Community aware network security and a DDoS response system

Stamatis Karnouskos

Fraunhofer FOKUS - Institute for Open Communication Systems
Kaiserin-Augusta-Allee 31, D-10589 Berlin, Germany
Stamatis.Karnouskos@fokus.fraunhofer.de

Abstract – *Due to the considerable growth of Internet as well as its usage as a commercial platform, attacks against networks such as Distributed Denial of Service (DDoS) attacks, have emerged, with victims even among prestigious commercial sites. Such attacks in traditional networking are difficult to recognize and to handle. Managing them requires a network that can dynamically detect, share info, respond to event-triggered requests and proactively secure itself. We present here a community aware network security as well as hands on experience with a specific threat i.e. a DDoS scenario and attack response system approach. We demonstrate the dynamicity and flexibility of the community-aware networks in dealing with this kind of threats. The implementation is based on agent-enabled active networks and makes heavy use of the mobile agent technology in order to asynchronously respond to critical situations. Finally we comment on the pros and cons of our approach and discuss future directions that could be followed.*

Keywords: *Intrusion Detection System, Community-aware Network Security, Distributed Denial of Service Attacks, Agent Technology, Active Networks, XML Digital Signatures, PKI.*

1. Introduction

Distributed Denial of Service (DDoS) attacks launched against prestigious commercial sites such as Yahoo!, Amazon, eBay Inc., Buy.com and others have attracted a lot of publicity. The problem is well known to the networking community and difficult a) to recognize early enough and b) to handle. This occurs because not enough attention was paid initially in security matters in Internet, and was only added as an afterthought. Therefore we are only left with a collection of best practices and tools in our quest for network security.

In this paper we primarily discuss how new technologies can team-up with innovative concepts and point us to a direction to which the author believes could be the next step in handling of security attacks such as DDoS attacks. Therefore we will first analyse the technologies and concepts in order to set the context for DDoS attacks. Then we will describe our efforts in implementing a prototype based on the synergy of the proposed technologies, and via a scenario we will demonstrate its functionality. Finally we will comment on the approach and propose directions that could be followed in future work.

2. Technologies and concepts

We present here the major technologies and concepts whose combination could offer us a new level of dealing with network security attacks and specifically in defending against the DDoS ones. Community aware Network Security sets the concept area in which two action lines are fundamental, namely:

- Detection of the attack
- Efficient response to the attack

Both of the above should be done in an intelligent way, whereby intelligence as well as detection and action teams collaborate while roaming the network and are not static. For the first, an evolution of Intrusion Detection Systems [34][35] would be a good start. For the second we have to combine and deploy new technologies such as mobile agents and active networks in order to better realize our goals.

2.1 Community aware Network Security

Securing a network nowadays is synonymous with hardening of its services and potential security threats. However this approach makes the network inflexible and blurs the line between security and usability. Networks are highly complex and require too much human interaction in the administrative level in order to be secure. Furthermore no common base exists among various security solutions available in the market. In other words, available products do not communicate with each other (interoperate) and work alone for their own and their distribution company's good and not necessarily for the user's network. Community aware tactic may offer a better alternative. Adopting modelling approaches from the evolution of biological systems [36], they are seen as networks formed from cooperating living parts that interoperate at various levels and share information. This

results that DDoS attacks may be better handled within such an infrastructure as we are able to combine approaches and techniques from various standalone (like in the current networks) components, take more complex decisions and form more efficient action plans. Such networks, like living systems, are envisioned to deal with intrusions and pre- and post- protect their parts. In these networks *detection* and *response* on the one side but also adaptation of security as a precaution and not necessarily as the result of an attack on a specific node, constitute the driving force of dealing with complex intrusions e.g. a DDos attack. Community networks are not only aware of their standalone node status, but may share info about the whole network situation or at least the neighbouring parts. In general as all living organisms, they promote a different view of security by maximizing flexibility as long as the mechanisms to detect violations and rapidly respond to them exist.

One of the major problems that make difficult the migration from the current approaches to the community aware networks is interoperability. Vendors do not necessarily want to interoperate for various reasons, and even if they agreed to a common mechanism so that their products talk to each other, the standardization process would take a long time to achieve what we want, and even then we will not be sure if we have a result expressive enough to cover the future needs of network security related products' inter-communication. Active Networks [5] offer an advanced infrastructure that can be used to realize the community aware systems as described here. In active networks, we do not have anymore an underlying passive infrastructure but programmable network resources via high level Application Programming Interfaces (APIs) for the next generation of network aware applications. Furthermore they manage heterogeneity in hardware and software matters by allowing the deployment of non-standardized approaches that can be wrapped via the widely accepted high-level APIs. This can be used to tackle our interoperability problems as stated above.

2.2 Distributed Denial of Service Attacks

Denial of service attacks (DoS) are attempts to overwhelm a service with requests, resulting to rejection of legitimate requests. If more than one computers are used, then we have distributed denial of service attacks which are more difficult to deal with and their effect is magnified in comparison to simple DoS attacks. Several highly sophisticated tools [1] such as Smurf and Trinoo but also modern ones like Tribe Flood Network (TFN), TFN2K, Stacheldraht, Mstream and Shaft make such attacks easier than ever before.

