Dealing with Denial-of-Service Attacks in Agent-enabled Active and Programmable Infrastructures

Stamatis Karnouskos
German National Research Center for Information Technology
Research Institute for Open Communication Systems (GMD-FOKUS)
Kaiserin-Augusta-Allee 31, D-10589 Berlin, Germany
karnouskos@fokus.gmd.de

Abstract

Denial of Service (DoS) attacks is a well-known problem with victims even among prestigious commercial sites. Such attacks in traditional networking are difficult to recognize and to handle. An active infrastructure that can dynamically respond to event-triggered requests can deal better with recognition and handling of DoS attacks. We present here a DoS attack response system architecture and we demonstrate via an application scenario its dynamicity and flexibility in dealing with this kind of attacks. The approach is based on agent-enabled active programmable infrastructures 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 Systems, Distributed Denial of Service Attacks, Agent Technology, Active Networks.*

1. Introduction

Distributed Denial of Service (DDoS) attacks launched against prestigious commercial sites such as Yahoo!, Amazon, eBay Inc., Buy.com and others the last months, have attracted a lot of publicity. The problem is known and difficult a) to recognize early enough and b) to handle.

Today's Intrusion Detection Systems (IDS) are 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 can not deal with the challenges set by current dynamic infrastructures. The network must have the ability to dynamically change its behavior 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.

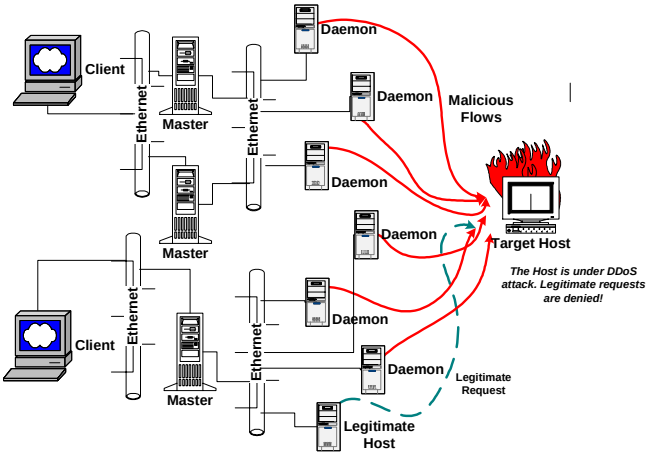

Figure 1. Distributed denial of service attack

In order to achieve that we need networks that can sense the environment and react to its changes.

We believe that active networks constitute the right step to this direction, as applications can foster task specific network customization. That, in combination with agent technology can lead to a better way of dealing with DoS scenarios as we will demonstrate in this paper.

1.1. 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 such as Smurf [1] and Trinoo [2] but also modern ones like Tribe Flood Network (TFN) [3] TFN2K [4] Stacheldraht [5] Mstream [6] and Shaft [7] 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

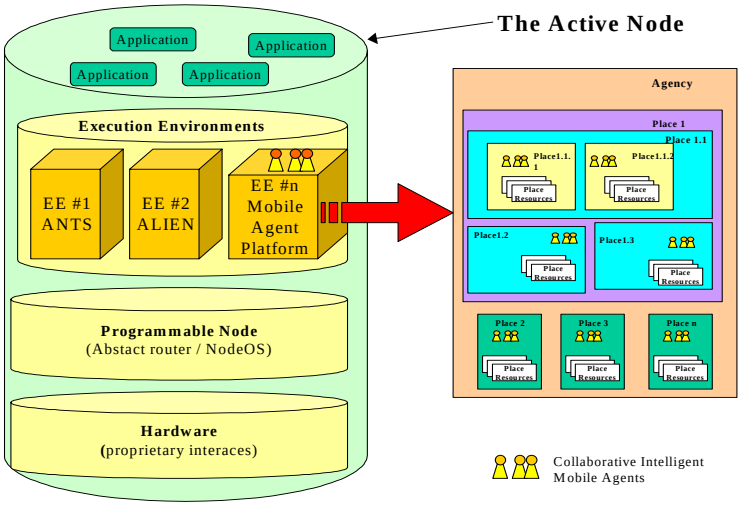

Figure 2. The Agent-based active network node architecture

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 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 (scanning for vulnerabilities and installing the malicious code) could be as little as 7 seconds per host.

