Proactive Intrusion Defense
Against DDoS Flooding Attacks:
Adaptive Filtering with Security Datamining
– The NetShield Approach at USC*

Kai Hwang, Sapon Tanachaiwiwat, and Pinalkumar Dave
University of Southern California, Los Angeles, CA. 90089

Abstract: The NetShield security system was developed at USC to defend against network worms and flood attacks. The system prevents malicious hackers from orchestrating DDoS flooding attacks on any IP-based public network. This article presents new packet filtering and anomaly detection techniques developed with the NetShield system. All packets from each IP source are counted and timed during their life cycles. Special IP counters and timers are used to support the filtering process. Attack profile datamining is used to support protocol anomaly detection of flood attacks. We use an alarm-matrix model to assess the effectiveness of the attack/alarm verification and packet filtering processes.

Index Terms: DDoS attacks, IP address spoofing, packet filtering, datamining for security, attack classification, anomaly detection, quality of service, and intrusion response.

Introduction

The Computer Emergency Response Team (CERT) at Carnegie-Mellon University has reported the doubling of security incidents each year since 1988 [1]. Major victims include Yahoo, Amazon, CNN, and E*Trade among many other web sites in public and private sectors. Hundreds of network servers and millions of client hosts were pulled down worldwide by those DDoS (distributed denial of service) attacks. The widespread flood attacks have resulted in huge financial losses and intolerable interrupts to Internet services. Worst of all, the problem is escalating to a global scale of impact, taking a long time to recover.

To stop future attacks, automated intrusion detection and response (IDR) systems should be timely deployed to secure E-commerce, digital government, network service industries, etc. The problem remains unsolved, because SYN, UDP, and ICMP attack tools are easy to use [2]. The attackers can simply act faster than most defense systems deployed so far. Our NetShield system was developed to meet the increasing demand of cost-effective IDR solution to the flooding problem [2, 4, 12]. We aim at containing the flood attacks on any open networks. We choose a new approach through adaptive packet filtering and attack profile datamining for protocol anomaly detection.

Corresponding author: Professor Kai Hwang, Internet and Wireless Security Lab., EEB 212, University of Southern California, Los Angeles, CA 90089. Email: kaihwang@usc.edu
Network hackers exploit system vulnerabilities, installing backdoor software, and covering the root kit tracks. Most intrusion detection systems (IDS) cannot cope with DDoS attacks, effectively [5, 15]. Reflected floods were reported to launch from a large number of compromised TCP servers [4]. Industrial flood control devices appeared as the NetEnforcer (http://www.allot.com/html/solutions_enterprise_dos_attacks.shtm) and FloodGuard (http://www.reactivenetwork.com). These are mainly designed to monitor network traffic and to enforce new security policies to block the DDoS attacks from reaching the victim’s hosts.

The NetShield concept was first reported in a RAID 2003 paper [7]. This article presents the enabling security mechanisms and integrated architecture of the NetShield system. In particular, we introduce the original concepts of adaptive filtering using IP counters, datamining for anomaly detection, and alarm matrix for performance assessment. This work was inspired by several previous works [3, 5, 8, 9, 10, 11, 13, 14]. The new features presented have never been implemented in any of the industrial flood blockers.

The NetShield System at USC

The NetShield system was originally designed to detect and classify different attack types and assesses the residue risks if certain countermeasures are deployed [6]. The system design has gone through refinement over the years to support real-time intrusion analysis, attack/alarm notification, and user-controlled mitigation procedures [7]. The NetShield architecture is only briefly introduced below to prepare readers with the system environment, where several new defense mechanisms are tested. Detailed architecture of NetShield system can be found in our earlier reports.

Testbed Network Environment: Figure 1 shows the network setting, where the NetShield system is positioned to play a role in security control. The testbed is built around an Ethernet-base Linux cluster at USC. There is a firewall gateway between the cluster and the network router. The router is connected to the Internet backbone. The NetShield system is built with a run-time software library for automated intrusion detection and responses. The risk assessment system (RAS) assesses the risks of any alarm raised and estimates the damages. The intrusion response system (IRS) is used to deploy fast countermeasures.
The NetShield core interacts with the gateway firewall, network router, and Internet service provider (ISP) at the upstream. Smart filtering is done at the router stage to drop packets from suspicious or verified IP sources. An intrusion prevention system (IPS) is installed in the firewall to stop the hackers before they can do any damage. The IPS stops the DDoS attacks by building defenses against unchanged signatures, behaviors, or patterns of those attacks. We spread some of the IPS functionalities into the firewall, network router, IDS, and ISP using software-based heuristics, sandbox protection, and kernel-based protections, etc.