As shown in Figure 1, behind the *Client* is the person that orchestrates the whole attack. The *Master* is a compromised host, which runs the software that controls several *Daemons*. The *Daemon* is also a compromised host, running the program that generates a stream of packets (the malicious flow) towards the intended victim. The attacker first initiates a scan phase in which a large

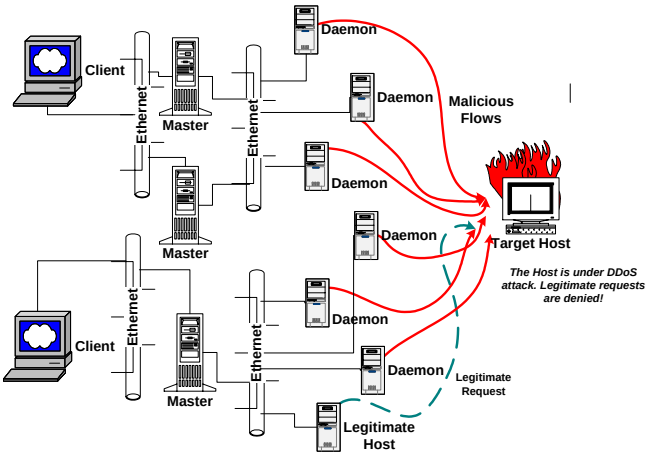

Figure 1. Distributed denial of service attack

number of hosts is probed for known vulnerabilities. Once these vulnerabilities are identified the host is compromised and the malicious software is installed. After this initial step the whole process is magnified since compromised hosts are used for further scanning and compromises. As this process is automated via the use of scripts, several thousands of hosts can be compromised in very little time i.e. an average time for the whole process (full scan for vulnerabilities and installation of malicious code) could be as little as seven seconds per host.

2.3 Agent Technology

Agents [16] are software components that act alone or in communities on behalf of an entity and are delegated to perform tasks under some constrains or action plans. The mobile agent paradigm is a fascinating one for design and implementation of dynamic distributed systems as they can autonomously transport themselves from node to node and continue their execution. Mobile agent technology has established itself as an improvement of today's distributed systems due to benefits such as dynamicity, on demand provision and distribution of services, reduction of network traffic and dependencies, fault tolerance etc. The number [6] of mobile agent platforms coming from the commercial sector, as well as the academia is increasing day by day.

2.4 Active Networks

Active Networks (ANs) [5] consist an evolution of current dumb passive network carriers, where the level of

abstraction is the protocol, to a more general programmable network model where the level of abstraction is raised to application programming interfaces for programming the new network resources. The idea is to move service code, which traditionally was placed outside the transport network, directly to network's nodes. Those nodes allow applications to configure them optimally for their tasks via open interfaces (programmable networks). Furthermore, those nodes will be able to compute on data they receive before they pass them to the next node (active networks). Active and programmable networks or their combination [19] and generally network-aware software, is expected to change the way we design and deploy applications and services. While network programmability and the capabilities it offers is attracting and with increasing interest within the research community, its state of development is still at an early stage.

3. Current Protection from DDoS

Protection for such kind of attacks can be achieved at various levels primarily by deploying i) Firewalls, ii) Intrusion Detection Systems and iii) Network Pumps.

Firewalls are the main line of defence against attack from the outside. Their purpose is to enforce the security policies defined, reflecting the decisions about which Internet services we want to be accessible from whom and from where. Firewalls are packet-filtering based therefore they represent a static router with traffic screening rules enforcing local policies concerning which packets are allowed through the network interfaces. Configuring a firewall to block traffic based on source, prevents novice well known attacks. By limiting the type of traffic allowed to pass through we reduce the attack set that intruders can operate on.

Intrusion Detection Systems *(IDS)* monitor the network for a series of events that occur prior to an attack (known as attack signatures), e.g. the Ping of Death attack has a signature of an ICMP echo request with payload over 4000 bytes. If the IDS detects a packet like this it generates an alarm. Advanced IDSs have the ability to learn attack signatures in real time, but produce many false reports. Furthermore IDSs are mostly static and in order to add new functionality or even reconfigure them, we must take them offline and restart them. This is an inflexible monolithic approach that cannot deal with the challenges set by current dynamic infrastructures. The network must have the ability to dynamically change its behaviour based on the status of external events. If a DDoS attack is initiated, for most commercial companies time of reaction is critical as they loose thousands of dollars and therefore the response time should be as low as possible. In order to achieve that, we need networks that can not only sense the environment and react to its changes but also share knowledge and collaborate.

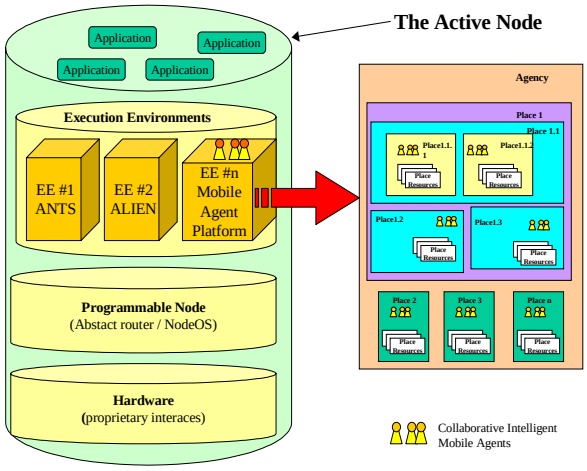

Figure 2. The Agent-based active network node architecture

The Network Pump *[23][33]* is introduced in order to protect multi-level secure systems in which devices have different priorities. The network pump makes sure that high and low level security systems stay interconnected, while preserving confidentiality of the high (and more important) level systems. The aim here is to provide some level of quality of service between systems, coupled with reliability (even in case of power failures and system crashes), flexibility (customized interfaces), low cost (initial and operating) and high performance.

The existing solutions presented here shortly, are expected to advance and are seen as an integral part of a community aware system. IDSs are used to detect the anomalies and send out the alerts. It is up to the network then to decide based on an IDS' alert what to do next and how to deal with this incident on a local and network-wide level. Community aware networks share info and cooperate in order to respond proactively to unacceptable behaviour. Furthermore, the borders of the network are extended, as within such a network its parts (components and other segmented networks) are able to interoperate. This is also fundamentally different to the current approaches where the firewall marks the network border. Firewalls are still useful in such an infrastructure but their functionality has to be seen as part of the whole, as they are now part of the security components within a living network.

4. The Agent-enabled Active Network Node

The active node architecture with the agent execution environment (EE) is depicted in Figure 2. An active node (router, switch, etc) can be realized via the composition of three different layers, representing hardware and software parts i.e. the static part, the programmable part and the active part.

