにマッチする違反パケットを検出した場合に、警告などのアクションを実行する。図 1 に Snort ルールの簡単な例を示す。このルールはユーザが個別に記述し、組織独自のルールセットを作成することができるが、ルールセットが侵入検知の性能に直結するため、通常は無料の Community ルールセットを用いるか Personal や Business といった Cisco Systems 社が提供する有料のプランを選択し、組織ごとのカスタマイズを加える。

Snort ルールを最新に保つための仕組みとして、Pulled Pork と呼ばれるアップデータ、および Snort ルールをダウンロードする際の認証コードである Oinkcode が必要となる。いずれも専用サイトから取得することができる。サイトから取得した Oinkcode は Pulled Pork の設定ファイルに埋め込むか、後述する Security Onion のインストール時に指定する。

2.3 Security Onion の概要

Security Onion は Ubuntu Linux を基本 OS とし、その上に侵入検知やパケット解析、ログの蓄積・可視化など多くのセキュリティツールを一つのパッケージにした Linux ディストリビューションである。最新バージョンは Ubuntu 16.04 LTS をベースとする Security Onion 16.04.6.1 (2019/8/1 現在) であり、本家の Ubuntu 同様、2021 年 4 月までサポートされる。OS の ISO イメージが配布されておりメニューに従ってインストールすることができるほか、Linux をインストールした後、Security Onion のパッケージを展開して構築することもできる。ただし、パッケージ展開で正式にサポートされる OS は Ubuntu16.04 のみである。

Security Onion は、インストール時に Snort または Suricata のどちらか一方を IDS として選択することができる。IDS 以外に以下のようなツールが標準でインストールされる。

Erasticsearch[4]	全文検索エンジン
Logstash	ログイベントの監視と保管
Kibana	ブラウザベースのログ可視化ツール
Sguil/ Squert[5]	データ分析ツール
Bro[6]	振る舞い指向フォレンジックツール
Wireshark[7]	パケット指向フォレンジックツール
netsniff-ng[8]	任意パケット生成ツール

これらのツールは単独で起動して使用する場合もあるが、Sguil や Squert などのデータ分析ツールを入り口として、相互に連携し補完することで様々なサイバーセキュリティの脅威をシームレスに解析できるよう設計されている。

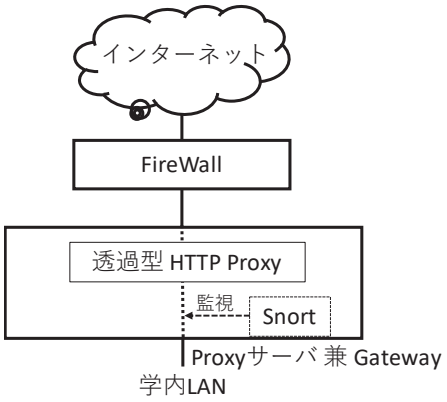


図 2 Snort の設置場所と全体の構成

3．Snort の導入と構築

3.1 Snort 導入時の検討事項

熊本高専八代キャンパスでは、2017 年度に学内外の通信の異常を検知する目的で、Proxy サーバを通過するパケットの監視をする Snort を導入した。導入に際しては、Snort の Business プランの費用を 2017 年度の予算で確保している。Snort はネットワーク型 IDS であり、センサノードを複数個所に設置することもできるが、監視対象の数が多いとランニングコストも高額になるので、今回は熊本高専八代キャンパスの Proxy サーバ 1 台のみに設置した。

次に必要となるのは Snort を稼働するサーバである。先の Proxy サーバは HTTP Proxy 以外に、学内（LAN）と学外（WAN）を接続するゲートウェイとして動作しており、学内外の全パケットが通過する。2017 年度はサーバ費用が捻出できなかったため、Snort を別サーバではなく Proxy サーバ上に搭載することとした。全体の構成を図 2 に示す。この図からもわかるように、Snort 側の設定は監視対象のネットワークインタフェースを指定する方式であり、Proxy サーバ内で構築することで Proxy や Default Gateway の設定変更は必要ない。Default Gateway の設定変更が必要ないということは、たとえば Snort の利用率が上昇し、サーバが高負荷になったことで物理的なパケット遅延やドロップが発生した場合でも、Snort プロセスを緊急停止すればすぐ元に戻るという大きなメリットがある。