1.2. Agent Technology

Agents [22] 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. One key characteristic of agents is mobility, which allows them to 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 its benefits such as dynamic, on demand provision and distribution of services, reduction of network traffic and dependencies, fault tolerance etc. The number [12] of mobile agent platforms coming from the commercial sector, as well as the academia is increasing day by day.

1.3. Active Networks

Active Networks (AN) [11] 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 (APIs) 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). 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 its infancy.

2. The Agent-enabled Active 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 optimized components and algorithms implemented in their hardware form for performance reasons. Software approaches in this level will only slow node's function down e.g. the forwarding function.

Programmable part: This part integrates the manufacturer proprietary interfaces of the fixed part and exports an open standardized interface. The APIs are standardized by the IEEE P1520 project [14]. At this level the node can be programmed but only via a parameter specific approach. The programming can be done e.g. via an RPC method and it has the advantage that the node

always falls into deterministic states. This open interface represents the abstraction of the hardware available resources, ranging from computational resources (CPU, memory etc) to packet forwarding resources (bandwidth, buffer, etc). The Node Operating System (NodeOS) provides the basic functionality from which the execution environments (EEs) built the abstractions presented to the active applications. The architecture of the NodeOS and its functionality is outlined in detail by the AN Node OS Working Group [15]. Let us mention that the NodeOS could also be a distributed processing environment (DPE) that makes the necessary abstractions.

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. Applications that need task specific control of node's states, can implement in the form of active code the specific algorithms they need from the scratch, or by combining the services that are available in the node in a Lego-like way. The executed active code, uses also the interfaces that are provided by the programmable part (generic interfaces) in order to access functionality implemented in the hardware part. As also noted [16] 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 [17], ALIEN [18], 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.

As shown in Figure 2, one of the EEs is the agent execution environment. This is the agency as described within the MASIF [19] 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. A Place is associated with a location which consists of a Place name and the address of the agent system within which the Place resides. Places can contain other Places. All Places follow the parent-child paradigm of Unix processes in the way that each child is assigned/makes use of its parent's resources. Also its policy is an extension/customization of its parent's policy.

The existence of different EEs and sub-EEs for agents (which are the Places within the agent architecture) that have the same owner/characteristics serves the need to avoid unwanted interactions. Isolation done by Places is similar to the sandbox idea that exists in Java. Since in each Place agents with common characteristics (e.g. of the same owner) are gathered the possibility of attacking each other is lower as usual. Of course further security countermeasures [20] have to be taken in order to provide a secure working system.

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 agent can either be generated at a Place locally (e.g. out of a pool of ready-programmed objects) or it can just carry on with an execution it suspended in another node.

3. System Architecture

The architecture of our DoS (the same apply for DDoS scenarios) response system is depicted below. It is composed of the following parts:

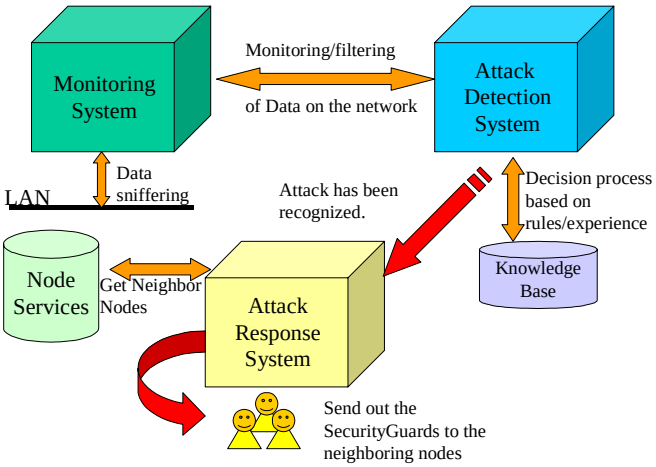

Figure 3. System architecture

Monitoring System (MS): This part is responsible for analyzing and capturing all data that passes via the network interface. The data that is captured/stored by the monitoring system 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.

