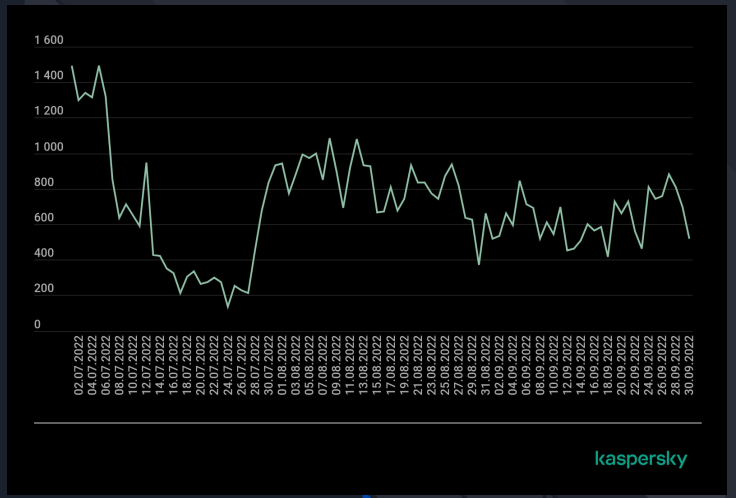
Wykrywanie i przeciwdziałanie atakom DoS/DDoS

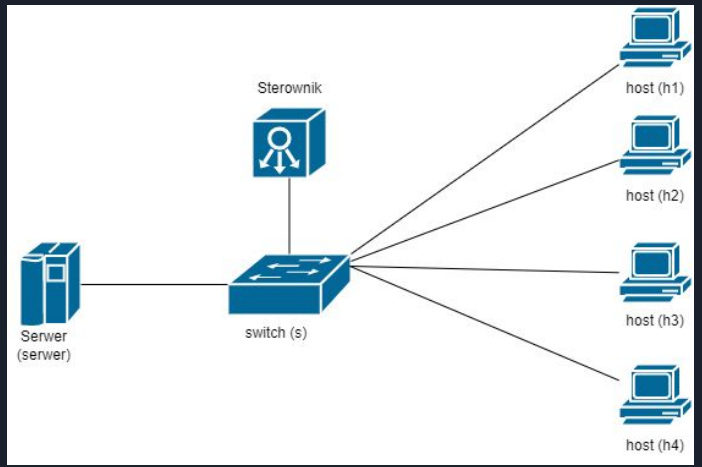
Kulig Sebastian, Mirowska Diana, Wnęk Karol

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

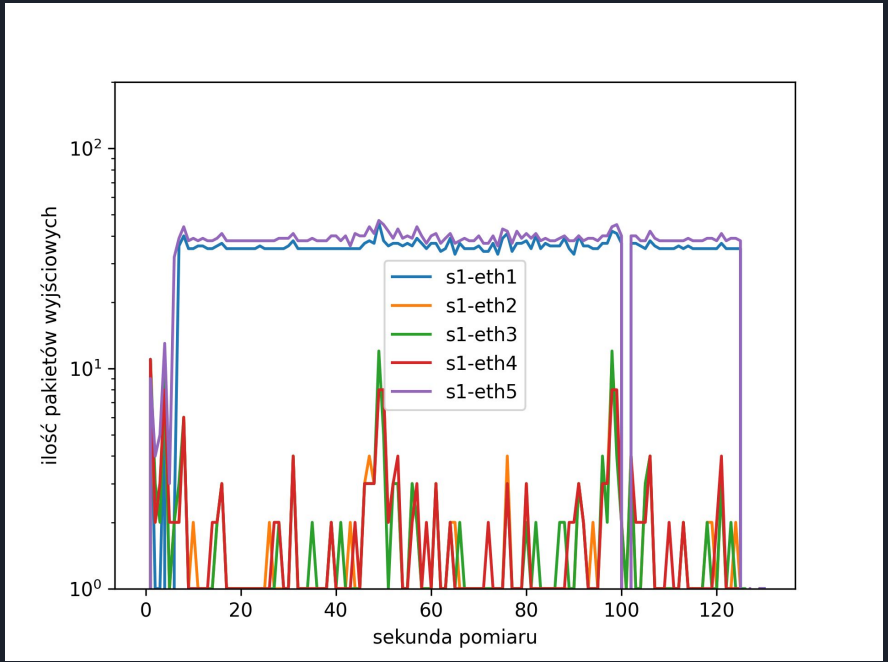


użyta topologia

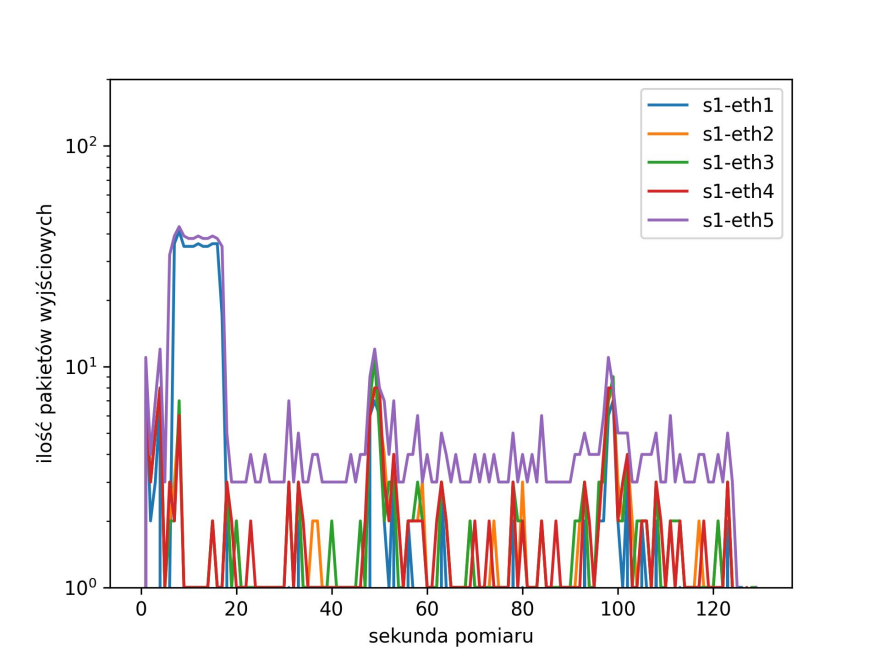
- wszystkie hosty komunikują się z serwerem
- h1 pełni rolę atakującego
- sterownik monitoruje statystyki ruchowe na przełączniku



scenariusz bazowy



scenariusz testowy




```
[p.e.a.k.StatisticsCollector$StatisticsPoller:Timer-1]    ==================== ATTACK DETECTED - ADDING BLOCKING RULE ====================
[p.e.a.k.BlockingRuleBuilder:Timer-1] ==================== BLOCKING RULE ADDED. BLOCK SRC IP: 10.0.0.1 TO DST IP 10.0.0.5 ====================
[p.e.a.k.BlockingRuleBuilder:Timer-1] Flow from ip address 10.0.0.1 dropped; match: OFMatchV1Ver10(eth_type=0x800, ipv4_src=10.0.0.1, ipv4_dst=10.0.0.5)
```

Sposób implementacji

zbieranie statystyk

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

```
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;
```

```
@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;
}
```

podjęcie decyzji o usunięciu przepływu

usunięcie przepływu

```java
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 git

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