3.2 サーバスペックの決定

Proxy サーバは、Intel Xeon 4Core 3.0GHz の Linux をホスト OS とする、2CPU 12000BogoMips、 Mem4GB の KVM 仮想マシンとして構築している。日中の通信量はおよそ 100Gbps であり、Proxy のみの状態で CPU 負荷は数パーセントである。この Proxy サーバ内に Snort を導入すると CPU 負荷が高くなることが予想されるが、Proxy サーバは仮想マシンであり、高負荷の際には仮想マシンのスペックを上げることを検討すればよい。

結果、Snort を稼働した際の CPU 負荷は 10～30%程度となり、外部との接続性に影響を及ぼすトラブルは確認できなかった。そのため、今回は仮想マシンのスペックをそのまま変更することなく本稼働に移行することができた。

12/18-11:27:57.684574 [**] [129:12:1] Consecutive TCP small segments exceeding threshold [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 10.*.*.*:41714 -> *.217.*.110:443
12/18-11:28:01.117007 [**] [129:5:1] Bad segment, adjusted size <= 0 [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 10.*.*.*:51521 -> *.52.*.18:80
12/18-11:28:02.986880 [**] [129:15:1] Reset outside window [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 10.*.*.*:50098 -> *.58.*.227:443
12/18-11:28:03.332611 [**] [129:14:1] TCP Timestamp is missing [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 10.*.*.*:52261 -> *.13.*.3:443

図 3 Snort による異常検知の例

3.3 異常検知の例

稼働した Snort が検知した異常の例を図 3 に示す。セキュリティ上の理由により、IP アドレスは一部加工してある。Snort の異常検知は、この図のように違反パケット 1 個につき 1 行のログ、およびパケットダンプを出力する。ここで特筆すべきは、pcap 形式として出力されたパケットダンプは違反したパケットのみであるということだ。管理者が異常を知るには、まず Snort のログを目視により検査し、怪しいと思われるパケットダンプを Wireshark などのパケット解析ツールで検査する必要がある。通常、一つの通信セッションは複数のパケットから構成されており、保存された 1 パケットのみから異常か否かを判断することは難しいが、Snort のみでは違反パケットの周囲のパケットまで含めた調査をすることはできない。

また、本校で Snort を稼働してわかったことは、違反パケットのログの個数が意外と多いということである。図 3 の例では約 1 分おきにログが出力されており、違反パケットはこのように頻繁に発生している。そのため、内容をすべて調査することは時間的に無理があると判断し、メールによりログを自動通知する仕組みを構築した。また、Snort が Prioriry 3 以上を付与したログやパケットダンプに絞って詳細に調査するなど実働レベルの工夫をした。

4. Security Onion の導入と構築

4.1 Security Onion 導入時の検討事項

前章で述べた通り、Snort において違反パケットのログと pcap 形式のパケットダンプ 1 個 1 個を手作業で解析するのは限界がある。そこで、2018 年度にネットワークの監視用 PC を導入するための予算措置を行い、2019 年 3 月に Snort を IDS とした Security Onion を稼働した。なお、Snort は 2017 年度の Business プランを継続することとし、Oinkcode は Security Onion にそのまま引き継ぐこととした。

4.2 サーバスペック

今回導入した専用 PC のマシンスペックを表 1 に示す。導入した PC は将来的な拡張を考慮してミドルタワー型のデスクトップ PC とした。HDD 容量は導入予算との兼ね合いで最小限の 2TB としたが、容量が大きければそれだけログやパケットダンプの長期保存が可能となる。熊本高専八代キャンパスのネットワークの利用状況を調査したところ、2TB の容量は、夏休みなどの長期休暇中で 1 週間程度、平日であれば 0.5〜1 日で使いきることが分かった。そのため、10 日のパケットダンプを保管する場合は単純計算で 20〜40TB の容量が必要となる。このことから、ログ用 HDD の増設に備えて SATA ポートや電源容量、ケースの空きスロットはできる限り余裕をもって確保しておいたほうが良い。

