



IPv6 网络协议中分段机制安全问题的研究^{*}

王相林, 朱 晨, 孙冬梅, 李明月, 沈清姿

(杭州电子科技大学计算机学院 杭州 310018)

摘 要:以开源 Snort 平台为基础,探讨 IPv6 网络的分组分段机制,对利用 IPv6 分组分段来攻击网络的行为进行了研究。针对其中 IPv6 过小组分段、重叠分段等问题提出了相应的应对措施,设计并实现了针对分段漏洞进行检测的入侵检测机制。通过实验表明,所设计的入侵检测机制能够检测相关异常的 IPv6 分段分组,对 Snort 支持 IPv6 功能进行了改进和优化。

关键词:IPv6 网络安全;入侵检测系统;扩展首部;分段机制

doi: 10.3969/j.issn.1000-0801.2013.10.019

Research of Fragment Mechanism Security in IPv6 Network Protocol

Wang Xianglin, Zhu Chen, Sun Dongmei, Li Mingyue, Shen Qingzi

(School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, China)

Abstract: Based on open source Snort platform, the packet block mechanism in IPv6 network was discussed. The behavior using IPv6 packet fragment to attack network was studied. Against to the problems, such as too little for IPv6 packet segment, overlapping section and so on, it put forward the corresponding countermeasures. The intrusion detection system were designed and implemented for piecewise leak testing. The experiments show that the design of intrusion detection system can detect the abnormal IPv6 packet, and Snort to support for IPv6 function was improved and optimized.

Key words: IPv6 network security, intrusion detection system, extension header, segmented mechanism

1 引言

IPv6 建议最小的 MTU (maximun transmission unit, 最大传输单元)为 1 280 bit,并丢弃所有小于这个大小的分组分段,最后一个分段除外。攻击者通过发送重叠的分片分组,可以将目的节点进行替换;通过发送大量的小分段,攻击者可以让目的节点的分段重组缓冲区溢出,甚至导致系统崩溃。为了有效地保障 IPv6 网络的安全,在 IPv6 环境下设计并实现针对分段漏洞进行检测的入侵检测机制具

有重要的现实意义。

所开展的研究工作以著名的开源入侵检测系统 Snort 为基础,深入研究了 IPv6 分组的分段机制,针对利用 IPv6 分段分组攻击网络的行为进行了深入研究;针对其中过小组分段、重叠分段等问题提出了相应的应对措施,通过在 Snort 中设计并添加支持 IPv6 分段扩展首部的解析模块,对于利用 IPv6 过小组分段分组进行攻击的行为,实现了及时检测和报警;结合 ICMPv6 功能,设计并实现了针对分段漏洞进行检测的入侵检测机制。

^{*} 国家自然科学基金资助项目 (No.61273075, No.61172133), 浙江省大学生新苗人才计划科技成果推广基金资助项目 (No.2012R407057), 杭州电子科技大学优秀学位论文培育基金资助项目 (No.xylw2013005)

2 IPv6 分组分段机制及存在的安全问题

2.1 IPv6 路径 MTU 发现机制

IPv6 路径 MTU 发现机制的过程如图 1 所示。通过采用路径 MTU 发现机制,IPv6 的源节点可以确定源节点到目的节点之间的整个链路中能够传送的最大分组长度,从而可以避免中间路由器的分段处理。RFC2460^[1]规定最小的 MTU 为 1 280 bit,并建议将链路配置为至少可以传送 1 500 bit 的长分组。

2.2 IPv6 分组分段技术

当网络层接收到大于 MTU 值的上层分组时,网络层则会对该分组进行分段。IPv6 和 IPv4 分段分组的重组机制不同,在 IPv4 中有分段和偏移(D 和 M)字段用于标识该分组是否为分段分组,而在 IPv6 中则使用扩展首部来标识该分组是否为分段分组。IPv6 分段扩展首部如图 2 所示,其中 M 值标识是否存在下一个分段首部。

没有进行分段的分组称为原始 IPv6 分组,原始分组

包括两部分:不可拆分部分和可拆分部分^[2]。不可拆分部分包括 IPv6 分组的固定首部、逐跳选项扩展首部、目的地选项扩展首部和路由扩展首部。可拆分部分则包括有效载荷和只能在到达最终目的地时才处理的扩展首部。将原始分组中可分段部分按顺序从左到右分成若干分段,并使用分段扩展首部标识,除了最后一个分段,每一个分段的长度都是 8 bit 的整数倍。IPv6 分组分段的过程如图 3 所示。