**Design Objectives of NetShield:** The IDS and RAS subsystems work together to distinguish among the hits, misclassifications, and false alarms. At the network level, we monitor the UDP and TCP packets over specific ports. A network-based IDS scans the network links to look for illicit traffic, such as the UDP packets from a known port. Protocol-based anomaly detection is built in the NetShield core with the help of some datamining and learning mechanisms. Listed below are five design objectives of the NetShield system:

1. A scalable security architecture to protect local/enterprise networks and expandable to secure upstream ISP networks
2. Mitigating flood attacks through adaptive filtering with differential quality of services between good packets and the bad ones.
3. Using datamining to support attack profiling for anomaly detection of flood attacks with low overhead
4. Performing risk assessment to enable automated intrusion response with frequent security policy updates
5. Long-term goal to support distributed security infrastructure for peer-to-peer and grid metacomputing

**Current System Status:** The NetShield system is built with the help of a simulator running on a Linux cluster at USC. So far, only the adaptive filtering router and datamining anomaly detection are simulated. Two existing IDS systems are used in our experiments, namely the LIDS for Linux hosts and the Snort for network-based detection. We augment these two systems with our own anomaly detection program modules. The RAS and IRS subsystems are yet to be completed. At present, these development tasks are still in progress.

Only new security features are reported in this article. The features are embedded in the network router, gateway firewall, and the IDS running with the NetShied simulator. We use the simulator to verify the adaptive packet filter design. The alarm-matrix framework is then used to model the platform/IDS behavior. Attack profiles are presently simulated with a small hypothetical database. Benchmark experimental results will be reported in the future, when the NetShield testbed gets completed and fully tested through experiments.

**Traffic Bursts Triggered by DDoS Attacks**

DDoS attack programs are often embedded in packets amid large bursts of traffic. We use several parameters in Table 1 to characterize traffic profiles [13]. Three profiles are shown in Table 1, ranging from
light to heavy traffic. These parameters are used in designing the router filter and the anomaly detector in our NetShield security system. A typical traffic spectrum is shown in Figure 2.

**Traffic Parameters Monitored:** All traffic rates are measured by *packets per second (pps)*. With a low traffic rate, most normal packets are allowed to pass through. With a high burst of traffic, we need to detect the possibility of a DDoS attack. The **CRS** rate indicates the new requests from the same IP source. The **NOR** indicates the open (yet to be established) requests from all sources. The **EAC** is the number of established connections per second from all sources.

<table>
<thead>
<tr>
<th>Traffic Profiles (pps)</th>
<th>Connection Requests from the Same IP (CRS)</th>
<th>New Open Requests from all IPs (NOR)</th>
<th>Established All-to-all Connections (EAC)</th>
<th>Maximum traffic rate handled by a router (MAX)</th>
<th>Insecure IP List (IPinSec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Light, 100 pps</td>
<td>4 pps</td>
<td>50 pps</td>
<td>75 pps</td>
<td>1000 pps</td>
<td>130.110.x.x</td>
</tr>
<tr>
<td>Medium, 190 pps</td>
<td>30 pps</td>
<td>423 pps</td>
<td>250 pps</td>
<td>1000 pps</td>
<td>132.23.34.x</td>
</tr>
<tr>
<td>Heavy, 500 pps</td>
<td>100 pps</td>
<td>600 pps</td>
<td>768 pps</td>
<td>1000 pps</td>
<td>123.x.x.x</td>
</tr>
</tbody>
</table>

Note that traffic covered by **CRS** is a subset of **NOR**. The **EAC** and **NOR** are disjoint traffic sets and their union is the total traffic being processed by the router. The **MAX** is the maximum traffic rate that can be handled by a given router. The **IPinSec** is a black list of insecure IP addresses of potential attackers or handlers and zombies. The **IPinSec** list is exemplified by IP addresses with the same prefixes. The **CRS** counts new requests for connection from a single source. This resembles the DDoS attack pattern in recruiting handler and zombie machines to act as agents. We call this phenomenon an **IP redundancy**.
The network traffic is monitored with small time windows periodically. Six observation windows are shown in Fig.2. The windows \((T_0, T_1)\) and \((T_1, T_2)\) have lower traffic rates, thus considered normal (risk-free). The traffic observed in window \((T_4, T_5)\) is labeled as high risky for having many spikes above the \(\beta = 300\) pps threshold. The remaining 3 windows are considered medium risky.

The higher threshold will benefit normal traffic to pass through. The lower threshold implies tighter security control and thus more packets must be dropped quickly, including some good ones to avoid a buffer overflow in the router. In what follows, we consider packets originating from multiple IP sources. In a normal window, the IPs are most likely from friendly sources. A medium-risky window may contain some suspicious packets. In a high-risk window, all new IP addresses are possible attackers.

