I.J. Wireless and Microwave Technologies, **2011, 3, 21-28**
Published Online June 2011 in MECS (http://www.mecs-press.net)
DOI: 10.5815/ijwmt.2011.03.04

Available online at http://www.mecs-press.net/ijwmt

A Scalable Simulation Method for Network Attack

Jinsong Wang[a], Wenchao Dou[b], Kai Shi[c]

[a,b,c]*Tianjin Key Lab of Intelligent Computing & Novel Software Technology, Key Laboratory of Computer Vision and System, Ministry of Education, Tianjin, China*

Abstract

In order to found a scalable platform for attack resistance test, this paper proposed a simulation method for network attack. We designed a modular framework and using XML to describe the test cases. We also realized both stateful and stateless attacks by Socket programming and Jpcap packet forging method. Experiment results showed that the system has good scalability and provides a template-based testing circuit.

Index Terms: transportation network; computer network; reliability; strategy

1. Introduction

With the rapid development of network technology, the number of network security threats grows more quickly. As the members of the network interconnection use network protocols and network protocol itself is a highly complex set, also some programmers do not have rigorous programming habit, attacks against network protocols continue happens. In 2006 alone, US-CERT [1] (U.S. Computer Emergency Response Team) had reported 60 million information security incidents. In which 2453 are database security vulnerabilities, and about 1,000 vulnerabilities are related with network protocol; CSI/FBI also reported [2] that in the 597 companies surveyed, 66% of them are using penetration testing techniques and automated tools, hoping to find security vulnerabilities as soon as possible; in 2007, widely deployed Cisco routers in network infrastructure had repeatedly found serious flaws, such as Secunia Advisory: SA23867 [3], CNNVD (China National Vulnerability Database) -200701-136 [4], etc. The attacker can forge network packets directly against network protocol of remote device, these attacks are often a direct threat to the security of the entire internet. As one of the main objectives in network protocol design and implementation process, reliability and security demand are increased to a new level, developing vulnerabilities mining tools of network protocols is urgently needed.

At present, the researchers made a number of related researchs on attack simulation method. SUN Chang-hua [5] analysed DDoS (Distributed Denial of Service) and made a classification for it; GUO Er-wang [6] realized a variety of buffer overflow attacks. These researches mainly focused on attack theory and single type attacks but did not form an intuitive, specific, extendible attack model.

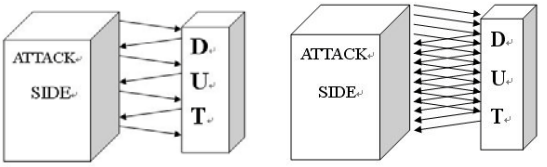
Foundation Item: A National 863 Project(2007AA01Z450); a Key Projects in the Science & Technology Pillar Program of Tianjin(08ZCKFGX00600); and an Information Technology Project of Tianjin(091052012)

In order to provide a better security strategy by studying the principles of attack methods, this paper proposed an attack simulation based testing method and achieved a modular network attack testing system with good scalability, provided a reference for improving network security mechanisms.

2. Attack Resistance Test

In network security test, a commonly used approach is by simulating network attack to make early detection of network security problems and take appropriate measures to reduce the possibility of malicious intrusion; this process is called ART (Attack Resistance Test).

According to the process of attack-side against DUT (Device Under Test) and the relevance among packets' context, an attack process can be divided into two types of stateful and stateless. In a stateful attack process, the attacker tries to send a packet and look forward to getting the response from a DUT, it will not send the next packet until gets the response. After sending the next packet it still has to wait for the corresponding response from the DUT. After one or more packets' interactions, the system will be introduced into a particular state. Only at this time, the packet sent from the attack-side will have attack effect and we can see whether the DUT has been tested to make the right response or trigger vulnerability, as shown in Fig. 1(a) below. In a stateless attack process, the attack-side forges packets to flood a DUT and never looking forward to getting the response from the DUT, each input is independent, we can see the status of DUT at any time and whether the attack triggers vulnerability. Fig. 1(b) shows the case of stateless attacks that did not forge source IP address.