IPv6 中只允许源节点对分组进行分段,而不允许中间节点对分组进行分段。如果中间节点接收到分组之后发现分组的大小超过 MTU 值,则该节点向源节点发送一个 ICMPv6 分组(类型为 4,代码为 0)。ICMPv6 分组携带了当前链路的 MTU 值以及一些错误信息,用于表明该分组过大。源节点收到分组过大类型的 ICMPv6 分组,则会根据该分组中的 MTU 值重新设置分组长度的大小,经过多次的试探,得到最终到达目的节点所需要的 MTU 值。此后,源节点主机所有的分组都依据该 MTU 值对分组进行分段。

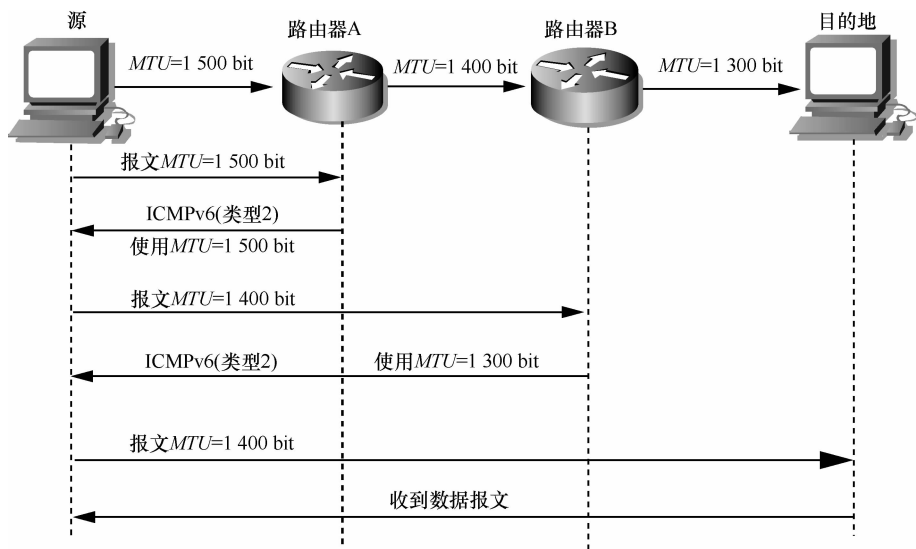


图 1 IPv6 路径 MTU 发现机制



图 2 IPv6 分段扩展首部

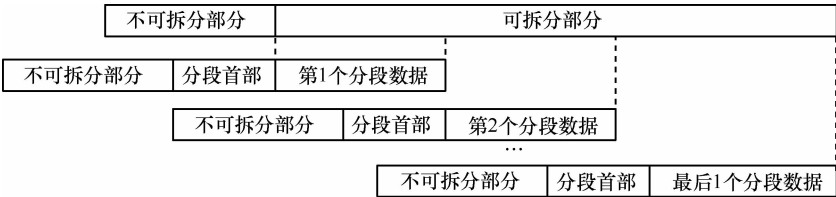


图 3 IPv6 分组分段的过程



2.3 IPv6 分段分段中存在的安全问题

分组分段的一个主要问题是高层信息也许并没有包含在一个分段内部,要确定攻击者的真实意图就需要重组所有的分组,但是对多个分组进行重组又会耗费系统资源。攻击者可以通过使用 IPv4 或 IPv6 分段来隐藏他们的攻击或攻击节点,通过将攻击信息放到多个分段之中,攻击者就可以躲过过滤或检测系统。在 IPv6 中,中间路由器从不执行分段,而是由源节点进行分段。这是因为协议开发人员考虑到中间路由器分段导致的性能影响。

IPv6 分段分组中可能存在的安全问题如下。

- 重叠分段问题在 IPv4 网络中已经引起了很大的关注,恶意用户可以通过重叠分段制造各种攻击以及逃避一些安全监测^[3]。所以 IPv6 主机不允许发送重叠的分段分组,一旦接收到此类型的分组,就应当及时丢弃^[4]。然而在实际网络中,大部分主机以及 IDS 都对 IPv4 和 IPv6 分段分组采用相同的重组机制,对于一些过小组分段(通常小于 100 bit)以及多重扩展首部(将具有攻击特性的分组装在第二个分组中)等的规避攻击,则无法进行检测。因此,攻击者可以利用分段进行重叠分段攻击或乱序分段攻击(分段 ID 不能正确地匹配数据)。
 - 在 RFC2460 中 IPv6 链路定义的 MTU 值并没有硬性规定具体为多少,只是申明“MTU 必须大于 1 280 bit”。可以看出,规定并没有禁止发送过小组分段,因此对于大小为 2 000 bit 的分组并不一定会被分为 1 280 bit 加上 720 bit,也可能被分成其他大小不同的分段分组。在使用隧道技术或 IPv4/IPv6 翻译技术时^[5],分组大小就有可能被切割成小于 1 280 bit 的分组。IPv6 链路定义的 MTU 必须大于或等于 1 280 bit,对于一些非常小型的分组就应该产生警觉,这可能是一次恶意的攻击行为。除非分组是最后分段且 M 字段的比特值设置为 0,否则不应该有小于 1 280 bit 的分组分段。
 - 由于中间路由不允许进行分段和重组,所以包含多个分段首部集的分组是不应该出现的,在 IPv6 分组内应该只有一个由源节点产生的分段首部。
 - 攻击者可以发送分段的一个不完全集合,强迫接收节点等待最后分段。默认分段超时是 60 s,这会消耗接收节点大量的系统资源。
- 对于一些可能的 IPv6 攻击,需要通过对 IPv6 分组进行