Static part: This is the hardware that is delivered by the manufacturer. It contains the optimised components and algorithms implemented in their hardware form for performance reasons.

Programmable part: This part integrates the manufacturer proprietary interfaces of the fixed part and exports an open standardized interface. The APIs are under standardization by the IEEE P1520 project [8]. At this level the node can be programmed but only via a parameter specific approach (programmable node). The Node Operating System (NodeOS) [9] provides the basic functionality from which the execution environments built the abstractions presented to the active applications.

Active part: The full ability of programming the node is unfolded here as this part hosts several execution environments that allow, via code injection and execution, sophisticated programmability of the node. As also noted in [10], the functionality of the active network node is divided among the NodeOS, the Execution Environments and the active applications. The architecture allows multiple EEs of various providers to co-exist and be present on a single active node. Each EE (e.g. ANTS [11], ALIEN [12], Agent EE) exports a programming interface or virtual machine that can be programmed or controlled by third party code. The mobile agent EE is where agents execute when they visit the node. The applications are able to access all the services offered by the EEs. Usually an application is bounded to one EE but we can foresee applications that will take advantage of the various characteristics of more than one EEs and possibly combine their services. The FAIN project [17] has developed such a multi EE architecture including a mobile agent EE based on the Grasshopper platform [3].

As shown in Figure 2, one of the EEs is the agent execution environment. This is the agency as described within the MASIF [13] standard. The agent system consists of Places. A Place is a context within an agent system in which an agent is executed. This context can provide services/functions such as access to local resources etc. Cooperating agents reside in the agent-based EEs and via the facilities offered to them program the node. These can be either mobile agents (e.g. visiting agents) or even stationary intelligent ones that reside permanently in the EE implementing various services. The integrated approach of agents and active networks allows us to apply several security techniques at the network programming level [14] that promote service and network security. Further info on the architecture presented here and its security issues can be found in [20].

5. System Architecture

The architecture of our DDoS response system is depicted in Figure 3. It is composed of the following parts:

Monitoring System (MS): This part is responsible for analysing and capturing all data that passes via the network interface. The data can be filtered prior to capturing, based on the filtering rules issued by the Attack Detection System (ADS). The MS offers back to the ADS a customized snapshot of the network traffic i.e. the raw data that will need to be further examined by the ADS.

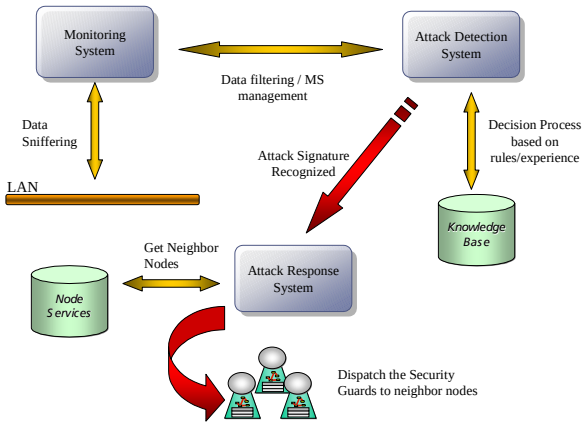

Figure 3. System architecture

Attack Detection System (ADS): This component is responsible for identifying the attack against the node. It features its own decision process based on internal rules, heuristics and expertise stored in its knowledge base. The trigger event identifying a DDoS attack could be dependent on one single event e.g. over the normal presence of SYN packets, oversized ICMP and UDP packets, connectionless TCP /UDP packets, or a result of many similar events indicating abnormal network activities e.g. amount of bandwidth exceeds a maximum threshold that is expected by normal traffic. Pattern recognition is the most well known method primarily to recognize existing DDoS tools and attempts (as any known DDoS attack is based on the traditional client-server paradigm) to install them into network nodes. One module of the ADS (e.g. an agent) could implement this functionality. The ADS can be seen as a cooperation of agents that reside within the agent-based EE of the active node and cooperate in order to recognize DDoS attacks based on the filtered data that they get from the MS. This is a component-based approach, and each agent

implements a specific algorithm or method based on which an attack can be recognized (attack signature). In a distributed scenario the ADSs from all nodes can co-operate and push/pull information a) from each other (distributed approach) or b) from a central network point (centralized approach), in order to obtain a network wide view of the situation and act accordingly.

Attack Response System (ARS): The ARS is an event-triggered system. Once informed by the ADS about the attack it organizes the countermeasures against the attacker. It instantiates the agent Security Guards (SGs) and dispatches them to the neighbour nodes with concrete instructions about how to deal with the attack e.g. to block traffic coming from a specific subnet and is directed to a specific port.

Agents: These are the actual actors. The mission of the SGs is to change (autonomously or in cooperation with local residing agents) the node's configuration so that the malicious flow is blocked. Having done that and based on the facilities offered in the remote node, the agent could clone itself, and let the clones transport themselves to the neighbouring machines. Please note that on each node the agents are able to sense the environment e.g. on the fly discover the neighbouring nodes and act accordingly. Also because we do not want to flood the network with SGs, we can constrain them in the number of clones they can create and the hops that they can live. Furthermore the agents can periodically poll a central security guard and either update their goals or just die if the DDoS attack is over or all nodes are protected.

The agents traverse the network and reprogram routers and firewalls. However they can do more than that. They can update components of the nodes' IDS system and keep it up to date or inform about new attack signatures found to the network they traverse. Therefore, once a node is attacked and the signature is new, the agents can rapidly propagate that info to the network and prevent further attacks in other parts of it. This semi-real time updates of customized rules, signatures, code and policies increases the dynamicity of the system and therefore the faster reaction of the network. Furthermore this role of agents is the backbone of the community aware network security we have been trying to realize. The agents are not only the sensors of the network (listen to events) but also the neurons (propagators of info etc) of our biological-like network.

