Wykrywanie i przeciwdziałanie atakom DoS/DDoS

Kulig Sebastian, Mirowska Diana, Wnęk Karol

Atak DoS/DDoS w sieci SDN

Ataki protokołowe / warstwy protokołów :

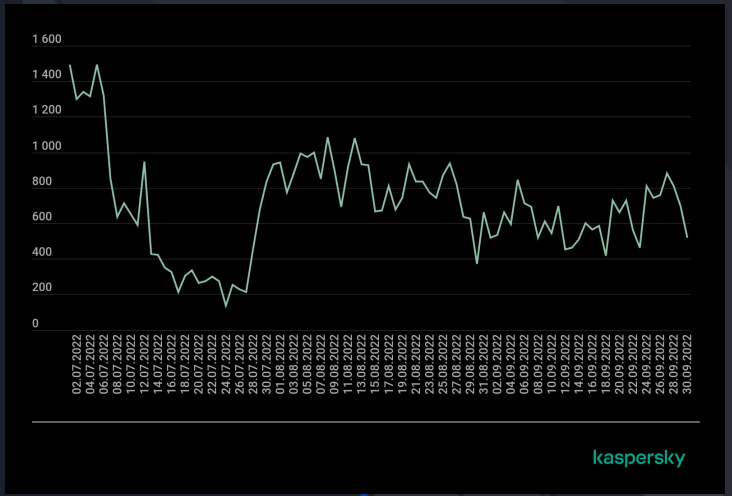
- Ping flood / ICMP flood
- SYN flood i ACK flood

Ataki amplifikacyjne:

- Low and slow / R.U.D.Y.
- Fork bomb

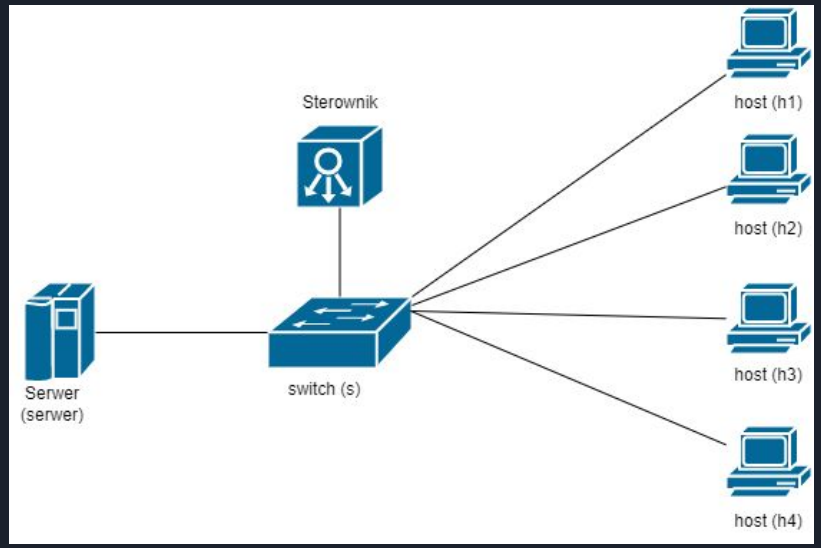
Kontrolery SDN cechuje wrażliwość na wyczerpanie zasobów, w szczególności takich jak ilość możliwych do zainstalowania przepływów. Aby przetestować tą podatność wybraliśmy atak SYN flood.

Na podstawie zbieranych statystyk, kontroler powinien wykryć anomalię ruchową a następnie zablokować złośliwy przepływ.

