Baseline—a passive approach to tolerate and detect DoS/DDoS attacks

JIN Shu[1], LIU Feng-yu[1], XU Man-wu[2]
(1. Department of Computer Science, Nanjing University of Sci. & Tech., Nanjing 210014, China;
2. Department of Computer Science, Nanjing University, Nanjing 210093, China)

Abstract: By employing a novel communication service surveillance algorithm called "Shepherd", a DDoS (Distributed Denial-of-Service) detection architecture named Baseline, which is considered a passive approach, is presented. It achieves high adaptability through delegating the QoS (Quality of Service)degradation judgement to the individual communication processes. By adding plug-able modules to the actuator of the daemon, Baseline can be easily integrated with IDSs (Intrusion Detection Systems). While compared with previous work, neither traffic analysis or packets content filtering nor any modification to the existing router systems is required, which is very feasible. Moreover, Baseline may achieve zero false positive to some extent.

Key words: Baseline service; availability surveillance; DoS/DDoS attacks detection
CLC number: TP 393.08 **Document code**: A **Article ID**: 1006-6047(2005)06-0007-07

0 Introduction

Although a perfect solution to defend against the DDoS (Distributed Denial-of-Service) attacks, of which several typical kinds such as Smurf, SYN-Flooding and TFN2k have been identified and investigated[1-3], can never be achieved in the current Internet infrastructure according to the nature and the underlying routing mechanisms of the Internet, many solutions have been suggested to address the problem. For example: Ingress Filtering[4]/Egress Filtering[5] can sieve out the spoofed packets even before they are injected to the Internet; IP traceback[6] and ICMP traceback[7] are introduced to find out the sources of the DDoS attacks; ACC (Aggregate-based Congestion Control)[8] is introduced to detect and thwart DDoS attack streams through the mechanism of Pushback [9], which calls for the cooperation of the routers in the path of packets forwarding from the source to the destination; Overlay networks[10] are used by the service providers to hide their Internet servers, and still some approaches try to detect DDoS attacks simply through the measurement of some specific variables such as the network throughput or the packet loss rate of the supervised host. These solutions, if applied to the hosts and routers in the Internet, will greatly reduce the risk of been attacked by the DDoS attacks.

Baseline service—the approach present in this paper tries to address the problem of DDoS attacks from a different perspective, which is often neglected by other approaches that: what effect the DDoS attack may cause to the network communication processes on the hosts? As the problem of DDoS attacks cannot be easily and totally resolved, if it doesn't hider the data communication of the running processes or only degrade the service to an acceptable extent, why not just tolerate and monitor on it instead of being over-alert? The philosophy that sits in the heart of our solution is: "If it ain't broken, don't fix it", Baseline service thus takes a passive attitude toward the possible DDoS attack traffic. The only thing that is cared by the Baseline daemon deployed in the host is the service availability status of the currently running communication processes, are they properly running with enough bandwidth to service their clients or peers? Or have already been DDoSed to a halt? Baseline service takes care of it as will be described in section 1.

After giving a thorough description of the Baseline service architecture in section 1, the Shepherd algorithm of DDoS detection, which is our core contribution, is described in section 2. An in

收稿日期: 2005-03-16

-depth analysis of the whole approach will also be given in this section as well. Section 3 discusses in detail the implementation issues of Baseline service.

1 Baseline service

1.1 Overview

Compared with approaches such as [8] and [4], which are proactive in the defending of the DDoS attacks, and some other solutions like [10] and [11], which is considered to be reactive, Baseline takes a somewhat passive attitude toward the DDoS attacks. With the creed that "if it ain't broken, don't fix it", Baseline service runs silently in the background and monitors on the running communication processes, if the availability of any such process is degraded to a certain degree, which is defined in the configuration of the process, a possible DDoS attack may be identified and thus further actions may be taken due to the decision made by the Baseline actuator.