6. Secure Interoperable Communication

We have a living network that is heterogeneous not only in its hardware but also its software components. These elements need to exchange messages in order to interoperate at various levels. As noted, the active networks offer some high level APIs with the necessary abstractions. These APIs are useful for interoperability but unfortunately they are not controlled by a general-purpose security manager that offers its services via a similarly high level API. Security in active networks is still in its infancy within this relatively new research area. There are some services allowing integrity and policy controlled access but they are inadequate and not generalized. From the agent side we do have for communication purposes the FIPA ACL (Agent Communication Language) [27] that allows heterogeneous agents to communicate and some more advanced security features as also noted in [14], including digitally signed agents. This offers us enough tools but again we can not assume that agents as proposed here are the only approach to be taken in future community-aware systems, nor it is sure that every node will be able to parse them and apply homogeneous security services on their actions.

```
<?xml version="1.0" encoding="UTF-8"?>
<AttackInfo xmlns="http://www.fokus.fraunhofer.de/attackinfo">
        <OriginatorNodeName>ddos-1.fokus.fraunhofer.de
</OriginatorNodeName>
        <OriginatorNodeIP>193.174.152.235 </OriginatorNodeIP>
        <AttackType> DDoS </AttackType>
        <Attacker>193.174.33.8,193.174.33.12,195.124.4.29</Attacker>
        <Victim>193.174.152.235</Victim>
        <VictimPort>80</VictimPort>
        <Action>flow-stop</Action>
        <RouterAction>https://www.fokus.fraunhofer.de/ddos/GR2K-
13.xml</RouterAction>
    </AttackInfo>
```

Listing 1 -- XML report on a DDoS attack

Therefore we turn to a general approach by allowing the heterogeneous components in our network to communicate with XML messages. XML is an expressive language that has a well-defined grammar for defining message structures (DTD, XSD) and there have been developed many tools for generating, consuming and working with XML documents (such as parsers). XML documents can be digitally signed [26] in order to guarantee their integrity over an untrusted network such as the Internet and in parallel we can use the authenticated authority that signed the XML document in order to take further authorization decisions. Furthermore XML encryption [25] can address areas not included in the Internet's de facto standard for secure communications TLS/SSL namely a) encrypting part of the data being exchanged and b) secure sessions between more than two parties. This practically allows us to send one XML document carried around by an agent and handle situations where different parts of the same document need different treatment. Listing 1 shows such an XML message generated by the ARS and propagated via agents to the network nodes in order to make them aware of the attack. The message provides attack info to the neighbouring nodes. If the node's policy allows it, the

agent can take several actions e.g. execute and reconfigure the router attached to the active node, install an Execution Environment dependant component that would reconfigure the router, update this node's ADS etc.

```
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
 <SignedInfo>
  <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315"></CanonicalizationMethod>
  <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"></SignatureMethod>
  <Reference URI="#Res0">
   <Transforms>
    <Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"></Transform>
   </Transforms>
   <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"></DigestMethod>
   <DigestValue>noqUQj6HE0+Yd7MxYXWhRsOYSb8=</DigestValue>
  </Reference>
 </SignedInfo>
 <SignatureValue>
  BlvF5oT+XEm7oPo6InN786m0vamC81RriXViC3vWo2zNW8IU6vZFww==
 </SignatureValue>
 <KeyInfo>
  <KeyValue>
   <DSAKeyValue>
    <P>
     /X9TgR11EilS30qcLuzk5/YRt1I870QAwx4/gLZRJmlFXUAiUftZPY1Y+r/F9bow9s
     ubVWzXgTuAHTRv8mZgt2uZUKWkn5/oBHsQlsJPu6nX/rfGG/g7V+fGqKYVDwT7g/bT
     xR7DAjVUE1oWkTL2dfOuK2HXKu/ylgMZndFlAcc=
    </P>
    <Q>l2BQjxUjC8yykrmCouuEC/BYHPU=</Q>
    <G>
     9+GghdabPd7LvKtcNrhXuXmUr7v6OuqC+VdMCz0HgmdRWVeOutRZT+ZxBxCBgLRJFn
     Ej6EwoFhO3zwkyjMim4TwWeotUfl0o4KOuHiuzpnWRbqN/C/ohNWLx+2J6ASQ7zKTx
     vqhRkImog9/hWuWfBpKLZl6Ae1UlZAFMO/7PSSo=
    </G>
    <Y>
     tQrwZlAaOyHQOebXNBMh/OMdbs3SmGnyzBH6JohSFeYO4nbT28F/BnGUvxvMsj5Lyo
     cUSGlUt9z0O8y7J3cPlbRf80nLI+PmiQd1S0XozEQ83tFfIPJYwvlKRx7eNqa1YLEF
     CjUrKlt/UvOzj8xXv0uKLxtSH/7DC1pf1OVfb50=
    </Y>
   </DSAKeyValue>
  </KeyValue>
  <X509Data>
   <X509IssuerSerial>
    <X509IssuerName>CN=Stamatis Karnouskos,OU=FOKUS,O=Fraunhofer,C=DE</X509IssuerName>
    <X509SerialNumber>1079710343</X509SerialNumber>
   </X509IssuerSerial>
   <X509SubjectName>CN=Stamatis Karnouskos,OU=FOKUS,O=Fraunhofer,C=DE</X509SubjectName>
   <X509Certificate>
MIIC1jCCApMCBEBbEocwCwYHKoZIzjgEAwUAMFAxCzAJBgNVBAYTAkRFMRMwEQYDVQQKEwpGcmF1
bmhvZmVyMQ4wDAYDVQQLEwVGT0tVUzEcMBoGA1UEAxMTU3RhbWF0aXMgS2Fybm91c2tvczAeFw0w
NDAzMTkxNTMyMjNaFw0wNjAzMTkxNTMyMjNaMFAxCzAJBgNVBAYTAkRFMRMwEQYDVQQKEwpGcmF1
bmhvZmVyMQ4wDAYDVQQLEwVGT0tVUzEcMBoGA1UEAxMTU3RhbWF0aXMgS2Fybm91c2tvczCCAbgw
ggEsBgcqhkjOOAQBMIIBHwKBgQD9f1OBHXUSKVLfSpwu7OTn9hG3UjzvRADDHj+AtlEmaUVdQCJR
+1k9jVj6v8X1ujD2y5tVbNeBO4AdNG/yZmC3a5lQpaSfn+gEexAiwk+7qdf+t8Yb+DtX58aophUP
BPuD9tPFHsMCNVQTWhaRMvZ1864rYdcq7/IiAxmd0UgBxwlVAJdgUI8VIwvMspK5gqLrhAvwWBz1
AoGBAPfhoIXWmz3ey7yrXDa4V7l5lK+7+jrqgvlXTAs9B4JnUVlXjrrUWU/mcQcQgYC0SRZxl+hM
KBYTt88JMozIpuE8FnqLVHyNKOCjrh4rs6Z1kW6jfwv6ITVi8ttiegEkO8yk8b6oUZCJqIPf4Vrl
nwaSi2ZegHtVJWQBTDv+z0kqA4GFAAKBgQC1CvBkgBo7ldA55tc0EyH84x1uzdKYafLMEfomiFlV
5g7idtPbwX8GcZS/G8yyPkvKhxRIYhS33PQ7zLsndw8htF/zScsj4+aJB3VLRejMRDze0V8g8ljC
+UpHHt42prVgsQUKNSsgW39S87OPzFe/S4ovG1lf/sMLWl/U5V9vnTALBgcqhkjOOAQDBQADMAAw
LQIVAJca/DwG5Azt57vxOLOggg2HV5x3AhQPZrpUlDlpeWF4eUpVSpj0lz0mLg==
   </X509Certificate>
  </X509Data>
 </KeyInfo>
 <dsig:Object xmlns="" xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" Id="Res0"><AttackInfo
xmlns="http://www.fokus.fraunhofer.de/attackinfo">
                <OriginatorNodeName>ddos-1.fokus.fraunhofer.de </OriginatorNodeName>
                <OriginatorNodeIP>193.174.152.235 </OriginatorNodeIP>
                <AttackType> DDoS </AttackType>
                <Attacker>193.174.33.8,193.174.33.12,195.124.4.29</Attacker>
                <Victim>193.174.152.235</Victim>
                <VictimPort>80</VictimPort>
                <Action>flow-stop</Action>
                <RouterAction>https://www.fokus.fraunhofer.de/ddos/GR2K-13.xml</RouterAction>
</AttackInfo></dsig:Object>
</Signature>
```