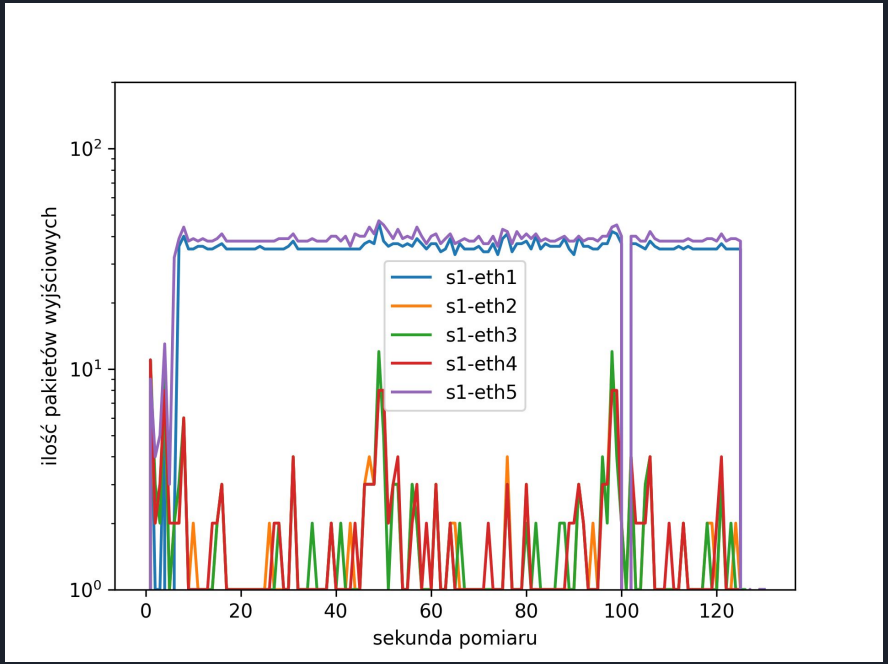


Użyta topologia - DoS

- Wszystkie hosty komunikują się z serwerem
- H1 pełni rolę atakującego
- Sterownik monitoruje statystyki ruchowe na przełączniku



Scenariusz bazowy - bez mitygacji ataku



Wykrywanie ataków - analiza pakietów

Scenariusz ataku - TCP Flood DoS

- Analiza pakietów IPv4 i zbieranie statystyk
- Szczegółowa analiza pakietów TCP pod kątem typów wiadomości przychodzących
- Liczniki różnych typów wiadomości TCP
- Wyliczanie entropii - zmian w ilości przychodzących pakietów
- Podjęcie akcji drop na podstawie sourceIP dla hosta, który przekroczył treshold

Entropia w sieci

Entropia H(X) dla dyskretnej zmiennej X jest zdefiniowana wzorem

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

Entropia – w teorii informacji jest definiowana jako średnia ilość informacji, przypadająca na znak symbolizujący zajście zdarzenia z pewnego zbioru. Zdarzenia w tym zbiorze mają przypisane prawdopodobieństwa wystąpienia.

Zbieranie statystyk

```java
private OFStatsRequest<?> prepareFlowStatsRequest(IOFSwitch sw){
    Match match = sw.getOFFactory().buildMatch().build();
    return sw.getOFFactory().buildFlowStatsRequest()
            .setMatch(match)
            .setOutPort(OFPort.ANY)
            .setTableId(TableId.ALL)
            .build();
}
```

```java
switch (this.statsType) {
case FLOW:
    OFFlowStatsReply fsr = (OFFlowStatsReply) values.get(0);
    for (OFFlowStatsEntry pse : fsr.getEntries()) {
        IPv4Address srcIp = pse.getMatch().get(MatchField.IPV4_SRC);
        IPv4Address dstIp = pse.getMatch().get(MatchField.IPV4_DST);
        if (previousValuesFlows.containsKey(srcIp)) {
            double tput = 8.0 * (pse.getByteCount().getValue() - previousValuesFlows.get(srcIp)) / PORT_STATISTICS_POLLING_INTERVAL * 1000.0 / 1024 / 1024;
            logger.info("\tSRC IP: {}, speed: {} MB/s", srcIp, tput);
            logger.info("\tSRC IP: {}, packets count: {}", srcIp, pse.getPacketCount().getValue());
            if(pse.getPacketCount().getValue() > 300L && isBlockingDoSEnabled){
                logger.info("\t====================== ATTACK DETECTED - ADDING BLOCKING RULE ======================");
                BlockingRuleBuilder.addBlockingRule(sw, srcIp, dstIp);
            }
        }
        previousValuesFlows.put(pse.getMatch().get(MatchField.IPV4_SRC), pse.getByteCount().getValue());

    }
    break;
```

```java
@Override
public net.floodlightcontroller.core.IListener.Command receive(IOFSwitch sw, OFMessage msg, FloodlightContext cntx) {
    OFPacketIn pi = (OFPacketIn) msg;
    Ethernet eth = IFloodlightProviderService.bcStore.get(cntx,IFloodlightProviderService.CONTEXT_PI_PAYLOAD);

    StatisticsCollector.getInstance(sw, cntx);

    if (eth.isBroadcast() || eth.isMulticast()) {
        doFlood(sw, pi, cntx);
    } else {
        doForwardFlow(sw, pi, cntx, false);
    }

    return Command.STOP;
}
```