**Sustained Quality of Service:** To filter out the malicious packets, we introduce a new concept of sustained Quality of Service (QoS) towards efficient packet filtering. We define the sustained QoS as the retaining rate of good IP packets. The major advantage of our adaptive filtering scheme is to act in favor of normal traffic and against packets from IP sources that have triggered traffic bursts. A rough estimation of the advantage is defined by the following quantitative measure:

\[
\text{Sustained QoS} = 1 - \text{Probability of dropping good IP packets}
\]  

Traditional filtering uses the egress/ingress filters. You can only prevent others from being attacked by your network. Checking against a black list, one can not handle massive spoofed IP addresses either, because the black list is usually too short to be practical. If the router drops all suspicious packets by brute force, it may deny innocent good customers as well. If the router does not drop any packet, the flooding may lead to a buffer overflow and a complete denial of services [3].

**Adaptive Packet Filtering**

In our approach, we filter out most suspicious packets by capturing traffic irregularity triggered by DDoS attacks. We aim to keep the sustained QoS level high for good customers. The level is tied to the “reputation” of an IP address during its life cycle. Our new idea of adaptive filtering complements other filtering approaches reported in [3, 8]

**Enabling Mechanisms:** In Figure 3, we show the functional components in our adaptive filter design. We use an active IP (AIP) table to maintain all active IP addresses created with incoming packets. The arriving packets are kept in the router buffer (RB) before being forwarded to the destination server or dropped with suspicious IP spoofing or other detected protocol violations. The priority queue scheduler (PQS) forward the normal packets out to the server ahead of those suspicious or bad packets.

Each entry in the AIP table is a 3-tuple: \((IP_i, IPC_i, T_i)\), where \(IP_i\) is an active IP address, \(IPC_i\) is an IP counter associated with the i-th IP address, and \(T_i\) is a timer to capture the life span of that IP address. The IP counter keeps track the number of consecutive normal windows (no attacks) that IP has experienced. The counter is incremented by 1 for each normal window monitored. The IP timer starts with a time limit \(\pi\) and
reduces gradually to zero, when there are no packets arriving from that IP source. Whenever new packets arriving from the source, the timer is reset back to the maximum time \( \pi \) and counting down again. A long-idled IP is removed from the AIP table, when its timer is expired with a zero time count.

When a packet arrives at the router, the packet IP is checked against the AIP table for packet classification. If the IP is not found in the AIP, a new table entry is created with initial values \( IPC_i = 1 \) and \( Ti = \pi \). The IP counter enforces differential QoS for packets from different IP sources. The higher readout in the IP counter, the higher QoS the packet will receive. Both IP counters and timers are enabling mechanisms to implement the differential packet filtering algorithm as introduced below.

**Priority Queue Scheduling:** Each incoming packet is sent to a priority queue for processing, depending on its IP counter value accumulated. Three packet classes are built with 3 priority queues: HQ, MQ, and LQ in Figure 3. As defined in Eq.(2), those packets with higher IP counter values are sent to the high-class queue (HQ), which have the least chance to be dropped. Packets entering the medium-class queue (MQ) have medium chance to drop. Finally, those packets sent to the low-class queue (LQ) will be dropped first, because they have survived none or only a few normal windows.

Newly arrived packets from a new source IP are always put in the LQ queue. As time elapses, the later packets from the same IP may be placed to the higher MQ or HQ queues with the increase of the IP counter readout. We assign a forwarding chunk size to these priority queues: \( w \) packets from HQ, \( x \) packets from MQ, and \( y \) packets from LQ queue at a time, where \( w > x > y \) and the chunk sizes are chosen by the filter designer based on security policy enforced. When the RB is full, the router drops packets in reverse order, until the RB overflow is avoided.

Let \( \delta \) be the maximum value of each IP counter. Consider an incoming packet associated with the \( IPC_i \) counter. During the normal traffic, the classification of the packet to the three priority classes is handled with the following thresholds on the counter readout.
The packet goes to the
\[\begin{align*}
 LQ \text{ queue, if } & \text{IPC}_i < 0.25 \delta \\
 MQ \text{ queue, if } & 0.25 \delta \leq \text{IPC}_i \leq 0.5 \delta \\
 HQ \text{ queue, if } & \text{IPC}_i > 0.5 \delta
\end{align*}\] (2)

When abnormal traffic is detected, the router drops all packets associated with IPC$_i = 0$ from the LQ queue and downgrades all IPs in class M to class L and all IPs in class H to class M. When an IP timer up, that IP is removed from the AIP table and corresponding packets flushed out of the router buffer. The threshold values, $0.25\delta$ and $0.5\delta$, are chosen here for illustrative purpose. The actual choice is up to the designer’s experience and priority considerations.

**Life Cycle of an IP Address:** The life cycle of each IP address is explained in Figure 4. All IP address starts with the low-class state (L). It will be switched to the medium-class state (M) and high-class state (H), when the IP counter has reached the above threshold regions. These transitions are marked by a. The reverse downgrade transitions are marked by b. An IP is down-graded to a lower class (H to M or M to L), every time an abnormal traffic is detected. The corresponding IP counter will be reduced to assume the upper bound of its class range defined in Eq.(2).