Listing 2 -- Signed DDoS report of Listing 1

The later actions depend on the access rights that are given to the agent himself. Allowing a piece of code to execute on a foreign network has several security implications, which are more complicated when dealing with multiple administrative domains. For this work it is assumed that Service Level Agreements (SLAs) between multiple administrative domains exist and allow limited control of the infrastructure (e.g. changing router tables) temporarily while there is the threat of a DDoS attack. If the agent itself is not allowed to execute (the administrative domain doesn't trust the code, but it trusts the carried XML DDoS report) the agent delivers the XML report to the appropriate component that takes further action. The report is signed by the administrator of the originating system and therefore is immune to the content changes. The signed XML report of Listing 1 is shown in Listing 2.

In the example above we have used a home-grown definition of documents that can be exchanged. This was done due to the lack of generally accepted templates for reporting a network intrusion such as a DDoS attack. In the future we have to come up with widely accepted schemas and vocabularies that solve the problem of ambiguity. Similar efforts exist today for other activities e.g. the Dublin Core Metadata Initiative [24] has developed a metadata standard for describing a wide range of networked resources. Currently work in this area is carried out in IETF's Intrusion Detection Exchange Format working group [28], which has released documents related to Intrusion Detection Message Exchange Requirements, the Intrusion Detection Exchange Protocol (IDXP), and the Intrusion Detection Message Exchange Format. In the future, these specifications should be integrated in messaging among the components, as this will bring us one more step closer to solving interoperability issues and realizing the seamless exchange of events in a community aware network.

7. Application Scenario

The application scenario, with which we will demonstrate our system, is depicted in Figure 4. The network topology consists of various active nodes (e.g. nodes A, B, C) and legacy nodes (e.g. node D). Each AN node is a combination of a router and a controller PC which runs the software part. In normal operation the agents that implement our system reside within the agencies and filter the flow that is directed to the node. Some time later the attacker initiates the DDoS attack via the compromised hosts against the AN node C. What exactly happens is described below:

- One agent of the ADS in node C, that is a specialist in a specific attack (knows the signature of the attack) detects the pattern and signals the alarm. The ADS then contacts the ARS and provides specific info for the attack that was encountered e.g. subnet, tcp ports etc.
- The ARS dispatches a SG agent that changes the local router table and from now on this router denies access to all malicious traffic targeted to the specified subnet.
- The ARS consults the local node services in order to find out the neighbouring nodes (the nodes that are one hop away). This could be accomplished simply by invoking the *traceroute* utility of Unix systems (to