Statystyki przepływu TCP - flagi pakietów przychodzących

```
TcpFlagStats stats = statsMap.get(key);

switch (tcp.getFlags()) {
    case 2:
        stats.syn++;
        break;
    case 18:
        stats.synack++;
        break;
    case 1: // FIN Flag
        stats.fin++;
        break;
    case 17: // FIN-ACK Flag
        stats.finack++;
        break;
    case 16: // ACK
        stats.ack++;
        break;
    default:
        break;
}
```

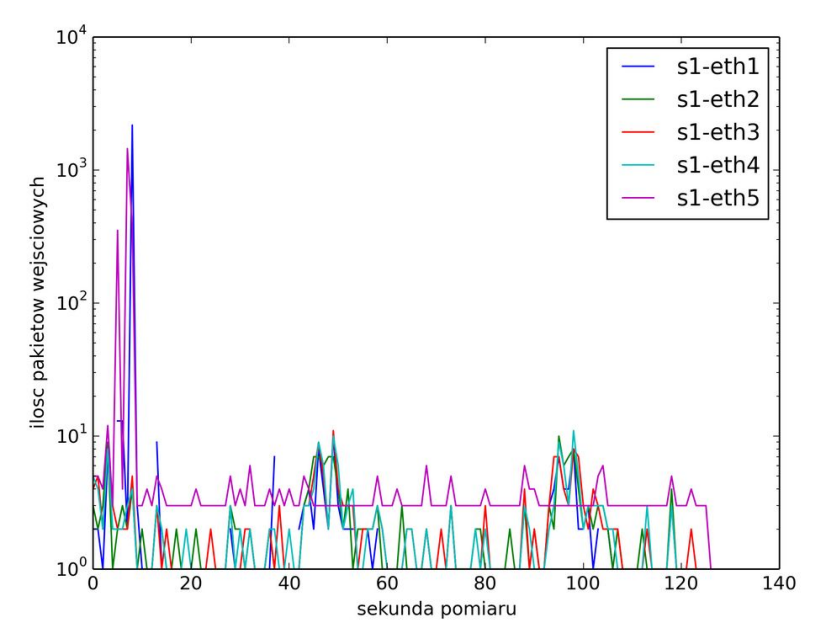
Podjęcie decyzji o usunięciu przepływu

- zbieramy statystyki ze switcha na podstawie których wyliczamy **prawdopodobieństwo pojawienia się pakietów danego typu** - pierwsza faza skryptu normal_traffic
- na podstawie statystyk wyliczamy **wartość entropii**
- po rozpoczęciu ataku przy pomocy narzędzia **hping3 –flood** można zaobserwować zdecydowaną **przewagę pakietów TCP typu SYN** co powoduje zmniejszenie się entropii
- wykrycie **zmniejszenia entropii** wywołuje podjęcie akcji drop dla pakietów pochodzących z podejrzanego adresu ip
- w celu zmniejszenia ilości false-positive drugi warunek detekcji ataku to duża **różnica** w ilości przychodzących **pakietów SYN a pakietów ACK**

```
    # Start generating traffic
    start_normal_traffic(net)
    time.sleep(2)
    start_monitor()
    time.sleep(5)
    start_attack(net)
    time.sleep(120)
#   CLI( net )

    stop_attack()
    time.sleep(5)
    stop_monitor()
    CLI(net)
    net.stop()
```