Under normal traffic, the L, M, and H are active states of the IP address. The IP state becomes inactive when the IP timer is expired as marked by c. All inactive IP addresses will be dropped out of the AIP table. At the low-class state, a new IP address is considered insecure and put in the IPinSec list, if its IP counter is zero during abnormal traffic. This zero count indicates the IP has not experienced any normal window yet. These are labeled as d for suspicious attack or a verified DDoS attack.

**Figure 4** Life cycle of an IP address from creation to becoming active or inactive, or being labeled as an insecure IP address

**Life Cycle of a Typical Packet:** In Fig. 5, there are five possible states during the life time of a packet. The packet is in the “invalid” or “insecure” state, if its IP address will be dropped from AIP table as marked by the transition e. There are three possible working states of a packet, labeled as HQ, MQ, and LQ, where the packet
is currently residing. From these states, the packet will be dropped if the specific priority queue has an overflow, as marked by $f$ in the diagram. The condition $f$ also includes the situation when the router buffer (RB) has an overflow. When the packet is at the front of its residing priority queue and ready to be scheduled, it is forwarded to the RB first before it is sent to the external server by the PQS, as marked by $g$.

![Diagram of packet lifecycle](Figure 5 The life cycle of a typical packet from an identified IP source to end up being forwarded to a server or dropped by the router)

**Implementation Considerations:** To serve the trusted IP sources, the router should make fast network connections upon user requests. In general, we use a larger buffer for the high-class queue (HQ), a medium-size buffer for MQ, and a smaller buffer for LQ. The size of the router buffer (RB) should be sufficiently large to accommodate all good users and avoid easy overflow. In fact, the larger is the RB, the better the service it may provide. The sustained QoS of the filtering router is also affected by the round-trip time and the link bandwidth applied. The IP counters and timers are implemented with dedicated memory tables.

Based on our experience with the end router at USC computer center, the normal traffic at USC has a peak rate close to 1000 $pps$. The active IP table (AIP) was initially chosen to have a size of 1024 entries to accommodate local users logging onto several servers on USC main campus. This implies that the number of IP counters used is rather limited. The IP timer limit was set to match with the new requests (CRS) arrival rate. The longer timer limit may lead to an AIP overflow problem. The short timer limit may affect the sustained QoS. Some of the design parameters are constantly refined in our simulation process.

**An Illustrative Filtering Example**

The above idea is best illustrated by the following example. The entries of a given AIP table are shown in Table 2. We consider a maximum count of $\delta = 256$ in the IP counters. In this case, high class IP addresses have counter readout in the range (128, 255). Medium class counters in the range (64, 127), Low class counters in the range (0, 63). Consider a total monitoring period of 30 minutes and each observation window is 5 minutes. The last column shows whether the timer has expired or not.
**Before The Attack:** Consider the high-risk case of a burst of traffic with an average traffic rate exceeding the threshold $\beta = 300$ **pps**. Assume that the two low-class IP addresses: 128.125.7.11 and 128.125.85.75 in Table 2 have large number of packets amid the burst of traffic. With the help of the anomaly detection to be presented next, these two IP addresses are detected as possible DDoS attacks. These two IP entries have not experienced any normal windows indicated by the zero readout in their respective IP counters. Therefore, they are the first two candidates to be dropped.

**Table 2**  **Active IP Table Used by a Network Router to Perform Adaptive Packet Filtering**

<table>
<thead>
<tr>
<th>IP Class</th>
<th>Source IP Address</th>
<th>IP Counter</th>
<th>Timer Expired</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>128.125.14.75</td>
<td>255</td>
<td>No</td>
</tr>
<tr>
<td>H</td>
<td>128.125.3.16</td>
<td>254</td>
<td>No</td>
</tr>
<tr>
<td>M</td>
<td>128.125.91.95</td>
<td>127</td>
<td>No</td>
</tr>
<tr>
<td>M</td>
<td>128.125.4.85</td>
<td>125</td>
<td>No</td>
</tr>
<tr>
<td>L</td>
<td>128.125.91.94</td>
<td>45</td>
<td>Yes</td>
</tr>
<tr>
<td>L</td>
<td>128.125.1.112</td>
<td>30</td>
<td>No</td>
</tr>
<tr>
<td>L</td>
<td>128.125.3.94</td>
<td>3</td>
<td>No</td>
</tr>
<tr>
<td>L</td>
<td>128.125.7.11</td>
<td>0</td>
<td>No</td>
</tr>
<tr>
<td>L</td>
<td>128.125.85.75</td>
<td>0</td>
<td>No</td>
</tr>
</tbody>
</table>

**After the Attack:** Based on the condition given in Fig.4, the router drops the packets in low class with IPC = 0. The last two entries in the AIP table are removed to enter the IPinSec list. The IP = 128.125.91.94 is removed from the AIP table for timer expired. After these updates, we have the new AIP table in Table 3.