(a) Stateful attack (b)Stateless attack

Fig 1. Attack process

3. Frame Design

Our framework of the testing system is as shown in Fig. 2. The system includes a control desk, a policy editor tool, some test cases and an attack simulator.

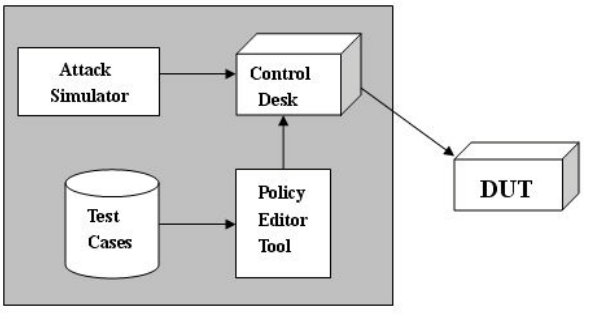


Fig 2. System framework

3.1. Control Desk

The attack-side: A tester uses the attack simulator and policies returned from the policy editor tool to send assembled packets to a DUT (a computer or a network device) for attack resistance test. In this process, the tester can use sniffer tools such as Sniffer Pro or Wireshark installed in the control desk to view and analysis the interactive data between the attack-side and the DUT. For example：

1) For remote attacks, we can monitor returned packets or search whether there is a “The Connection Closed By Remote” like obvious error message.

2) For local tests, if the response from a target device does not meet the RFC (requests for comments) standard, the device is in abnormal state. We can use Syslog, SNMP (Simple Network Management Protocol) to detect the DUT's serious disorders of hardware and software, such as system crash, restart, dead system process or output segment errors, etc [7].

3.2. Policy Editor Tool

The policy editor tool is the core of an ART (Attack Resistance Test) and in vulnerability mining. It is responsible for selecting and parsing one or more attacks from test cases. These attacks may include two types of stateful and stateless or just one type. The interface design of policy editor tool is as shown in Fig. 3.

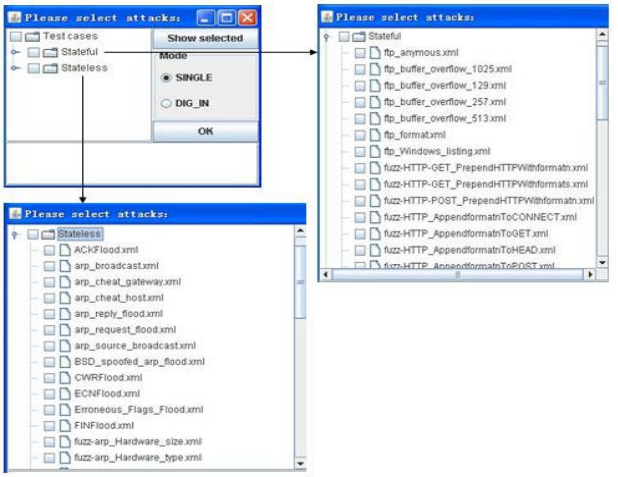


Fig 3. Interface design of policy editor tool

3.3. Test Cases

In order to realize good scalability in our framework, the system uses XML language which has the advantage of flexible data structure description to describe test cases. The XML description can overcome the difficulty of complex parameters in both stateful and stateless test cases, but in order to achieve a complex attack resistance test and build reasonable testing data, tester still need to know norms and standards of each protocol. For system description, there is a corresponding XML type file for each attack, which describes the corresponding characteristics of a specific attack, including attack description, protocol used, field values and other information; for system implementation, we use java DOM (Document Object Model) to parse XML files to obtain field

information of test cases. Furthermore, this system can call the corresponding function to generate random value and add it to the appropriate field according to the type and range of field value.

3.4. Attack Simulator

The attack simulator is installed in the control desk, by editing the parameters: source IP address, destination IP address, source MAC address, destination MAC address, destination port and testing duration, combined with field values returned from policy editor tool to realize specific network attack simulation and display some response of DUT. The interface design of attack simulator is as shown in Fig. 4.