Running as a background daemon, Baseline service architecture contains five building components as shown in Fig.1. They are namely Configuration, Register, Table, Shepherd and Actuator. Any process that engaged in data communication (TCP/IP based here) within the Internet may set their respective requirements of service quality in the Baseline Configuration. When started, the Shepherd of the Baseline daemon monitors all the communication operations performed by such registered processes continuously and if the required service quality cannot be ensured, the Shepherd will inform Actuator the situation and thus further actions may be taken due to a decision making procedure of the Actuator.

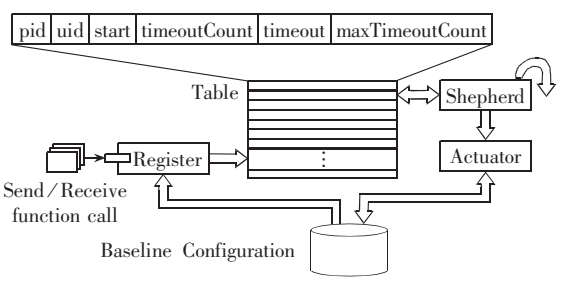

Fig.1 Baseline service architecture

1.2 Configuration

We base our approach on the point that whether denied services or not, the processes themselves know best. Thus the communication processes may provide their specific definitions of denial of service by set them in the Baseline Configuration. This strategy makes Baseline service different from many other DDoS Detection methods, which make decisions on whether there are DDoS attacks going on without even query the status of the running network services. When a possible DDoS attack is identified through a noticeable degradation (defined in the Configuration) in the service quality of a supervised communication process, Baseline service will make decisions for the proper reactions to be taken, which is supported by the Baseline Configuration. Variables that associated with the quality of service for instance timeout, max timeout times, minimum bandwidth required, maximum bandwidth required and many others may be specified in the Configuration and hence be used to define the normal service status and support the decision making process of the Baseline Actuator.

With no specific service quality requirements provided in the Configuration, a communication process may be only monitored on its timeout and timeout time's count, which is considered a universally applicable mechanism and can be efficiently implemented. In order to be applied by the Baseline service, more complex strategies need to be specified by the owners of the processes in the Baseline Configuration.

As shown in Fig.1, current version of the Baseline service architecture uses timeout and timeout counts to measure the availability of certain communication processes. If flooded by a DDoS attack, the processes monitored by the Baseline daemon may be deprived of their legitimate bandwidth occupation and their send/receive calls encounter timeouts; the Actuator will analyze the situation provided by the Shepherd then and take further actions.

1.3 Table

Baseline Table is the specific storage allocated by Baseline daemon to record the brief QoS(Quality of Service) status of every send/receive call invocated by the communication processes that registered with Baseline service, which resides in the memory and may be dynamically appended by the Register and modified by the Shepherd. When a send/receive operation is invocated, the Register generates a new entry with the service quality requirements specified in the Configuration. The entry contains such fields as described in Fig.1 and they are namely the id of the communication process (pid), id of the user that run the process (uid), beginning of method invocation (start), timeouts encountered (timeoutCount), timeout duration (timeout) and max times of timeout (maxTimeoutCount). The Shepherd modifies all the entries in the Table periodically and will delete any entry in the following two situations: 1) when the send/receive call associated with it successfully fi

nished; 2) more timeouts than can be tolerated encountered and the actuator is activated. The Table must be restricted to hold only a limited number of entries in order to minimize memory cost while still retain the efficiency, thus comes the problem of the Table size choosing, which is important in the Baseline daemon implementation and will be discussed in section 2.

1.4 Register

In the architecture of Baseline service, Register acts as the bridge between the user application and the Baseline service. The communication application that needs their availability monitored must register every function call of data sending / receiving they invocated to the Baseline Table when they are running. Similarly, before a data sending / receiving function call returns successfully, a process of un-registration must be performed by the communication process that invocate the function call in order to remove the entry related to the very function call from the Table. If the specific requirements of service qualities are not supplied with these registrations, the Register will perform lookups in the Configuration to retrieve the QoS requirements of the very processes. The multiple QoS requirements registrations of communication operations in the life time of a certain process can be simplified to a single one that when the process start communication in the network, its specific QoS requirements is cached and will be automatically filled to the subsequently generated entries by the Register.