**Table 3**  **The Active IP Table after the Detected Flood Attack Amid a Burst of Abnormal Traffic Surges**

<table>
<thead>
<tr>
<th>Class</th>
<th>Source IP Address</th>
<th>IP Counter</th>
<th>Timer Expired</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>128.125.14.75</td>
<td>255</td>
<td>No</td>
</tr>
<tr>
<td>M</td>
<td>128.125.3.16</td>
<td>127</td>
<td>No</td>
</tr>
<tr>
<td>L</td>
<td>128.125.91.95</td>
<td>63</td>
<td>No</td>
</tr>
<tr>
<td>M</td>
<td>128.125.4.85</td>
<td>125</td>
<td>No</td>
</tr>
<tr>
<td>L</td>
<td>128.125.1.112</td>
<td>30</td>
<td>No</td>
</tr>
<tr>
<td>L</td>
<td>128.125.3.94</td>
<td>3</td>
<td>No</td>
</tr>
</tbody>
</table>

In Table 3, we see the downgrade of the IP = 128.125.3.16 from high class to medium class. Similarly, the IP = 128.125.91.95 is lowered from class M to class L. These were triggered by the fact that these two IP sources were sending packets during the traffic surge period. After the attack, some of the IP priority classes.
are lowered because of suspicion of their participation in the DDoS attacks. In total, three IP entries are removed from the AIP table along with all packets originating from them.

**Performance Analysis:** In Table 2, we throw out the IP = 128.125.91.94 for being idle for too long. This leaves 8 IP addresses to consider. Assume that the router wants to keep 500 pps in the process by dropping some packets. Also, assume each IP source sends 100 pps to the router with a uniform arrival rate. Thus, the total incoming traffic rate is 800 pps. This implies that we have to drop packets with a rate of 300 pps. The probability of dropping good IP packets is thus estimated as 1/6 = 0.17 under the above assumptions. By Eq.(1), we have a very rough estimation of the sustained $QoS = 1 - 0.17 = 0.83$.

As a contrast, with the traditional static filtering, the good packet drop rate is estimated at 3/8 = 0.375 by dropping 300 packets from 8 IP sources uniformly. Thus, the sustained $QoS$ could be degraded to as low as 1- (3/8) = 0.625. This shows a more than 20% improvement in performance of our adaptive filtering over the static filtering under uniform packet distributions. We can expect even better improvement, if the bad IPs are sending more packets than good traffic to the router.

**Other Filtering Approaches:** One can use the queue length (the sizes of the priority queues or the router buffer) to refine the design of the adaptive filter. Queue length limits the capability of the router to handle incoming packets. The LQ queue is the most relevant indicator to identify the DDoS attack, because this queue accepts the newest IP addresses. Most spoofed IP addresses are first assigned in the low class. Our approach is complementary to the QoS regulation work by Garg and Reddy [3]. The pushback method by Ioannidis and Bellovin [8] offers another interesting solution to this problem.

Instead of using the total traffic rate, we can also use the CRS rate or the EAC rate alone to detect traffic irregularity. The idea is to check whether these rates exceed a given threshold. The EAC covers all open services requested to the router. We will describe how to use the CRS rate to perform anomaly detection shortly. Using these parameters, one can reduce the filtering or detection overhead, because the EAC count or the CRS arrival rate are much lower than the total connections counted or the total traffic rate experienced.

**Datamining for Anomaly Detection**

Anomaly-based detection demands protocol dynamics and multi-site correlation [11, 12]. If the attack is detected correctly, then an effective response is enabled in time to stop the attack. All attacks and their responses are stored in the database. The recorded information is provided to match the attack profiles. **Attack profiles** are the combination of the protocol used, traffic rate, port used, time interval between packets sent, etc [10, 13].

**Security Database Construction:** The security database cannot be built overnight. It is constructed incrementally over the entire life span of the network platform being protected. For flooding detection, we must adjust the detection threshold frequently. As shown in Figure 6, the packet filter eliminates some malicious packets. The security database stores all footprints of previous attacks, including all past attack patterns, the countermeasure deployed, and their effectiveness.
The database trainer sorts out the attack profiles for use in detecting new attacks in the future. Signature-based detection has fewer false positives and lots of false negatives. For new attacks, signature matching becomes ineffective to detect the DDoS attacks correctly. In an anomaly-based IDS, one can handle the new attacks by checking the attack profiles and generate some appropriate responses [10, 14]. Any significant deviation from the profile is reported as suspicious attacks. New network conditions should be also added to the profile. The traffic monitor is often an integrated part of the packet filter.

**Figure 6** An anomaly-based intrusion detection system (IDS) using datamining to train attack profiles and to drop malformed packets

**Protocol Anomaly Detection Process:** We propose a new protocol anomaly detection algorithm using the security database. Datamining can help anomaly detection in many ways as originally suggested by Noel. et al [14]. This method creates general representations of DDoS attacks. Datamining discovers strong associations with well-known protocol standards. The classifier induces the attack classes, based on attributes from training data associated with known attacks.