また、今回は CPU のコア数を 6 とした。コア数を多くした理由は、Security Onion では、Snort や Bro といった処理が重いプロセスのほかに、仮想マシンを複数動作させなければならないからである。今回、6 コアを確保できたため、Snort のプロセスを Security Onion 内部で 2 つ起動した。これによりパケット解析を 2 つのコアで分散処理でき、パケット解析の取りこぼし要因を減らすことができる。また、Security Onion ではそれ以外に Docker[9] と呼ばれるコンテナ型仮想化環境が必要である。通常の Linux ディストリビューションでは一部のファームウェアをバージョンアップしたことで動作不良が発生することがたびたびあるが、Docker ではコンテナと呼ばれる一揃いのファームウェアリソースが提供されるため、いわゆるファームウェアバージョンによる相性問題が発生しない。この Docker の特徴を利用して ElasticStack[4] のほぼすべてのサービスが提供される。手元の環境では、Security Onion が起動すると containerd-shim が 5 つ、docker-proxy が 9 つ自動的に起動している。

構築後に、Security Onion の CPU 使用率を調査した。具体的には、シェルにてログイン後、top コマンド実行中に'1'を押下するとコアごとのリアルタイムな使用率が表示される。図 4 に結果を示す。%Cpu の後に続く us がユーザプロセス、sy がシステムプロセス、id がアイドル時間であり、それぞれの割合(%)が表示されている。図からもわかるとおり、6 コアそれぞれが均等に負荷分散されており、平均して 20.0%の CPU コア使用率であることがわかった。CPU 使用率に余裕があるように思えるが、突発的な高負荷に対応するためにはこの程度で留めておいた方が良い。また図 4 の下部に CPU 使用率の高い順にプロセスが表示されている。図からもわかるように、java, Bro, Snort が CPU 時間を消費する上位プロセスであった。なお、このうち java が Docker で利用しているプロセスである。このことから、ネットワークの基幹部分に設置する Security Onion はコア数をできるだけ多く確保しておいた方が良いと考える。

表 1 Security Onion 専用 PC のマシンスペック

CPU	Intel(R) Core(TM) i5-8400 CPU @ 2.80GHz 6Core
MEM	16GB
HDD	2TB （可能なら 20〜40TB）
Network Interface	1000Base-TX ×2
SATA	6 ポート（できるだけ多い方が良い）
ケース	ミドルタワー型 5inch ベイ 5 スロット
電源	600W （できるだけ大きい方が良い）