1.5 Shepherd

Shepherd is the always-running part of the Baseline service. After started the Shepherd will keep running and monitor the Baseline Table periodically by updating the entries in the Table iteratively. If signs of denial of service are identified from an entry, for example the maximum timeouts is encountered, Shepherd will notify Actuator the situation and remove the very entry. In order to achieve high efficiency while introducing only trivial overhead to the system, no further decisions are made by the Shepherd, which only plays the role of a Shepherd that monitors his sheep—the function calls invoked by the communication processes that registered in the Table. In the current version of Baseline service architecture, Shepherd loops to check every entry in the Table to find timeouts, if a timeout is found Shepherd will increment the associated timeout counter and compare it with the threshold set by the processes to identify a possible severe QoS degradation and then ask the Actuator for the corresponding reactions to be taken. Detail of the algorithm applied by Shepherd and the implementation issues are given in the following sections.

1.6 Actuator

Baseline service daemon depends on the Actuator for decision making and taking further actions to the identified DDoS attacks. Any data sending / receiving function calls found to be unresponsive (for example reaching their respective maximum timeouts) by the Shepherd will be informed to the Actuator. Decisions on whether there is a DDoS attack or not will be made according to the corresponding strategies specified in the Baseline Configuration. The status of the running communication processes and the local network connections are also important to the decision making procedure for that if not conform to the registration protocol, a communication process may leave an entry in the Baseline Table even after a successful execution and hence makes the operation seemed blocked by too many timeouts. Such accident as an imprudence unplugging of the network cable may also deny the running communication processes their accesses to the network infrastructure, which should not be classified as DDoS attack either. Based on all the factors aforementioned, the Actuator decides whether the reported situation is caused by a DDoS attack. As the decision making engine is still under research, a possible implementation will be provided in section 2.

If a noticeable degradation in QoS is identified to be caused by a DoS/ DDoS attack, the Baseline Actuator may take some reactions such as to raise an alarm to the system administrator, calling the beeper or send SMS messages to the security administrator; notify the DDoS attack to the interoperated IDS (Intrusion Detection System) or just log the incident for the further analysis. Provided many choices of actions to be taken when a DDoS attack is detected, Actuator may provide a good scalability.

1.7 Overview

With the cooperation of the five components, namely Register, Table, Configuration, Shepherd and Actuator, Baseline service provide the system user a DoS/DDoS attack detection mechanism through monitoring the service quality properties of their communication processes and services. Different from those approaches which detect DDoS attacks based on their own judgments, the criteria adopted by

Baseline service to identify a DDoS attack are provided by the processes that really use or provide some services in the Internet infrastructure, which makes detection more accurate and results a less false positive rate. It seems that a more false negative rate may be resulted due to the passive nature of Baseline service, however we think it differently that if no communication processes are affected or their service qualities are just degraded to an acceptable extent, it's just ok to tolerate and monitor on the situation. Like many other approaches, the unique application of the Baseline service cannot solve the entire problem, a closely cooperation with more active tools is considered a better practice.

2 Algorithm and analysis

2.1 Shepherd algorithm

The algorithm that performed by the Shepherd of the Baseline service is presented in Algorithm 1, which sits in the center of the Baseline service architecture. The Shepherd loops in the background and checks periodically the entries of the registered communication function calls to see if there are some timeouts occurred through comparing and modifying the entries. If more timeouts than the user specified threshold for a denial-of-service is witnessed, the Actuator will be notified the situation, with the corresponding entry in Baseline Table passed as the arguments, for the further decision-making procedures and actions to be taken.