Figure 7 gives the detailed profile matching and subsequent IP and traffic checking for revealing abnormal behaviors. This method compares the profiles of the incoming traffic with those in the database. If a match is found, the received packet is a verified attack. Partially matched profiles are classified as either suspicious attacks or no attacks. The thresholds used in the flowchart are determined by long time observation and analysis of the traffic spectrum.

For an example, the HTTP protocol allows an application to use the shortest UTF-8 Unicode strings. In case of a Nimda attack, the network worm exploits directory traversal vulnerability using some overlong UTF-8 characters. This is a typical protocol violation checked by the top decision box in Fig.7. If the source IP of the incoming packet is in the **IPinSec** list, the attack is verified as a Nimda attack from its source IP address. The traffic shows an irregular behavior with a sharp surge of packet rate in the spectrum.
Figure 7  Protocol anomaly intrusion detection aided by datamining of attack profiles

The third decision box checks the IP redundancy problem. If the $CRS$ arrival rate exceeds a threshold rate $\Omega$, it is verified as an IP redundancy from the same source. The sum $NOR + EAC$ accounts for both yet-to-be-established ($NOR$) and established connections ($EAC$). The rate of established connections in $EAC(\theta)$ is a function of the detection threshold $\theta$ applied. We separate abnormal from normal traffic rates by checking the flowing condition, where the $MAX$ is determined by the buffer size and router capability.

$$NOR + EAC(\theta) > MAX$$

(3)

Alarm Classification: In Figure 7, there are three conditional paths leading to a verified $DDoS$ attack: (1) The protocol file violated established protocol standards. (2) The source IP is in the insecure IP list. (3) Source IP redundancy is detected even the packet is not in the $IPinSec$ list. A packet is a suspicious $DDoS$ attack in two possible ways: (1) The requested connections exceeded the maximum number allowed. (2) Oversized fragmentation has occurred as explained below.

The maximum size is 576 octets for a datagram accepted by most hosts. If the IP packet size is larger then 576 octets, the host cannot resemble them correctly. This is called an oversized fragmentation, often caused by protocol violations. This will confuse the OS and brings the system to a shutdown. When no IP redundancy is detected, either a suspicious attack or a no attack may occur. Following the “no” paths in all five decision boxes in Figure 7 leads to a no attack classification.
Performance Assessment Model

We use an alarm matrix to keep track of various attacks launched and the corresponding alarms raised by the IDS. Sometimes, the alarm is raised correctly to identify the right attack and sometimes they are confused. This model is extended from the confuse matrix concept, introduced by R. P. Lippmann and J. Haines at MIT Lincoln Laboratory (see side bar in [5]). This alarm-matrix model is generated by an IDS report and verified by datamining and learning mechanisms, after a long period of monitoring the traffic conditions. The matrix can be used as a predictor of the future detection behavior.

Alarm Matrix: The detection results of an IDS are represented by a square alarm matrix \( A = (a_{ij}) \) of order \( n + 1 \), where \( n \) is the number of distinguishable DDoS attack types. The rows correspond to various attacks and the columns are the corresponding alarms raised. The matrix element \( a_{ij} \) is the number of times that attack type \( i \) has triggered the alarm type \( j \). The last row is for no attacks which lead to false positive alarms. The last column indicates the false negatives caused by missed detection. For simplicity, we show below a \( 4 \times 4 \) alarm matrix corresponding to three attack types.

\[
\begin{bmatrix}
    a_{11} & a_{12} & a_{13} & a_{14} \\
    a_{21} & a_{22} & a_{23} & a_{24} \\
    a_{31} & a_{32} & a_{33} & a_{34} \\
    a_{41} & a_{42} & a_{43} & a_{44}
\end{bmatrix} = A \quad \text{(Alarm Matrix)} \quad (4)
\]

The four rows correspond to 4 attack types as labeled. The first 3 columns are the alarms raised. The 4-th column shows the false negatives from missed detection. The detection hits are represented by all diagonal elements. The misclassified alarms are recorded by the off-diagonal elements. The false positives correspond to entries at the last row. Not that the corner entry \( a_{44} = 0 \), because no alarm will be raised for no attacks at all.

**Attack Port Numbers:** We use signature-based *Snort* to detect the communication between the handler and Zombies. Anomaly-based detection is used at network router. Specific communication ports are specified in Table 4 for four attack tools and protocols used. These port numbers are used to distinguish the DDoS attacks of various types.