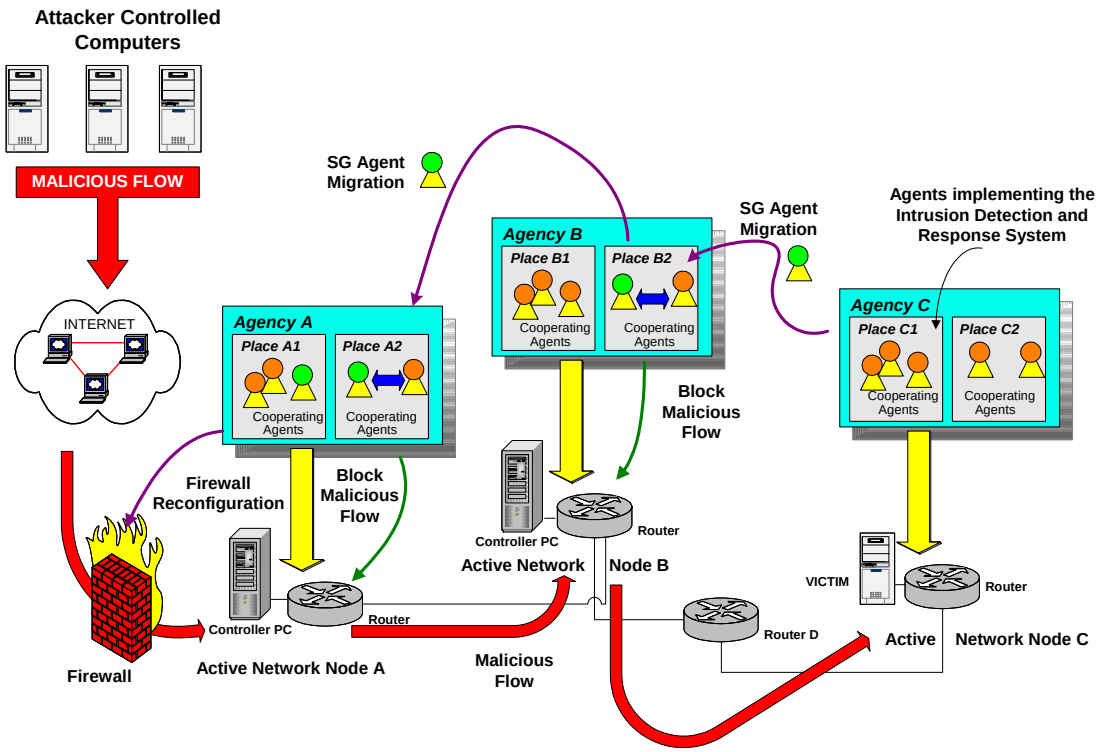

Figure 4. DDoS attack and response Scenario

find the next hop router for that subnet) or by looking up the router tables.

- The ARS sends out the Security Guards (SGs) to the neighbouring nodes with specific goal to block the malicious traffic.
- The SGs transport themselves to their destination AN node B and in the execution environment provided there (Place B2) they continue execution. Having passed successfully the authentication and authorization mechanisms of the visiting node (this is analysed in previous work [20]) they can either change the node policy/configuration by themselves if they have the right to do so, or collaborate with the local agents providing feedback on the attack and eventually blocking the malicious traffic to AN node C. Here note that now the malicious flow is blocked at AN node B and never reaches the rest of the network.
- Subsequently the SGs clone themselves and depending on their (or the platform's) capabilities, they keep on detecting the next hop AN node and transporting themselves there.

Following recursively the steps above, the point where the malicious flow is blocked, is every time getting closer to the source of the problem, or at least pushed up to the network domain boundaries.

This is an incremental approach that goes hop by hop to the estimated source of attack i.e. the *Daemons* as shown in Figure 1. If the attack is ongoing we can in real-time detect and traceback the source of the attack, otherwise, we need to examine the logs in each router. The whole process could fail if one router does not support upstream source identification. However more promising traceback technologies have been proposed [21][22] and can be used.

8. Prototype Implementation

A first prototype of this approach was implemented and demonstrated within the scope of BANG project [2]. Since then several enhancements to the programming and conceptual part have taken place. Our testbed is the same as the one described with the Figure 4. The active nodes consist of Hitachi GR2000 gigabit routers [7] that are managed via a Controller PC. The Grasshopper agent system [3] was selected as a standard compliant (both to FIPA [15] and OMG MASIF [13]) mobile agent platform to be embedded in the control PC. Both ADS and ARS modules are implemented as Java mobile agents. Although ARS and ADS do not need to be mobile, the motivation behind that is that in the future it might be a good idea to allow the whole system to roam the network and partially clone itself for survivability and load balancing reasons. The MS module consists of a modified version of the *ethereal* network protocol analyser [4] for real time capturing of the data in the network interface. The SGs, which are also Java mobile agents that are able to execute within the controller PC and have direct access via *telnet* protocol (this was available for our tests, and the risk could be minimised if the two are connected via e.g. a serial port and not over IP, however a more secure

connections should be realised in the future) to the attached router and its services (e.g. modifying routing/filtering tables etc). In Listing 3, we have an exact snip of the Java code within the mobile agent that executes and changes the router's filter tables in order to block the attack. The same router-specific instructions are embedded in form of a secure URL for the non agent-based active network systems, within the <*RouterAction*> as shown in Listing 1.

```
// login in the Hitachi GR2000 router via the wrapper interface
 if(!wrapper.openConfig()) {
      System.out.println("Accessing GR2000 Router configuration failed!");
      System.out.println("Current mode is " + wrapper.prompt);
      } else {
      // Change Router's filter rules to block the attack
      wrapper.writeCommand("filter yes");
      wrapper.writeCommand("filter-list 1 -drop " +
                " -protocol " +attackProtocolNumber+
                " -ip_source " +attacker+
                " -ip_destination "+victim+
                " -port_destination "+attackFlowDestinationPort);
      wrapper.writeCommand("filter-group DDoS 1");
      wrapper.writeCommand("filter-interface ether4 in "+
                " filter-group DDoS");
      System.out.println(" OK. The attacker is now blocked.");
return;
```

Listing 3 -- Changing router's flow filter

As also described in the attack scenario, the SGs dynamically reconfigure hop by hop the AN nodes in order to block all malicious traffic directed to AN node C. The preinstalled policy rules of the agent platform make the controlled execution of mobile code possible. When they reach the source of the attack or the boundaries of the network, they simply die. The sniffing of the network as well as the analysis of data in this scenario was done in real-time (any other option would be inappropriate for any IDS system). The message exchange by the mobile agents and the components is digitally signed. One could also use encryption schemes in order to make available to the public only portion of these messages (encrypting portions of an XML file) or nothing at all (encrypting the whole message). The exact combination is up to each network and its policies. We assume here that the messages are only digitally signed, which provides message integrity and authentication of origin, and therefore can be exchanged among different entities (e.g. agents) freely.