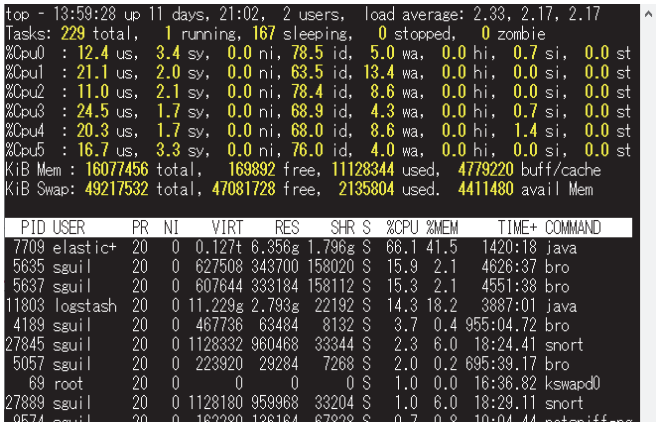


図 4 各コアおよびプロセスごとの CPU 使用率

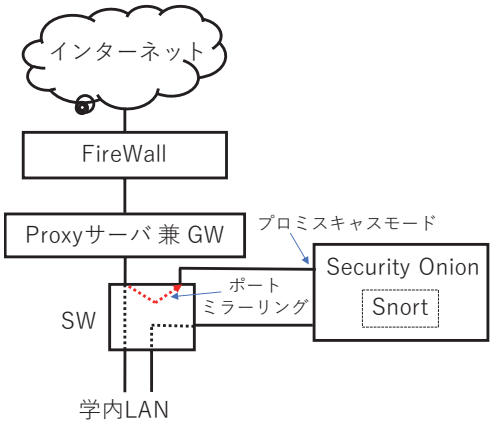


図 5 Security Onion の設置場所と全体の構成

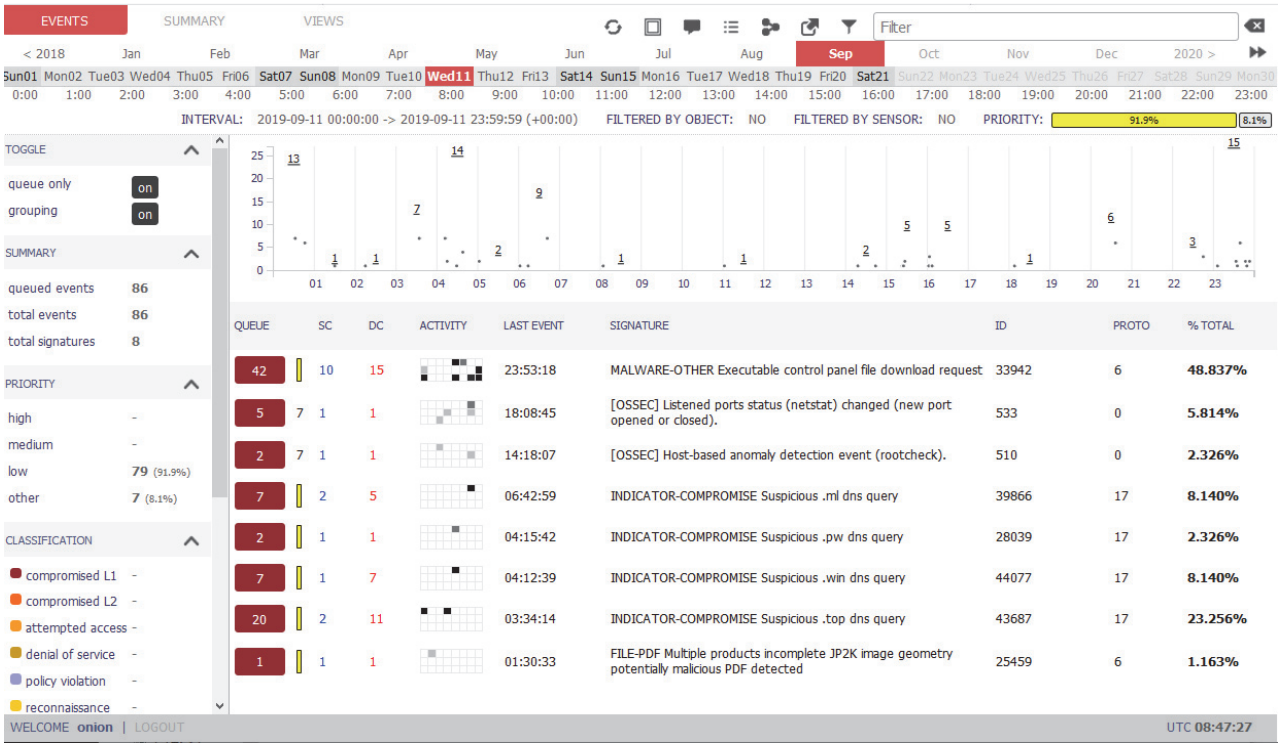


図 6 Squert による検出結果の分類と時間分布

4.3 全体構成

今回設置した Security Onion の設置場所および全体構成を図 5 に示す。Security Onion 用のサーバには、２つのネットワークインタフェースを用意した。一方は学内ネットワークに接続するため、もう一方は膨大な通信を監視するためのものであり、後者はプロミスキャスモードで動作させることとした。パケット監視専用のポートを用意したことで、通常のサーバとの間の通信パケットが分離でき、監視パケットの欠損を減らすことが可能となる。

4.4 異常検知の例

ここでは、Security Onion で検知した具体的な違反パケットや可視化されたログの例を示しておく。図 6 は、Squert と呼ばれるデータの分析に使用するツールのスクリーンショットである。Snort が違反パケットとして判定した情報を基に、横軸に時間、縦軸に発生件数を示したグラフが描画され、違反パケットの時間変化を確認することができる。また、図の下部には、観測された違反パケットが種類ごとに分類され表示されている。これにより、違反パケットの総数や傾向が一目でわかる。