Algorithm 1: Shepherd Algorithm

```
Loop
  Set current time to t;
  For each entry {e_i} in the Baseline Table
    If((t - e_i.start) > e_i.timeout)
      If((e_i.timeoutCount+1) < e_i.maxTimeoutCount)
        e_i.timeoutCount ++;
        Let e_i.start = t;
      Else
        Actuate(e_i);
      End If
    End If
  End for
  Sleep(SLEEP_INTERVAL);
End loop
```

e_i in the algorithm refers to an entry in the Baseline Table, which contains the fields as shown in Fig.1. As any entry in the Baseline Table will be removed before the corresponding communication function call finishes its operation, a timeout is recognized if the entry survived a longer time than is expected by the user, which may be denoted by the expression $((t - e_i.\text{start}) > e_i.\text{timeout})$. The counter will be incremented (e_i.timeoutCount++) accordingly in the situation and if its value exceeds the threshold specified in the Baseline Configuration, which is loaded into the field maxTimeoutCount of the entry through the process of registration, a denial-of-service is thus identified. In the infinite loop as shown in Algorithm 1, it is important to keep the CPU from being overwhelmed, so we use a sleep operation at the end of the loop body to yield the control of execution for a given time span (SLEEP_INTERVAL), which is an empirical value selected according to the supporting operation system and the specific implementation of the Baseline service.

For an always-running daemon like the Baseline service, to run efficiently while introducing only a low overhead to the system is of great importance. The computational complexity of the algorithms typically applied in some other DoS / DDoS attack detection approaches such as those based on pattern matching is linear to the cost needed to process a single packet-denoted as P_{pkt}, which is determined by the average packet length together with the size of the rule set, thus entails many comparison operations. Although Shepherd algorithm is also considered to be executed in a time linear to the average size of the Baseline Table, which is intuitively the same with the number of packets passing by, the operations that performed on a Table entry (only several additions and comparisons) cost greatly less than P_{pkt} — that needed in a pat-tern-matching one. Let alone other algorithms that are directed by some more complex heuristics such as neural networks, genetic algorithms and others, all these approaches may cost even more computation resources.

2.2 Simulation and analysis

Different from other DoS / DDoS detection approaches, which focus on the correctness and effectiveness of their algorithms, the most important issue in designing and implementing of the Baseline service is its efficiency. As described in the section 1, if the Baseline service is efficiently implemented and deployed, any DoS / DDoS attacks that disturb the normal communication of the legitimate processes and services will be identified through the QoS degradation defined by them. As communication processes register every data sending/receiving function calls to the Baseline Table and remove them after they are successfully finished, the throughput of real network communication can be simulated

through just registering and removing entries in Baseline Table. A simulation architecture is thus suggested as shown in Fig.2, in which figure a helper thread named generator generate Table entries and register them into a buffer, which represents the Baseline Table, and a thread named pseudo server to remove selected entries according to some given rule from the buffer. With the both helper threads running continuously, the real communication environment can be simulated.

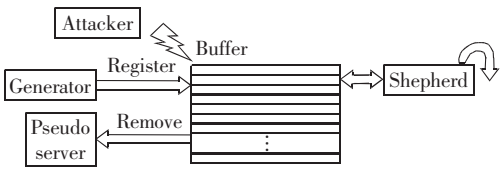

Fig.2 Shepherd algorithm simulation environment

In a real life network communication environment, the specific rate of data sending / receiving function calls may be limited due to their cost and the delay of the underlying network, which depend on the computation power provided by the computer system closely, for example the limit in our test bed system is around 46 k calls per second, which is achieved by a test process that grabs the CPU and sends 32-byte UDP packets incessantly in our test-bed machine — a normal Pentium-4 PC (IBM NetVista 3305HC3) running at 2.4 GHz with 256 M RAM. In the simulation of the Shepherd algorithm, as a size of 46 k multiply with the maximum value for a typical timeout may be just enough (for example: when the timeout is set to 5 s and max timeouts set to 3, 46 k*5*3<700 k), we set the buffer big enough to hold any number of entries through applying dynamic reallocation.