9. Network approach

The scenario above demonstrated a prototype that recognizes and handles simple attacks in the network. It was implemented as a proof of concept that the above technologies are able to interoperate in a desired way. This approach up to now handles events locally within the node and the neighbouring ones, but can be generalized in order to see the big picture. A living network with parts that communicate and exchange info is seen in Figure 5. The network takes a modular approach. Multiple MS, ADS and ARS components exist from different providers, which are able to interoperate in different levels. ADS and ARS that are in the same domain can exchange information, and can also report it to a Central Analysis Point (CAP) for a more thorough examination.

The central analysis point has the overview of what is happening in its domain, making therefore easier to recognize attacks that include multiple nodes in different parts of the network. Furthermore the CAPs form an overlay network that shares info in a centralized or distributed way (Figure 5). With this flow of information, attacking one part of the network results in recognition of the attack pattern and propagation to all other parts, which are immune afterwards. Furthermore external services can be used via a push/pull approach in order to acquire info and enhance proactiveness. In the later although no part of the network has been attacked, the new info pushed from the external service (e.g. CERT security advisories [18]) propagate within the network, are evaluated and possibly automated changes tale place that proactively protect the network from possible future attacks.

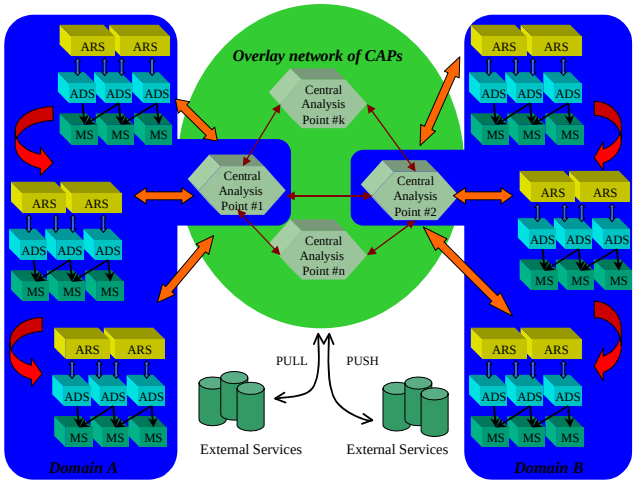

Figure 5 – A living network

10. Evaluation

There are two ways of defending against the DDoS attacks: a) proactive and b) reactive.

Proactive protection involves taking measures before an attack has occurred. This includes securing the nodes by patching possible security holes and being able to stop any attempts (e.g. network scanning, daemon installation etc) that may eventually lead in a DDoS attack.

Reactive protection involves taking measures to reduce the effect of an attack after it has occurred. That includes blocking even partially the attack at the most

outer point of the provider's domain and as close as possible to the originating hosts.

Of course the proactive protection is preferred, however it is difficult to be realized. Modern IDSs are moving somewhere between these two approaches. Early IDSs used a monolithic architecture where data was collected in each node and analysed by a central node. Of course the approach was obsolete as it failed to recognize attacks that included multiple nodes. Network based IDSs tackled this problem by monitoring the network behaviour. However even the most modern IDSs lack flexibility and do not scale good enough if they spawn heterogeneous infrastructures. Furthermore they have a limited response capability and do not provide open interfaces neither can exchange security info with third party IDS implementations. The latest is an application area for software agents where a lot can be achieved.

Agents have several characteristics that we require. Beyond being designed with intelligence and mobility in mind, they can also:

- spawn heterogeneous networks (the agents depend only on the execution environment),
- implement missing services on network nodes or even be used as wrappers for the existing ones
- encapsulate protocols and exchange messages in a standardized format [27].
- reduce network load by processing the data locally on the node and not transfer everything to a central network point where the analysis is done
- execute autonomously and adapt to a changing environment
- clone themselves for replacement or redundancy
- decompose and solve problems in a collaborative fashion and share knowledge

Applying agent to intrusion detection is not entirely new. The Autonomous Agents for Intrusion Detection (AAFID) [29], the Java Agents for Meta-Learning (JAM) [30], the Intrusion Detection Agent (IDA) [31] and the Mobile Agent Intrusion Detection System (MAIDS) [32] are some of the existing efforts that deploy agents in IDSs. However most of these approaches deploy static agents and not mobile ones. Again these projects should be seen as complementary components that can be integrated as parts in the effort to realize the community-aware network approach presented here. Furthermore research has to be done in the agent field (e.g. malicious agents should not fool the SGs in blocking legitimate traffic), in the IDS domain (e.g. correctly identifying a system as being attacked and not overloaded) as well as in standardization activities in order to use commonly understandable semantics.

Active networks provide the necessary programmability required by the underlying nodes in order to allow flexible network customisation. Therefore a combination of agent technology and active networks (as the one presented in this paper) is promising. The generic building blocks of the architecture in Figure 3 can be fully implemented by agents. However we do not think it is effective to make all of them mobile. A hybrid approach in which the intelligent parts are static and the mobile ones are small pieces of code that move around might more appropriate and more realistic.

Mobile agent enabled active network infrastructures alone do not directly improve any techniques for detection of DDoS attacks. However they can reshape the existing ones, add a modular more open approach to the whole existing implementations therefore improving efficiency, effectiveness and re-usage. Agents can also wrap existing IDSs and enhance them with new capabilities, making them live integral parts of a living system. Community-aware security is seen as the next step of evolution in not only dealing with DDoS attacks, but generally with security breakouts, and with which we will be able to realize more efficient techniques in protecting our networks.