### Table 4 Communication Port Numbers used by the Attack Tools

<table>
<thead>
<tr>
<th>Attack Tools</th>
<th>Communication Port Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trin00</td>
<td>1524 TCP, 27665 TCP, 27444 UDP, 31335 UDP</td>
</tr>
<tr>
<td>TFN</td>
<td>ICMP Echo, ICMP Reply</td>
</tr>
<tr>
<td>Stacheldraht</td>
<td>16660 TCP, 65000 UDP, ICMP Echo, ICMP Reply</td>
</tr>
<tr>
<td>Shaft</td>
<td>20432 TCP, 18753 UDP, 20433 UDP</td>
</tr>
</tbody>
</table>
Proactive Intrusion Defense Against DDoS Flooding Attacks

(Not for distribution or attribution: for review purposes by IEEE Security & Privacy Magazine only)

Four Attack Scenarios: To demonstrate the use of the alarm matrix for reporting IDS results, we consider four alarm matrices, H, Z, R, and V. The matrix entries are based on a good understanding of the detection behaviors of the handler, the zombie, the router, and the victim platforms, respectively. In Table 5, we summarize the platform vulnerabilities, the IDS behavior, and alarm matrix characteristics.

<table>
<thead>
<tr>
<th>Target Platforms</th>
<th>Intrusion Detection System (IDS) (FP: false positive, FN: false negative)</th>
<th>Alarm Matrix Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handler command the Zombies to attack</td>
<td>Use Snort, the TFN attack differs from Trin00 and Shaft attacks. Both false positive and false negative alarms exist</td>
<td>Matrix H: High detection rate, moderate false alarm rates, and no confusion between Trin00 and Shaft</td>
</tr>
<tr>
<td>Zombie exploited to launch attacks on the victim</td>
<td>The Snort is less effective here, but false alarms are fewer, Stacheldraft attack is easier to be confused with other two attack types</td>
<td>Matrix Z: Low detection hits and lower false alarm rates, high misclassification rate</td>
</tr>
<tr>
<td>Internet router used to launch DRDoS attacks</td>
<td>Anomaly-based IDS using a low threshold to enforce high security at expense of router throughput, no misclassifications</td>
<td>Matrix R: High detection hit rate, no missed detections, and no confused detection</td>
</tr>
<tr>
<td>Server or Client Host targeted as the attack victim</td>
<td>Use a signature-based IDS, no false positive alarms, and no misclassified alarms due to use of different protocols on 3 attack types</td>
<td>Matrix V: High detection hit rate, no false positive and misclassification, but high false negative rate</td>
</tr>
</tbody>
</table>

An Example Alarm Matrix: In [7], we have considered 4 hypothetical alarm matrices. For illustrative purpose, we show below the alarm matrix Z for the zombie machine. The zombie machine is not the victim of the attack, just serving as an intermediate agent to launch the attack under the command of the handler. The Zombie IDS is unable to detect the communications between zombies and the handler. This matrix Z corresponds to three DDoS attack types: Trin00, TFN, and Stacheldraft.

\[
\begin{pmatrix}
5 & 0 & 10 & 1 \\
0 & 3 & 6 & 2 \\
8 & 20 & 4 & 2 \\
1 & 2 & 1 & 0
\end{pmatrix} = \text{Alarm Matrix } Z \tag{5}
\]

The matrix elements show the attack/alarm frequencies detected. For examples, the Stacheldraft attack has been detected 20 times and wrongly alarmed as the TFN attacks. The zombie IDS is confused with large nonzero entries on the off-diagonal elements. The Stacheldraft attack tool is easy to be confused by the IDS, because it uses the same ICMP and TCP protocols that are used by the Trin00 and TFB tools.
The above alarm matrix has nonzero entries distributed in all regions. The handler machine has a similar matrix. The router matrix has zeros in all off-diagonal and the rightmost column. For the victim machine, the bottom row and off-diagonal elements are zeros [7]. Different matrix distributions correspond to different platform and IDS behavior as summarized in Table 5. The matrix elements need frequently updated at different monitoring periods.

Analysis of Detection Results

Consider an attack of type \( i \), the \textit{attack frequency} \( F_i = a_{i1} + a_{i2} + a_{i3} + a_{i4} \) is the row sum of matrix elements. Similarly, the \textit{alarm frequency} \( G_j = a_{1j} + a_{2j} + a_{3j} + a_{4j} \) for the \( j \)-th alarm type equals the sum of elements in the \( j \)-th column of the alarm matrix. The relative magnitude of the two frequencies varies with the IDS/platform applied. If \( G_j > F_i \), an IDS may be cautiously designed to over-kill with many false-positive alarms. An ineffective IDS may have missed many attacks such that \( G_j < F_i \).