Before introducing the helper threads of the generator and the pseudo server, a set of simulations are performed to evaluate the performance of the Shepherd algorithm when different Table sizes are applied. Trying to exclude the possible interferences, the coefficient of SLEEP_INTERVAL in the simulations is set to a fixed value of 500 ms. As shown in Tab.1, mainly for the operations of memory allocation, the Shepherd algorithm utilizes only 0.025 %~0.2 % of CPU time when the Table is set to contain less than 250 k entries. A steep rise is witnessed only after the size of the Table is set beyond 300 k, in which situation the iterations of the Shepherd algorithm, not the operations of memory allocation, take most of the CPU time assigned to the process and the CPU utilization rate rises almost linearly to the size of the Baseline Table. Table sizes that exceed 5 M are not tested for the fact that with too many memory allocation operations incurred in those situations, the whole system will be overwhelmed by the frequent swapping performed by the operation system. Considering all the facts discussed, a Table with 128 k entries may provide enough room for the communication processes to register their data sending / receiving operations while inviting only trivial computational overhead.

Tab.1 Performance under different Table sizes

Table Size	CPU Usage / %	Table Size	CPU Usage / %	Table Size	CPU Usage / %
1 k	0.024	30 k	0.063	500 k	3.176
2 k	0.024	40 k	0.050	1 M	7.246
3 k	0.021	50 k	0.100	2 M	15.330
4 k	0.024	100 k	0.130	3 M	21.720
5 k	0.024	200 k	0.255	4 M	26.870
10 k	0.050	300 k	0.713	5 M	27.420
20 k	0.050	400 k	2.911		

To evaluate the performance of the Shepherd algorithm from another perspective, different sleep times are tested with a fixed Table size of 32 k. As seen from Tab.2, the simulation process occupies approximately 83 % CPU time when the SLEEP_INTERVAL is set to 0 no sleep at all. When a sleep is added to the end of a round of the algorithm, which yield the CPU for a period of time designated by the SLEEP_INTERVAL, a steep fall of the CPU utilization percentage is resulted, which remains stable (0.05 % of CPU utilization) regardless to the prolongation of the sleep time.

Tab.2 Performance under different sleep intervals

Sleep Interval / ms	CPU Usage / %	Sleep Interval / ms	CPU Usage / %
0	82.246	25	0.060
1	0.200	100	0.051
2	0.060	250	0.051
5	0.054	500	0.051
10	0.076	1 000	0.051

Another generator thread—attacker, as described in Fig.2, is introduced to simulate the attack traffic in DoS/DDoS attacks through generating Table entries at a much higher rate than that may be possible in the real communication sessions. Two types of entry generation strategies are employed in the attacker thread; the first is a linear strategy, which generate Table entries at a fixed rate and the second a Poisson one that generate k entries in any time of Δt with a probability of $\frac{(\lambda\Delta t)^k}{k!}e^{-\lambda\Delta t}$.

In Tab.3, when the entry generation rate of the linear attacker exceeds 5 k entries per second the CPU utilization ratio rises along with it almost

linearly. Similar results can be seen on the Poisson attacker thread when the λ falls into the region of 1k~30k, more fluctuations are witnessed when λ is set higher than 30k due to the probabilistic nature of the Poisson process. To make the conclusion, while occupying more and more CPU time as the rate of the Table entry generation grows up, the Shepherd algorithm remains efficient.

Tab.3 Performance of Shepherd algorithm under simulated DoS/DDoS attacks

Attack Rate /(kpkt·s^{-1})	CPU Usage/%		Attack Rate /(kpkt·s^{-1})	CPU Usage/%	
	Linear Attacker	Poisson Attacker		Linear Attacker	Poisson Attacker
1	0.050	0.079	25	5.100	5.300
2	0.050	0.077	30	6.300	6.400
3	0.050	0.052	35	7.525	8.251
4	0.127	0.205	40	8.755	8.671
5	0.204	0.184	45	9.946	10.760
7.5	0.947	1.177	50	12.396	10.532
10	1.510	1.480	55	13.333	12.056
15	2.791	2.791	60	14.170	13.650
20	3.958	4.113			