図 7 には Kibana と呼ばれるパケット可視化ツールのスクリーンショットを示した。図は、平日一日の DNS クエリについて、横軸を時間、縦軸を回数としてグラフ表示している様子である。就業時間である午前 8 時半ごろから上昇し、17 時を過ぎると徐々に下降していく。異常が発生したときは DNS クエリに変化を来たすことが多く、このように日頃から平常時の DNS クエリの様子を観測しておくとよい。

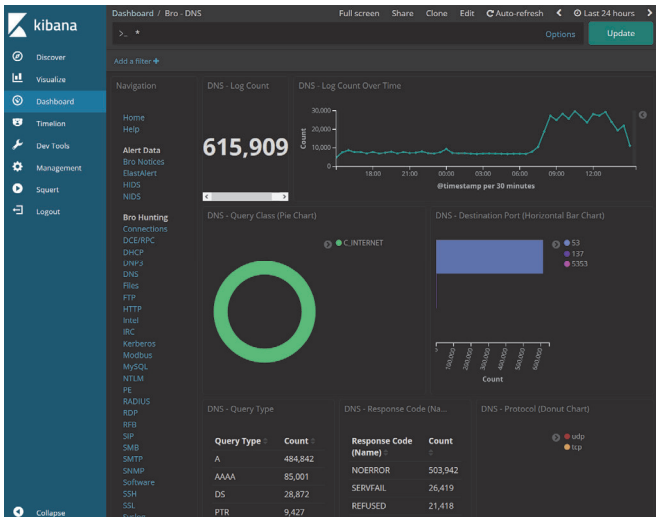


図 7 Kibana による DNS パケットの分類と時間変化

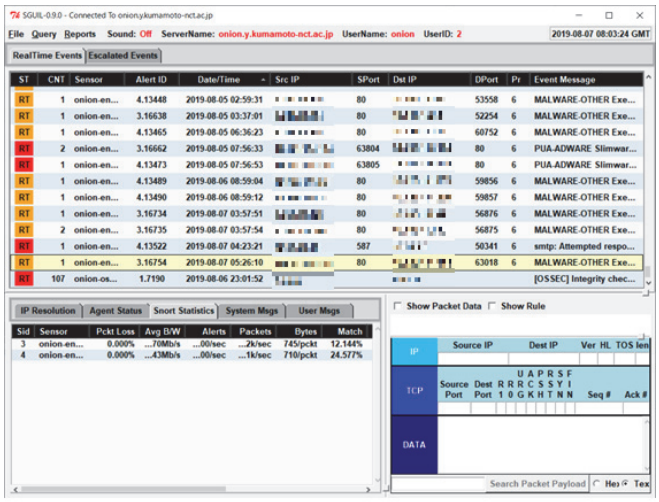


図 8 Sguil による検出結果の分類と可視化

図 8 には Security Onion のもう一つの代表的な分析ツールである Squil のスクリーンショットを示しておいた。セキュリティ上の理由により図の一部は加工してある。

このツールは、Snort により観測された違反パケットを時間順や IP アドレス順など目的に応じて瞬時に並べ変えることができる。また、似たような違反パケットは 1 行にまとめられ個数が表示されるため、全体的な見通しが非常に良い。違反パケットを右クリックすることで、IP アドレスを VirusTotal[10]や whois で調査したり、代表的なパケット解析ツールである Wireshark を起動し、シームレスに詳細に解析することが可能である。

この Sguil を 5 か月間使用した結果、熊本高専八代キャンパスでは平均して一日に数件～10 件程度の違反パケットが検出されることがわかった。この違反パケットの多くは学生 VLAN からの接続である。広告サイトやブラックリスト IP アドレスとの通信、セキュリティ脆弱性をついた PDF ファイルダウンロードなど多岐に及んでいるが、詳細な調査の結果、マルウェア対策ソフトのパターンファイルやアプリケーションのアップデートパッチが提供されるなど、対策が施されているものが大半を占めていることがわかった。