重组之后才能进行相关的检测,可以通过对分组的特征信息进行重组与还原,将相同分组的分段进行重组。具体设计思路是从各个分组的分组中提取协议头、协议体的数据,对 IPv6 分组中的特征信息与源地址、目的地址、分段标识等的关键字进行比较,将属于同一分组的分段分组插入到分组文链中,这样就将原来的分段分组变成了数据链。

3 用于 IDS 的 IPv6 分段扩展首部的解析模块设计

在 Snort 中可以根据 IPv6 分组扩展首部中下一个首部的值确定该分组是否为分段分组。在 IPv6 中,分段扩展首部的扩展首部类型值为 44。当 Snort 检测到下一首部的值为 44 的 IPv6 分组,就可以调用 IPv6 扩展首部解析函数对该扩展首部进行解析。

3.1 分段扩展首部的解析模块设计

Snort 中处理 IPv4 分段分组先通过分段预处理器对分组进行重组,这样就能防止攻击信息存放于不同的分段分组中而逃脱检测。在入侵检测系统中,预处理器的主要作用包括两个方面。

- 对具有可疑行为的分组进行检查或修改,以便检测引擎是否能够对其进行正确的解释。因为很多攻击并不能仅仅通过特征匹配检测出来,所以需要预处理器来进行检测。
- 负责对流量标准化以便检测引擎能准确地进行特征匹配。

对于新接收到的 IPv6 分组分段,分段扩展首部的解析模块 IP6Frag 模块通过调用解析函数将分段分组进行解析,并将相关信息存入分段扩展首部中。IPv6 分段扩展首部的定义如图 4 所示。

```
typedef struct _IP6Frag
{
    uint8_t ip6f_nxt; /* 下一个首部*/
    uint8_t ip6f_reserved; /* 保留字段*/
    uint16_t ip6f_offlg; /* 分段偏移值*/
    uint32_t ip6f_ident; /* 标识*/
} IP6Frag;
```

图 4 IPv6 分段扩展首部的定义

在分组解析模块中,通过解析函数 void DecodeIPv6 Options(int type, const uint8_t *pkt, uint32_t len, Packet *p) 将分段分组的相关字段填充到 packet *p 结构当中。存储分组的 packet 部分结构如图 5 所示。

```

IP6Extension *ip6extension; //路由扩展首部
IP6HopByHop *ip6hopbyhop; //逐跳选项首部
IP6HopByHopSupPad *ip6hopbyhopsuppad; //具有超大载荷选项的逐跳选项扩展首部
IP6Dest *ip6dest; //目的站选项首部
IP6Route *ip6route; //路由首部
IP6Route0 *ip6route0; //添加的路由类型为0的首部
IP6Frag *ip6frag; //分段首部
IP6AH *ip6ah; //AH首部
IP6ESP *ip6esp; //ESP首部

```

图 5 存储分组的 packet 部分结构

对接收到的分组先转换为 IPv6 分段分组的格式,之后再对标志位进行判断是否正确赋值。填充的相关过程如图 6 所示。

```

...
IP6Frag *ip6frag_hdr = (IP6Frag *)pkt; //转换为IP6Frag格式
p->ip6_frag_index = p->ip6_extension_count;
p->ip_frag_start = pkt + sizeof(IP6Frag); //指向扩展首部的数据部分
p->frag_flag = 1; //标识是分段分组
p->df = 0; //是否允许分段标志位
p->rf = IP6F_RES(ip6frag_hdr); //保留位
p->mf = IP6F_MF(ip6frag_hdr); //是否有更多分段位
p->frag_offset = ((ntohs((ip6frag_hdr->ip6f_offlg) & 0xfff8) >> 3);
//计算偏移字段值
...