Scenariusz testowy - wykrywanie i blokowanie złośliwych przepływów




```
SdnLabListener:nioEventLoopGroup-3-2]    ====================== ATTACK DETECTED - ADDING BLOCKING RULE ======================
SdnLabListener:nioEventLoopGroup-3-2]    Blocking path: 10.0.0.1->10.0.0.5:80
BlockingRuleBuilder:nioEventLoopGroup-3-2] ====================== BLOCKING RULE ADDED. BLOCK SRC IP: 10.0.0.1 TO DST IP 10.0.0.5 ======================
BlockingRuleBuilder:nioEventLoopGroup-3-2] Flow from ip address 10.0.0.1 dropped; match: OFMatchV1Ver10(eth_type=0x800, ipv4_src=10.0.0.1, ipv4_dst=
SdnLabListener:nioEventLoopGroup-3-2]    ====================== ATTACK DETECTED - ADDING BLOCKING RULE ======================
SdnLabListener:nioEventLoopGroup-3-2]    Blocking path: 10.0.0.1->10.0.0.5:80
```

Wyliczanie entropii i przekroczenia tresholdu

```
    long totalFlags = stats.ack + stats.synack + stats.syn;
    if (totalFlags > 0) {
        double entropy = Integer.MAX_VALUE;
        double probSyn = (double)stats.syn/totalFlags;
        double probSynAck = (double)stats.synack/totalFlags;
        double probAck = (double)stats.ack/totalFlags;
        double temp1 = 0;
        double temp2 = 0;
        double temp3 = 0;
        if(probSyn != 0)
            temp1 = probSyn*(Math.log(probSyn)/Math.log(2));
        if(probSynAck != 0)
            temp2 = probSynAck*(Math.log(probSynAck)/Math.log(2));
        if(probAck != 0)
            temp3 = probAck*(Math.log(probAck)/Math.log(2));
        entropy = -(temp1+temp2+temp3);
        if (entropy<1.3 && Math.abs(stats.syn - stats.ack)>100) {
            log.info("\t====================== ATTACK DETECTED - ADDING BLOCKING RULE ======================");
            log.info("\tBlocking path: {}", key);
            BlockingRuleBuilder.addBlockingRule(sw, ipv4.getSourceAddress(), ipv4.getDestinationAddress());
        }
    }
}
```

Usunięcie przepływu

```
public static void addBlockingRule(IOFSwitch sw, IPv4Address srcIp, IPv4Address dstIp) {
    OFFlowMod.Builder fmb = sw.getOFFactory().buildFlowAdd();
    Match.Builder mb = sw.getOFFactory().buildMatch();
    if (srcIp != null) {
        mb.setExact(MatchField.ETH_TYPE, EthType.IPv4).setExact(MatchField.IPV4_SRC, srcIp)
        .setExact(MatchField.ETH_TYPE, EthType.IPv4).setExact(MatchField.IPV4_DST , dstIp);
        logger.info("====================== BLOCKING RULE ADDED. BLOCK SRC IP: {} TO DST IP {} ======================", srcIp, dstIp);
    }
    Match m = mb.build();

    // actions - no actions to drop packet
    OFActionOutput.Builder aob = sw.getOFFactory().actions().buildOutput();
    List<OFAction> actions = new ArrayList<OFAction>();
    actions.add(aob.build());

    fmb.setMatch(m).setIdleTimeout(FLOWMOD_DEFAULT_IDLE_TIMEOUT)
            .setHardTimeout(FLOWMOD_DEFAULT_HARD_TIMEOUT)
            .setPriority(FLOWMOD_DEFAULT_PRIORITY);


    // write flow to switch
    try {
        sw.write(fmb.build());
        logger.info("Flow from ip address {} dropped; match: {}", new Object[] {srcIp, m.toString() });
    } catch (Exception e) {
        logger.error("Error {}", e);
    }
}
```

Repozytorium GitHub

https://github.com/SebastianKulig/SDN

Źródła

- https://securelist.com/ddos-report-q3-2022/107860/
- instrukcja do laboratorium nr 8
- https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/21856267/How+to+Collect+Switch+Statistics+and+Compute+Bandwidth+Utilization
- https://github.com/floodlight/floodlight/tree/71fe8a7e72096eb0fd96c1d814a04e3b7b782830/src/main/java/net/floodlightcontroller
- https://github.com/floodlight/floodlight/tree/71fe8a7e72096eb0fd96c1d814a04e3b7b782830/src/main/java/net/floodlightcontroller
- Denial of Service Attack Mitigation Strategy on SDN Controller , Mutalifu Kuerban, 2016
- DDoS Attack Detection and Mitigation in SDN, Apruv Ranjan, Mohit Kumar Jaiswal, Abhinav Yadav, Deepak Bhist, 2017