また、これまでに、学生が学内に持ち込んだ端末がアドウェアに感染していたり、ブラックリストとして登録されている IP アドレスと通信したりしていたケースが 5 件あった。いずれも学内へのマルウェア拡散や情報漏洩などのインシデントには至っていない。発見後の対応として、違反パケットの IP アドレス情報と学内 Wi-Fi の使用記録、IP アドレス払い出し情報を突き合わせることで学生を特定することができたため、本人に注意喚起しマルウェア対策や HDD フルスキャンを求めた。また、一部の学生には 1 時間程度の面談を実施して詳細の確認などを行った。

4.5 稼働状況の調査

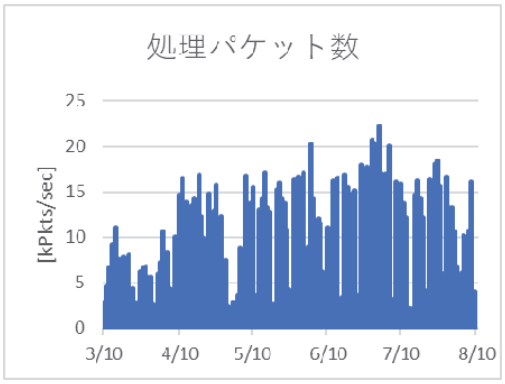
Security Onion の稼働状況を調査した結果を図 9 に示す。Security Onion は 5 分おきにログを出力しており、グラフの縦軸はすべて 5 分間あたりの平均値として示してある。図 9(a)の通過パケット数は、5-20[kPackets/sec]であり、多くのパケットを処理している様子がわかる。また、図 9(b) の CPU 使用率の調査では 20-40%程度を示しており、いずれの時間帯においても CPU100%とならないことがわかった。

一方、図 9(c) の違反パケットのアラート数については注意が必要である。グラフの最大値 0.93 [件/sec]は 5 分間あたりの平均値としての表記であり、つまり 5 分間に発生した違反パケットのアラート数を 300 秒で割った 1 秒あたりの件数として表示している。そのため、グラフ値に 300 を乗じたものが実際の 5 分間当たりのアラート件数となり、最大で 5 分間あたりおよそ 280 件のアラートが報告されていることがわかる。ちなみにこのアラート最大件数はトップレベルドメイン（TLD）の DNS 検索に関するものであった。それ以外にも多くの違反パケットのアラートが挙がっている。

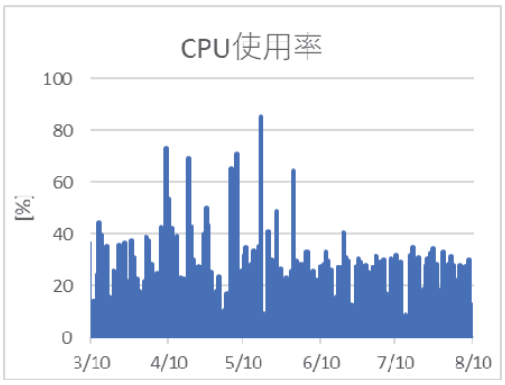
図 10 では、2019 年 8 月 8 日（木）一日当たりの処理パケット数を示している。この日は本校では通常授業（試験返却等）が実施されている。図から、就業開始急激に上昇し、就業終了後に徐々に減少するという一日の処理パケットのおおよその変化が読み取れる。

5．まとめ

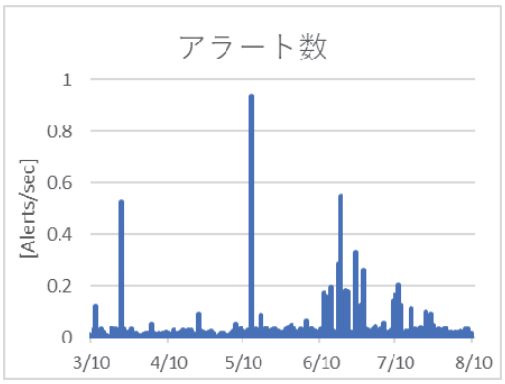
今回、Snort と Security Onion を導入した。導入時のコストはそれほど高くはないものの、導入後のコストが非常に高く、1 件の違反パケットを調査するだけで多くの時間を必要とすることがわかった。違反パケットの調査を進めていく中で、違反パケットの多くが誤検知で占められていることが分かってきた。導入当初は違反パケットのすべてについて追跡調査をしていたが、非常に多くの時間を浪費してしまうことから、過去に調査したものとよく似た違反パケットについては調査しないなどの工夫も必要であり、ある程度の経験が必要である。また、将来的には誤検知を減らすよう Snort ルールの全体的な見直しも必要となろう。