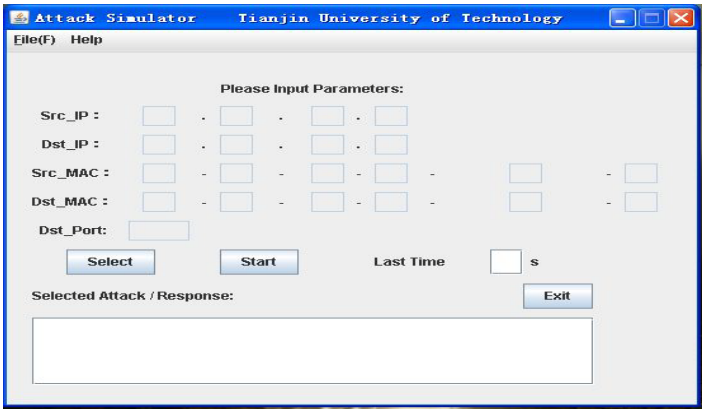


Fig 4. Interface design of attack simulator

The complete process of attack simulation is as shown in Fig. 5.

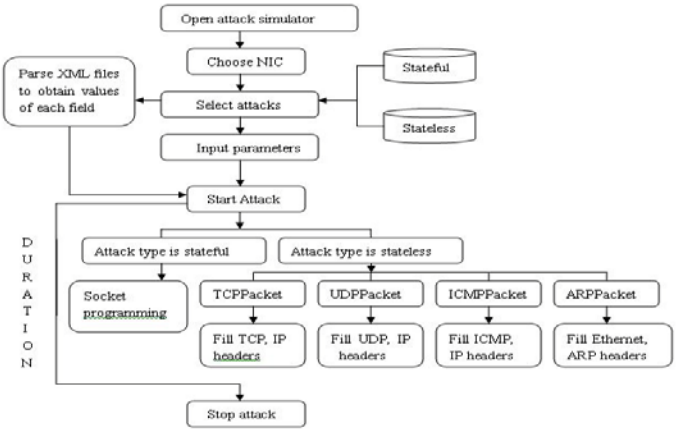


Fig 5. The complete process of attack simulation

We will give a detailed description of the realization of stateful and stateless attacks in the next section.

4. Impelemetation

4.1. Socket programming to realize stateful attack

The stateful attack is also called application-based attack, it requires the interactive data between the attacker and the DUT must according to context. The typical approach is to establish a connection at first by a TCP three-way handshake process.

From the perspective of network security, according to the TCP/IP requirements, the use of TCP protocol for communication need to provide two sequence numbers to ensure synchronous connection and secure communication, it is very difficult to forge sequence number, that is the reason why this article does not use Jpcap to forge packets but use real IP address to ensure the connection synchronized.

A stateful attack is based on the context of the interaction process with the logic of concrete chronological and input accuracy. Therefore, the attacker sent a packet needs to wait until it receives DUT's response, based on which the attacker sends a next packet. Thus, it is required a RTT (Round-Trip Time) which changes according to different testing environment. In order to achieve proper interaction and minimize waiting time more than RTT, system defines the "time" attribute in XML files of stateful attacks which value represents how many milliseconds to wait between two packets sent from attacker. A typical test case is shown in Fig. 6.

```
<?xml version="1.0" encoding="utf-8"?>
<threatSignature>
  <ThreatProperties>
    <EngDesc value="This generic threat sends a long buffer [129 bytes] against an
FTP server.  A buffer overflow attack attempts to crash the service by causing the
service to write to out of bounds memory by going beyond the allocated buffer."/>
  </ThreatProperties>
  <Protocol>
    <Packet type="stateful"/>
  </Protocol>
  <FilledField>
                <Sentence>
                        <Field name="sentence1" value="USER
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"/>
                        <Field name="sentence2" value="wait" time="400"/>
                        <Field name="sentence3" value="PASS
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"/>
                </Sentence>
  </FilledField>
</threatSignature>
```

Fig 6. ftp_buffer_overflow_129

According to the description of above test case, the specific testing process is described as follows:

First, through a three-way handshake with the FTP server (DUT) to establish a connection, then the attacker sends the value of "sentence1" (Expressed by "A", 129 bytes) to input user name, after waiting for 400 milliseconds, sends the value of "sentence3" (Expressed by "A", 129 bytes) for password confirmation, followed again the attacker uses available port for three-way handshake with the FTP server, repeat this process. The logic of this test case is whether the attack-side received the response "331 User name okay, need password" within 400 milliseconds after attacker sent user name string. The meaning is mining buffer overflow vulnerability from FTP server.