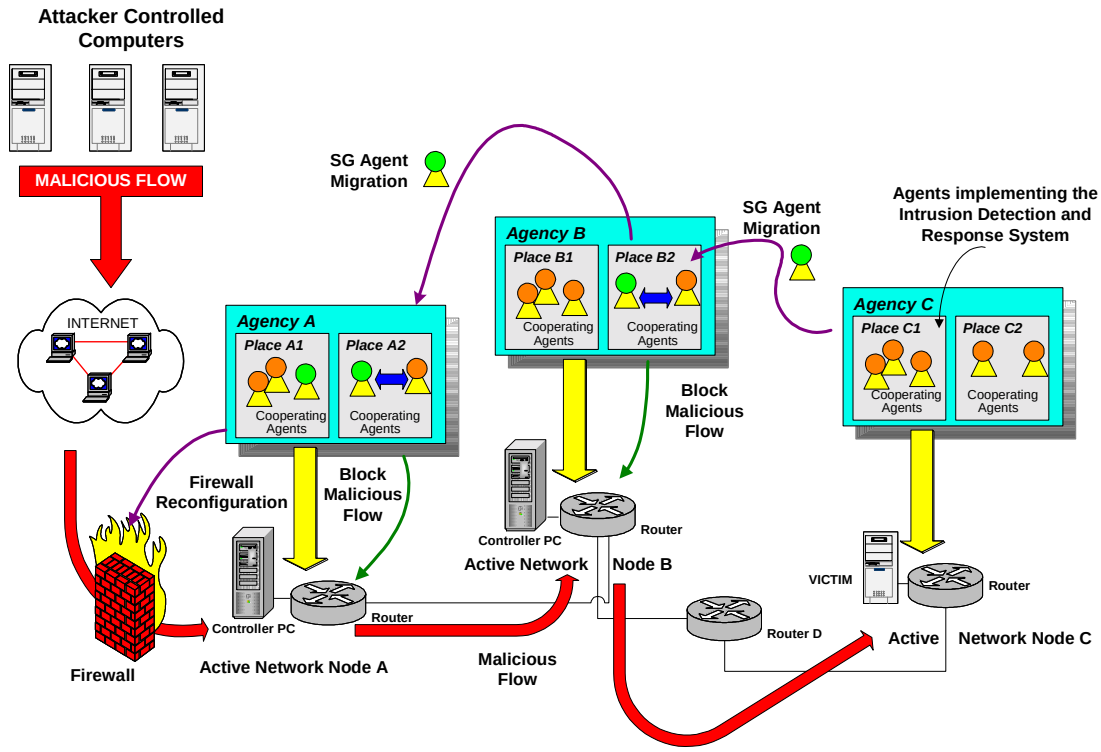

Figure 4. DDoS attack and response Scenario

Attack Detection System (ADS): This component is responsible for identifying the attack against the node. It has its own decision process based on internal rules, heuristics and expertise stored in its knowledge base. The trigger event identifying a DoS 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 DoS 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.

In a distributed scenario the ADSs from all nodes can co-operate and push/pull information from each other 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 neighbor 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 to transport themselves to the neighboring machines. Please note that on each node the agents are able to sense the environment e.g. on the fly discover the neighboring 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 DoS attack is over.

4. 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 the 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, detects the pattern of the attack 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 neighboring nodes or the nodes that are one hop away. This could be accomplished simply by invoking the *traceroute* utility of Unix systems (to 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 neighboring 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 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.