(a) 5 か月間の処理パケット数（5 分間あたりの平均）



(b) 5 か月間の CPU 使用率（5 分間あたりの平均）



(c) 5 か月間の違反パケットアラート数（5 分間あたりの平均）

図 9 Security Onion の稼働状況調査

一方、誤検知の中には、SSL など暗号化された通信も含まれており、暗号化されたランダムなパケットが違反パケットのパターンと偶然一致したようなものも見受けられた。1 日あたり 2TB の通信が発生する Proxy サーバの環境下ではこのような偶然が数多く発生するが、この誤検知を減らす抜本的な解決策はない。

今後の検討事項として、暗号化パケットの追跡がある。現在、熊本高専八代キャンパスの Proxy サーバでは SSL などの暗号化通信がパケット全体の 5 割を占めている。現在導入している Security Onion は、暗号化通信の中でまで調査する手段がないため、単純計算で違反パケットの半数が見過ごされていることになる。今後、暗号化通信の割合はま

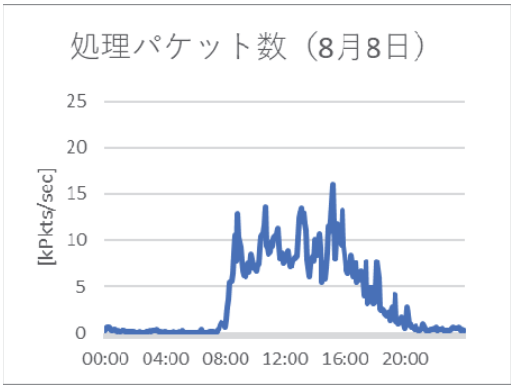


図 10　一日あたりの処理パケット数の推移

すます多くなることが予想されており、SSL インスペクションなど暗号パケットを平文パケットにする新たな仕組みが必要となる。この実現には技術的な課題もさることながら、倫理的な問題もある。たとえば、ログインが必要なサイトのアカウントやパスワードなどを含めすべてが Security Onion 上で平文に戻されてしまうため、パスワードを盗聴していることと何ら変わらない。実現には学内のコンセンサスを得る必要があるが非常に困難であると考える。

最後に、Security Onion は多機能であり、使いこなすには更なる慣れと経験が必要であることを痛感している。セキュリティに関連する教職員で勉強会などを開催するなどしていきたい。

（令和元年 9 月 25 日受付）
（令和元年 12 月 5 日受理）

参考文献

(1) Cisco : Snort - Network Intrusion Detection & Prevention System, https://snort.org/, 2019/9/21
(2) Security Onion Solutions : Security Onion, https://securityonion.net/, 2019/9/21
(3) OISF : Suricata, Open Source IDS / IPS / NSM engine, https:// suricata-ids.org/, 2019/9/21
(4) Elastic : オープンソースの Elastic Stack（Elasticsearch、Kibana、Beats、Logstash）でリアルタイムな検索と分析, https://www.elastic.co/jp/, 2019/9/21
(5) Squert Proejct : The squert project, http://www.squertproject.org/, 2019/9/21
(6) Bro Project : The Zeek Network Security Monitor, https://www.zeek.org/, 2019/9/ 21
(7) Wireshark Foundations : Wireshark - Go Deep, https://www.wireshark.org/, 2019/9/21
(8) The netsniff-ng Proejct : netsniff-ng toolkit, http://netsniff-ng.org/, 2019/9/21
(9) Docker Inc. : Enterprise Container Platform | Docker, https://www.docker.com/, 2019 /9/21
(10) VirusTotal Community : VirusTotal, https://www.virustotal.com/, 2019/9/21