The system also implements a http GET flood, using long format string (eg “%n%n%n%n%n%n”) to fill up http Method content, and random length format string to fill up http Method for fuzz testing, etc. It is usually identified with special characters to distinguish data from different upper layer protocols. For example, it must fill each request field with "\r\n" at the end when the upper layer uses http protocol.

Table I shows the main implementation in java code.

Table 1 Java main code for realizing stateful attack

```
Socket dut = new Socket(InetAddress address, int port);
// connect with DUT's IP address and specific port number
dut.setSoTimeout(300);
/* The read() method of socket input stream may cause thread
blocked, so it should use setSoTimeout() method to set a timeout.*/
PrintWriter out = new PrintWriter(dut.getOutputStream(),true);
BufferedReader in = new BufferedReader(new
InputStreamReader(dut.getInputStream()));
//Open outputstream and inputstream.
StringBuffer sb = new StringBuffer(65535);
while (loop) {
      if (in.ready()) {
            int j = 0;
            while (j != -1) {
         j = in.read();
         sb.append((char) j);
       }
       loop = false;
      }
}
// Receive returned data and judge the status of DUT
```

4.2. Jpcap programming to realize stateless attack

Jpcap is a java class library which allows java applications to capture or send packets. In order to provide a public interface to achieve platform independence, Jpcap calls winpcap in windows platform and libpcap in unix platform. The most important class for Jpcap to forge packet is JpcapSender, it is used to send the constructed packets.

According to different protocols the packets used, in Jpcap library, the hierarchy of main packet classes is as shown in Fig.7.

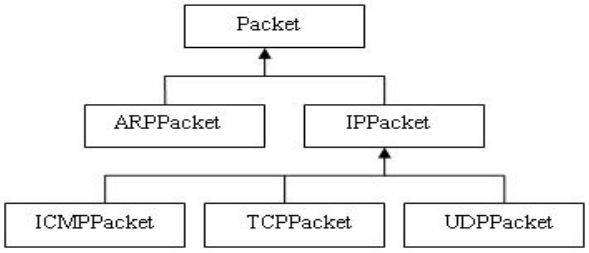


Fig 7. The hierarchy of main packet classes described in Jpcap

Stateless attack also known as network-based attack, the interaction between attack-side and DUT does not according to context, each test case need only describe the fields of a packet and then flooding it, and each field can be forged or set random values. A typical example of SYN Flood, using XML to describe the fields of the packet is as shown in Fig. 8.

```
<TCP_header>
  <Field name="src_port" value="GetDynamicPort" />
  <Field name="dst_port" value="Customize" />
  <Field name="sequence" value="0" />
  <Field name="ack_num" value="0" />
  <Field name="urg" value="false" />
  <Field name="ack" value="false" />
  <Field name="psh" value="false" />
  <Field name="rst" value="false" />
  <Field name="syn" value="true" />
  <Field name="fin" value="false" />
  <Field name="rsv1" value="false" />
  <Field name="rsv2" value="false" />
  <Field name="window" value="GetRandom65535" />
  <Field name="urgent" value="0" />
</TCP_header>
<IP_header>
  <Field name="priority" value="0" />
  <Field name="d_flag" value="false" />
  <Field name="t_flag" value="false" />
  <Field name="r_flag" value="false" />
  <Field name="rsv_tos" value="0" />
  <Field name="rsv_frag" value="false" />
  <Field name="dont_frag" value="true" />
  <Field name="more_frag" value="false" />
  <Field name="offset" value="0" />
  <Field name="ident" value="GetRandom65535" />
  <Field name="ttl" value="GetRandom255" />
  <Field name="srcIP" value="randomIP" />
  <Field name="destIP" value="toIP" />
</IP_header>
```

Fig 8. SYN Flood described by XML

This test case uses TCP as transport layer protocol, uses IP as network layer protocol for forwarding packets, runtime meaning of field value is described in Table II below.