3 Implementation and discussion

The test implementations of the Baseline service support the results drawn from the analysis given in the previous chapter. Two different versions of the Baseline service are implemented to test the usability of the Baseline service, a Java one and a C++/Windows one. The Java implementation of the Baseline service is running as a daemon, any process that needs socket communication may send its request through the JNDI interface provided by the Baseline daemon. When a request is received, the daemon returns a socket object wrapped with the information needed by the Baseline service and register the very process in the Baseline Table and the communication operations invocated by the process will be supervised by the Baseline daemon afterwards. The C++/Windows implementation comes closer to the operation system as a hook is set between the users' socket function calls and the real Winsock function invocations, which performs the additional operations required by the Shepherd algorithm. As compared with the Java implementation, no modifications in the source codes of the communication applications are needed to make in the C++/Windows version and all the legacy applications that utilize the socket API may use with this version of Baseline service, which provide an easier and smoother way to deploy the Baseline service.

As the algorithm introduced in the chapter 2 judges the degradation of service quality and availability solely through the counting of the timeouts (as against the given values), the Baseline service architecture is essentially universal. Different versions of Shepherd algorithms the related service daemon may be implemented with different physical values supplied to be measured, which make the Baseline service architecture highly adaptive and thus can be deployed in any network communication environment.

4 Future work

A priority queue mechanism is scheduled to be implemented in the future version of the Baseline service, for the fact that with a priority queue introduced, the communication process that has a higher priority of availability may be ensured supervision by the Shepherd with a higher certainty even if the system is in a busy state when the Shepherd occupies only a trivial percentage of CPU usage. Still another issue is to implement the Baseline service daemon as a system service of the NT based systems and configured to start automatically along with the system boot process.

As discussed in the previous chapter, the Baseline service architecture may achieve high adaptivity and usability through taken different physical values of the communication processes for measurement, for example the timeouts count, the bandwidth occupied, the number of connections established and the throughput. Simulations in different kinds of network communication environments should be taken to analyze these physical values to evaluate their specific usability and effectiveness. With enough data acquired, we may try to give a theoretical model of the Baseline service independent of all the physical values to be measured.

References:

[1] CERT. CERT advisory CA-98.01: Smurf IP denial-of-service attack via pings [EB/OL]. http://www.cert.org/advi-sories/CA-98.01.smurf.html, 1998.

[2] CERT. Overview of attack trends [EB/OL]. http://www.cert.org/archive/pdf/attack_trends.pdf, 2002.

[3] CERT. Denial of service attack [EB/OL]. http://www.cert.org/tech_tips/denial_of_service.html#1, 2001.

[4] FERGUSON P, SENIE D. Network ingress filtering: Defeating denial of service attacks which employs IP source address spoofing [EB/OL]. http://www.ietf.org/rfc/rfc2267.txt, 1998.

[5] SANS Institute. Egress filtering [EB/OL]. http://www.