11. Conclusions

We have presented here our ideas and thoughts on future network security. We support the idea that networks should adopt social characteristics as seen on biological networks and therefore built up communities within the network that will be able to cooperate and exchange info in a coherent way. By having one part of the network attacked, and via the information flows, the other parts can be proactively protected while the initial attack is pushed step-by-step to the borders of the network and eliminated. One possible way to implement this concept is to use mobile agents and active networks as discussed through the paper. Both of them are relative new technologies developed in the last decade or so. We do not foresee a commercial future at the moment as these approaches have to mature and their complex parts and logic are still under heavy research. However we strongly support the concept of community aware networks as an approach that will lead to more secure and self-sustainable network infrastructures in the years to come.

12. References

[1] David Dittrich, Tools for Distributed Denial of Service attacks, http://staff.washington.edu/dittrich/misc/ddos

[2] The BANG project, http://www.fokus.fraunhofer.de/research/cc/glone/projects/bang/

[3] The Grasshopper Agent Platform, http://www.grasshopper.de/

[4] The Ethereal Network Protocol Analyzer, http://www.ethereal.com/

[5] Jonathan M. Smith, Scott M. Nettles, “Active Networking: One View of the Past, Present and Future”, IEEE Transactions on Systems, Man, and Cybernetics (T-SMC), Volume 34, Number 1, pp. 4-18, February 2004 (ISSN: 1094-6977).
[6] Mobile Agent Platforms, http://www.agentlink.org/resources/agent-software.php
[7] Hitachi GR2000 Gigabit Routers, http://www.internetworking.hitachi.com/products/products_GR.html
[8] Biswas, J., Huard, J.-F., Lazar, A.A., Lim, K.-S., Mahjoub, S., Pau, L.-F., Suzuki, M., Torstensson, S., Wang, W. and Weinstein, S., "The IEEE P1520 Standards Initiative for Programmable Network Interfaces", IEEE Communications Magazine, Special Issue on Programmable Networks, Vol. 36, pp. 64-72, IEEE, October 1998.
[9] Node OS Interface Specification. AN Node OS Working Group, Larry Peterson, ed., January 24, 2000.
[10] Architectural Framework for Active Networks, Draft version 1.0, K.L. Calvert, ed., July 27, 1999.
[11] D. J. Wetherall, J. Guttag and D. L. Tennenhouse, ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols, IEEE OPENARCH'98, San Francisco CA, Apr. 1998.
[12] D. Scott Alexander, ALIEN: A Generalized Computing Model of Active Networks, Ph.D. Thesis, University of Pennsylvania, December 1998.
[13] MASIF - Mobile Agent System Interoperability Facility, http://www.omg.org/docs/orbos/98-03-09.pdf
[14] Stamatis Karnouskos, “Security Implications of Implementing Active Network Infrastructures using Agent Technology”, Special Issue on Active Networks and Services, Computer Networks Journal, Volume 36, Issue 1, pp 87-100, June 2001 (ISSN 1389-1286).
[15] FIPA Web Site: http://www.fipa.org/
[16] Mobile Agents Technology: http://www.cetus-links.org/oo_mobile_agents.html
[17] Future Active IP Network (FAIN) Project, http://www.ist-fain.org
[18] CERT security advisories, http://www.cert.org/advisories/
[19] S. Karnouskos, H. Guo and T. Becker, Trade-off or Invention: Experimental Integration of Active Networking and Programmable Networks, Special Issue on Programmable Switches and Routers, IEEE Journal of Communications and Networks, Volume 3, Number 1, pp 19-27, March 2001 (ISSN 1229-2370)
[20] Stamatis Karnouskos, “Realization of a Secure Active and Programmable Network Infrastructure via Mobile Agent Technology”, Special Issue on Computational Intelligence in Telecommunications Networks, Computer Communications Journal, Volume 25, Issue 16, pp. 1465-1476, October 2002 (ISSN: 0140-3664).
[21] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, Practical Network support for IP Traceback”, ACM SIGCOMM Conference 2000, Stockholm, Sweden
[22] K. Park and H. Lee, On the Effectiveness of Probabilistic Packet Marking for IP Traceback under Denial of Service Attack, Department of Computer Sciences, Purdue University, IEEE INFOCOM 2001.
[23] M. Kang, I. Moskowitz, D. Lee, A Network Pump, IEEE Transactions on Software Engineering, pp. 329-338, Vol. 22, No. 5, May 1996.
[24] Dublin Core Metadata Initiative, http://dublincore.org/
[25] XML Encryption Syntax and Processing, http://www.w3.org/TR/xmlenc-core/
[26] XML-Signature Syntax and Processing, http://www.w3.org/TR/xmldsig-core/
[27] FIPA Agent Communication Language (ACL), http://www.fipa.org/repository/aclspecs.html
[28] IETF Intrusion Detection Exchange Format Working Group, http://www.ietf.org/html.charters/idwg-charter.html
[29] Autonomous Agents for Intrusion Detection Group (AAFID), http://www.cerias.purdue.edu/homes/aafid/
[30] Java Agents for Meta-learning (JAM), http://www.cs.columbia.edu/~sal/JAM/PROJECT/
[31] Intrusion Detection Agent System (IDA), http://www.ipa.go.jp/STC/IDA/
[32] Mobile Agent Intrusion Detection System (MAIDS), http://latte.cs.iastate.edu/Research/Intrusion/
[33] Andrew P. Moore, “Network Pump (NP) security target”, Naval Research Laboratory, 29 May 2000. http://chacs.nrl.navy.mil/publications/CHACS/2000/2000moore-NPST.pdf
[34] J. P. Anderson, "Computer security threat monitoring and surveillance," Technical report, James P. Anderson Co., Fort Washington, PA, April 1980
[35] Dorothy E. Denning. An intrusion-detection model. IEEE Transactions on Software Engineering, 13(2):222-232, February 1987.
[36] S. Forrest, S. Hofmeyr, and A. Somayaji, “Computer Immunology” Communications of the ACM Vol. 40, No. 10, pp. 88-96 (1997).