5. Implementation

A first prototype of this approach was implemented and demonstrated within the scope of BANG project [8]. Further discussions on the concepts of this approach and enhancements in the implementation will be possibly done within the FAIN project [23]. Our testbed is the same as the one described with the Figure 4. The active nodes consisted of Hitachi GR2000 gigabit routers [13] that were managed via a Controller PC. The Grasshopper agent system [9] was selected as the mobile agent platform to be embedded in the control PC. Both ADS and ARS modules where implemented as Java mobile agents. Although ARS and ADS did not need to be mobile, the motivation behind that was that in the future it might be a good idea to allow the whole system to roam the network and clone itself for survivability and load balancing reasons. The MS module was consisted of a modified version of the *ethereal* network protocol analyzer [10] for real time capturing of the data in the network interface. The SGs, which were also Java mobile agents, were able to execute within the control PC and had direct access via *telnet* protocol to the attached router and its services (e.g. modifying routing/filtering tables etc). As also described in the attack scenario in section 4, the SGs dynamically reconfigured hop by hop the AN nodes in order to block all malicious traffic directed to AN node C. When they reached the source of the attack or the boundaries of the network, they simply died. The sniffering 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).

6. Evaluation and Conclusions

There are two ways of defending against the DoS/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 analyzed 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 behavior. 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 [21].
- 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
- collaborate and share knowledge

Active networks provide the necessary programmability required by the underlying nodes in order to allow flexible network customization. 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.

It is clear that mobile agent enabled active network infrastructures do not directly improve any techniques for detection of DDoS attacks. However they can reshape the existing ones and 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 their capabilities as discussed before. In the future we will try to address further the IDSs requirements and experiment with hybrid approaches where an ADS can subscribe to network-wide security notifications (e.g CERT [24]) and its database is pulled/pushed with real-time info from the whole network. Furthermore the ARS needs to have better control over the SGs once they have left the node. Collaboration of all components in a heterogeneous network is not expected to be a trivial task and needs to be further investigated.

References

[1] Smurf DDoS tool, http://www.powertech.no/smurf/

[2] Trinoo DDoS tool, http://staff.washington.edu/dittrich/misc/trinoo.analysis.txt

[3] Tribe Flood Network (TFN) DDoS tool, http://staff.washington.edu/dittrich/misc/tfn.analysis

[4] Tribe Flood Network 2000 (TFN2K) DDoS tool, http://packetstorm.securify.com/distributed/TFN2k_Analysis-1.3.txt

[5] Stacheldraht DDoS tool, http://staff.washington.edu/dittrich/misc/stacheldraht.analysis

[6] Mstream DDoS tool, http://staff.washington.edu/dittrich/misc/mstream.analysis.txt

[7] Shaft DDoS tool, http://netsec.gsfc.nasa.gov/~spock/shaft_analysis.txt

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

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

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

[11] Active Networks at DARPA, http://www.darpa.mil/ito/research/anets/

[12] Mobile Agent Platforms http://www.informatik.uni-stuttgart.de/ipvr/vs/projekte/mole/mal/mal.html

[13] Hitachi GR2000 Gigabit Routers, http://www.internetworking.hitachi.com/products/products_GR.html

[14] IEEE P1520 Project , http://www.ieee-pin.org/

[15] Node OS Interface Specification. AN Node OS Working Group, Larry Peterson, ed., January 24, 2000.

[16] Architectural Framework for Active Networks, Draft version 1.0, K.L. Calvert, ed., July 27, 1999.

[17] 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.

[18] D. Scott Alexander, ALIEN: A Generalized Computing Model of Active Networks, Ph.D. Thesis, University of Pennsylvania, December 1998.

[19] MASIF - Mobile Agent System Interoperability Facility, http://www.omg.org/docs/orbos/98-03-09.pdf

[20] 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).

[21] FIPA Web Site: http://www.fipa.org/

[22] Mobile Agents Technology: http://www.cetus-links.org/oo_mobile_agents.html

[23] FAIN Project, http://www.ist-fain.org

[24] CERT security advisories, http://www.cert.org/advisories/