sans.org/y2k/egress.htm, 2000.
[6] SAVAGE S, WETHERALL D, KARLIN A, *et al*. Practical network support for IP traceback[A]. **Proceedings of the 2000 ACM SIGCOMM Conference**[C]. New York, US: [s.n.], 2000. 295-306.
[7] BELLOVIN S. The ICMP traceback message [EB/OL]. http://www.research.att.com/.smb, 2000.
[8] MAHAJAN R, BELLOVIN S M, FLOYD S, *et al*. Controlling high bandwidth aggregates in the network[J]. **Computer Communications Review**, 2002, 32(3): 62-73.
[9] IOANNIDIS J, BELLOVIN S M. Implementing pushback: Router-based defense against DDoS attacks[A]. **Proceedings of the Network and Distributed System Security Symposium** (**NDSS**)[C]. Reston, US: The Internet Society, 2002. 112-123.
[10] KEROMYTIS A, MISRA V, RUBENSTEIN D. SOS: Secure overlay services [A]. **Proceedings of the ACM SIGCOMM**[C]. New York, US: [s.n.], 2002. 61-72.
[11] KRISHNAMURTHY B, WILLS C, ZHANG Yin. On the use and performance of content distribution networks [A]. **Proceedings of the ACM SIGCOMM Internet Measurement Workshop** [C]. New York, US: [s.n.], 2001. 33-44.
[12] SONG D X, PERRIG A. Advanced and authenticated marking schemes for IP traceback [A]. **Proceddings of IEEE INFOCOMM** [C]. Piscataway, US: IEEE Press, 2001. 93-104.
[13] SNOEREN A C, PARTRIDGE C, SANCHEZ L A, *et al*. Hash-based IP traceback [A]. **Proceedings of ACM SIGCOMM**[C]. New York, US: [s.n.], 2001. 3-14.
[14] YAN Jian-xin, EARLY S, ANDERSON R. The XenoService: A distributed defeat for distributed denial of service[EB/OL]. http://www.cl.cam.ac.uk/ftp/users/rja14/xeno.pdf, 2000.
[15] BURCH H, CHESWICK B. Tracing anonymous packets to their approximate source [A]. **Proceedings of the 14th Systems Administration Conference**(**LISA**)[C]. San Francisco, US: USENIX Association Press, 2000. 319-327.
[16] STONE R. Centertrack: An IP overlay network for tracking DoS floods[A]. **Proceedings of USENIX Security Symposium**[C]. San Francisco, US: USENIX Association Press, 2000. 107-118.
[17] YURCIK W, DOSS D, KRUSE H. Challenges to the end-to-end Internet model [A]. **Proceedings of the American Conference on Information Systems** (**AMCIS**)[C]. Atlanta, US: Association for Information Systems, 2000. 289-293.
[18] JANNOTTI J, GIFFORD D, JOHNSON K, *et al*. Overcast: Reliable multicasting with an overlay network [A]. **Proceedings of the Fourth Symposium on Operating Systems Design and Implementation**[C]. San Francisco, US: USENIX Association Press, 2000. 197-212.
[19] BELLOVIN S M. Computer security—An end state?[J]. **Communications of the ACM**, 2001, 44(3): 131-132.

(**责任编辑**: 李育燕)

Biographies:

JIN Shu (1979-), male, born in Nanjing, China. He now studies for his Ph.D. degree in Nanjing University of Science & Technology. He mainly focuses his research on information security (**E-mail**: kimsuepc@public1.ptt.js.cn).

LIU Feng-yu (1943-), female, born in Nanjing, China. She is a Professor and Ph.D. supervisor. She mainly focuses her research on information security, multimedia technologies and artificial intelligence (**E-mail**: Liu.Fengyu@263.net).

XU Man-wu (1944-), male, born in Nanjing, China. He is a Professor and Ph.D. supervisor. He mainly focuses his research on novel software architecture.

Baseline——一种容忍与检测DoS/DDoS攻击的被动方法

金　舒[1], 刘凤玉[1], 许满武[2]
(1. 南京理工大学 计算机科学与技术系, 江苏 南京 210014;
2. 南京大学 计算机系, 江苏 南京 210093)

摘要: 提出一种检测 DoS/DDoS 攻击(拒绝服务攻击/分布式拒绝服务攻击)的"Baseline"服务体系结构,并实现了其中关键的"Shepherd"检测算法。该体系结构通过把服务及其质量是否受到影响的判断指标交由具体的通信进程来定义而具有较强的适应性,可以与各种现有入侵检测系统很好地结合且不给宿主系统增加过多的开销。同时相比其他 DoS/DDoS 检测方式,Baseline 服务无须对 Internet 路由基础设施作出更改来提供支持,是一种十分可行的 DoS/DDoS 检测方案。在理想情况下,Baseline 服务可以做到对 DoS/DDoS 入侵零误报。

关键词: Baseline 服务; 可用性监视; DoS/DDoS 攻击检测
中图分类号: TP 393.08　　**文献标识码**: A　　**文章编号**: 1006-6047(2005)06-0007-07