Detection and Alarm Rates: Using the alarm matrix in Eq. (4), we define 4 performance expressions. For attack type \( i \), we compute the \textit{detection hit rate} by \( H_i = a_{ii}/F_i \) and the \textit{detection miss rate} or the \textit{false negative rate} by \( M_i = a_{i4}/F_i \). The \textit{false positive rate} for the \( j \)-th alarm is given by \( S_j = a_{4j}/G_j \). The \textit{misclassification rate} is determined by the off-diagonal matrix elements using the expressions: \( T_1 = (a_{21} + a_{31})/G_1 \), \( T_2 = (a_{12} + a_{32})/G_2 \), and \( T_3 = (a_{13} + a_{23})/G_3 \) for 3 attack types. Figure 8 plots these detection and false alarm rates for the 4 attack scenarios specified above.

Performance Results: Examining the four plots in Figure 8, we have the following observations on their relative performance. These results are analytically obtained using the above equations for \( i \) or \( j \) = 1, 2, and 3 attack/alarm types. The height of the bars indicates the detection hit, miss, and false alarm rates.

\begin{itemize}
  \item[a.] Among the four cases, the network router has the perfect detection rate with 100% hits for all 3 flood attacks. The handler has also high detection rate (58-80%). The Zombie has lower detection rate less than 26%. This is considered a typical performance rating of innocent hosts being drafted unwillingly to serve as agents in DDoS attacks.
  \item[b.] The Zombie machine has the highest rate of misclassification. This implies that the Zombie is quite confused among three attack types: Trin00, TFN, and Shaft. The false alarm rates in all cases are relatively low. The router has encountered some false positives but no false negatives or misclassifications.
  \item[c.] The victim machine has moderate detection rate (45%, 75%) and highest miss rate (25%, 55%), but no false alarms and misclassifications at all. This implies that most victims are vulnerable and they need to be fortified with some powerful host-based IDS to overcome the degraded performance caused by high miss rate in detecting the DDoS attacks.
\end{itemize}
Conclusions and Extended Research

The NetShield Protect is primarily designed to protect local or enterprise networks. The system is scalable and expandable to secure upstream ISP networks for distributed grid computing. To sum up, we highlight below major findings from this work and present important lessons learned so far. The adaptive filter design, profile datamining and mitigation, and protocol anomaly detection offer distributed intelligence and comprehensive defense against known and unknown flooding attacks.

A. We filter out malicious packets and block the DDoS traffic at two levels in the NetShield system. The first level is filtering at the router and then anomaly detection at the firewall/IDS level. This will reduce the overhead of executing filter on the interface. The effects of filtering out malicious packets at the expense of slowdown the flow of normal packets should be further studied. Both overkill and
undercut are not acceptable. The gain of smart filtering through differential QoS regulation is also an open problem worthy of further study.

B. We have pursued protocol anomaly detection, which is improved from signature anomaly detection by modeling well-defined protocol standards. Instead of using a misuse detection model, our approach applies a normal-use defense model, which requires no signature updates, a major advantage in reducing the intrusion detection overheads. We need to reveal the relative merits between these two detection schemes through benchmark experiments.

C. We have proposed to apply datamining to automate the protocol anomaly detection process through machine learning. This is still a wide open area. Risk assessment is supported by datamining of logged security records. The security database should have a learning capability from the past attacks to face future unknown attacks. Our continued research intends to answer some of the open issues.

D. The alarm-matrix model is useful to evaluate any IDS-based security systems. The model enables a powerful tool to evaluate the performance of intrusion detection process and to streamline the packet filtering process. This model helps implement the adaptation processes in changing detection or filtering thresholds and in dynamic updating of the security policies.

Acknowledgements: We would like to thank Yue Chen of USC Computer Science Department for technical discussions leading to the development of the alarm matrix concept, which was inspired by the MIT/LL Intrusion Detection Evaluation Report in 1999. We also appreciate the material and network support from USC Electrical Engineering Department in the construction of the NetShield testbed at USC Internet and Wireless Security Laboratory.

References:


**Biographical Sketches:**

**Kai Hwang** is a Professor of Electrical Engineering and Computer Science and the Director of the Internet and Wireless Security Lab at the University of Southern California. An IEEE Fellow, he specializes in computer architecture, parallel processing, Internet security, and grid computing. Dr. Hwang has published numerous papers and books in these areas. Presently, he leads a research group at USC developing new Internet security architectures and distributed intrusion detection and response systems for cluster and grid computing. He can be reached at kaihwang@usc.edu

**Sapon Tanachaiwiwat** is presently pursuing Ph.D. degree in the Electrical Engineering Department at the University of Southern California. He has received his B.S. degree in Electrical Engineering from the Mahidol University in Thailand and M.S. degree in EE from USC. His current research interest includes Internet security and distributed Intrusion detection and response. He can be reached at tanachai@usc.edu

**Pinalkumar Dave** is completing his M.S. degree in Electrical Engineering at the University of Southern California in May 2003. He will continue pursuing the Ph.D. degree at USC. He has received his B.S. degree in Department of Electronics and Communication, Gujarat University, India. His current research interest includes wireless and ad hoc network security and distributed grid computing. He can be reached at pdave@usc.edu