Table 2 Runtime meaning of field value

field	meaning	value	value meaning
src_port	Source port	GetDynam-icPort	generate random port between 1024 and 65535
dst_port	Destination port	Customize	submit in attack simulator GUI
syn	Synchroni-zation bit	true	synchronous request
srcIP	Source IP address	randomIP	generate random IP addresses
destIP	Destination IP address	toIP	submit in attack simulator GUI
ttl	time to live	GetRandom 255	generate random number between 0 and 255

The significance of this test case is to use random source IP addresses flooding request DUT and generate a large amount of semi-connections to occupy DUT's resources.

For the experiment in LAN environment, we use this test case to attack port 139 of DUT with CPU Pentium4 2.40GHz, Memory 512MB, causing DUT's CPU occupancy rate maintains over 98% most of the attack duration.

In the XML description, the system supports a variety of input forms for each characteristic field: one can specify a fixed value, a function similar to “randomIP” or a variable similar to “toIP” submitted by the attacker. The meaning of all fields can refer to Jpcap document [8].

In stateless test cases, the system includes some typical attacks, such as ARP flooding, fuzz testing for all fields of the ARP protocol, all kinds of ICMP reports flooding, land attack, port scanning, and UDP flood.

In the process of select attacks, although you can choose as many test cases for attack resistance test, but if some threats in selected attacks are conflict with the rest of the selected attacks, or excessive number of selected threats, this will cause large quantities consumption of network resources and thus affect the overall testing

performance. In practice, according to our testing results, the best way is to select one type of stateful or stateless attacks in different test plans and limit the number of test cases selected no more than 20 for each test.

5. Conclusion

Based on the theory of stateful and stateless attacks, this paper developed instances of socket programming based stateful attack and Jpcap programming based stateless attack, also introduced several test cases with XML description, the significance is to provide a referable implementation that can simulate network attacks for attack resistance test. In this method, the testing system uses XML language to describe the interface of test cases, tester can create or modify XML documents according to their own requirements to integrate the latest attack library. Furthermore, the tester can also define a function for generating random values to fuzz test a field of a protocol. Through practical use, experiments show that the attack simulator has good scalability and attack effect. In the future, we will test more attack types and try to support more network protocols in the simulator.

Acknowledgment

This work was supported in part by a National 863 Project (No. 2007AA01Z450), a Key Project in the Science & Technology Pillar Program of Tianjin (No. 08ZCKFGX00600), and an Information Technology Project of Tianjin (No. 091052012).

References

[1] US-CERT Vulnerability Notes. http://www.cert.org/.
[2] L. Gordon, P. Loeb, W. Lucyshyn and R. Richardson. 2005 CSI/FBI Computer Crime and Security Survey. Computer Security Institute. 2005.
[3] Secunia Advisory. Cisco IOS Multiple Vulnerabilities. (2009207215). http://secunia.com/advisories/23867.
[4] China Information Security Evaluation Center. China National Vulnerability Database. (2009210218). http://www.cnnvd.org.cn.
[5] SUN Chang-hua, LIU Bin, “Survey on New Solutions Against Distributed Denial of Service Attacks”, ACTA ELECTRONICA SINICA, 2009, 37(7) (in Chinese).
[6] GUO Er-wang, XIA Nai, “A Generic Testbed for Security Enhancement Tools”, Computer Engineering and Applications, 2006.30(in Chinese).
[7] ZHANG Bao-feng, ZHANG Chong-bin, XU Yuan, “Network protocol vulnerability discovery based on fuzzy testing”, J Tsinghua Univ (Sci & Tech), 2009, 49(S2) (in Chinese).
[8] Keita Fujii. Jpcap a Java library for capturing and sending network packets.[2007-10-19]. http://netresearch.ics.uci.edu/kfujii/jpcap/doc/.