```

图 6 填充的相关过程

Snort 利用二叉树来存储分组,将分组放入缓存,之后当所有分段到达时检查分组。根据源地址、目的地址、分段标识等标识是否是属于同一个分组。经过先对分段分组进行检测是否合法 if (len<sizeof(IP6Frag))再确定是否要插入二叉树中。后收到的分组与已经收到的分组的分段偏移量进行比较,如果相等则表明产生了分段重复,可以将先到的分段进行保留,后到的直接删除。当分段分组到达之后,就需要判定分段分组是否全部到齐。首先可以通过判断第一个分组(offset 的值为 0)和最后一个分组(M 的值为 0)是否到达,之后再检测一个分段分组的偏移量是否等于一个分段的偏移值除去分段首部的长度。对分段分组进行检测如图 7 所示。

对于都到齐的分段分组可以对分组进行重组。IPv6 分段重组主要考虑需要将分段头部去除,将不可分段部分进行整合,之后再不可分段的扩展首部的值进行重新设置。

```

enum {
    IPv6_FRAG_NO_ALERT = 0,
    IPv6_FRAG_ALERT,
    IPv6_FRAG_BAD_PKT,
    IPv6_IS_NOT,
    IPv6_TRUNCATED_EXT,
    IPv6_TRUNCATED,

    IPv6_NEXT
};
CheckIPv6Frag((char *) pkt, len, p)
{
    ...
    switch (next_header) {
        ...
        case IP_PROTO_FRAGMENT:
            if (offset + sizeof(IP6Frag) > size) //检查分段数据是否超出原分组长
            if (!IP6F_MF(frag)) //检测是否有更多的分段
            if ((p->pkth->ts.tv_sec - *(time_t *)hash_node->data) >
                (time_t)ScIPv6FragTimeout()) //检测第一个分段是否超时
                ...
            }
    }
}

```

图 7 对分段分组进行检测

3.2 IPv6 过小分段分组检测设计

由于 IPv6 中不允许中间节点对分组进行分段,在传输链路上其 MTU 值为 1 280 bit。所以在检测到非最后分段但分段分组的大小小于 1 280 bit 时,系统就应该及时发生报警。

下面基于 Frag3 分段预处理器设计针对 IPv6 过小分段分组的检测函数。定义报警时间产生函数如图 8 所示。

```

static inline void EventTinyFragments(Frag3Context *context)
{
    if (!(context->frag3_alerts & FRAG3_DETECT_ANOMALIES))
        SnortEventqAdd(GENERATOR_SPP_FRAG3, //报警类型
            FRAG3_TINY_FRAGMENT, //报警信息的sid编号
            1, //规则版本
            0, //优先级
            3, //规则优先级
            FRAG3_TINY_FRAGMENT_STR, //规则描述信息
            NULL //其他规则信息
        );

    return;
    f3stats.alerts++;
}

```

图 8 定义报警时间产生函数

通过对标志位 M 的检测,确定是否为分段分组。之后检测分段长度是否小于分段分组的最小长度,如果小于,则记录该分段分组的行为。具体检测函数的代码如图 9 所示。



```

static inline int checkTinyFragments(Frag3Context *f3context, Packet *p,
unsigned int trimmedLength){
    if(p->mf)
    {
        if (f3context->min_fragment_length)
        {
            if (p->ip_frag_len <= f3context->min_fragment_length)
            {
                DEBUG_WRAP(DebugMessage(DEBUG_FRAG,
                    "Frag3: Received fragment size(%d) is not more
                    than configured min_fragment_length (%d) \n",
                    p->ip_frag_len, f3context->min_fragment_length));
                EventTinyFragments(f3context);
                return 1;
            }
        }
    }
    return 0;
}

```

图9 具体检测函数的代码

4 实验测试与结果

4.1 实验环境及网络拓扑

实验的测试目的是为了检测改进后的入侵检测机制能否检测到 IPv6 环境中存在的网络攻击。在系统启动之后,使用一台主机模拟攻击主机发送各种攻击类型 IPv6 分组给受攻击主机。将安装有改进后的入侵检测机制的主

机网卡设置为混杂模式,通过分组抓取工具 Wireshark 抓取本主机获取的分组信息,将抓取信息与利用改进机制捕获到的分组与发出的告警信息进行对比,分析该系统是否能够准确地检测 IPv6 环境下 IPv6 过小分段攻击类型的分组。实验网络拓扑如图 10 所示。

测试实验以学校已经建立起来的 IPv6 校园网为基础,将所设计实现的改进机制以“端口镜像”的方式连接在交换机的端口上,以保护学校 IPv6 网站 Web 服务器和校内 FTP 服务器,考虑到当前 IPv6 校园网的流量较小,而且遭受外界 IPv6 攻击的可能性较低,在实际的测试实验中以一台 Linux 主机(即区域 3 中的主机 A)作为 IPv6 攻击行为的实施者测试系统的上述功能。

实际网络中 IPv6 的攻击分组较少,在实验中使用 THC-IPv6-2.0^[6]模拟 IPv6 网络中的相关攻击。THC-IPv6 是一个开放源代码的 IPv6 工具,它提供了一个易于使用的 IPv6 报文构造库,能够方便地模拟各种 IPv6 攻击。同时实验测试各种类型的分组发送和构造使用的是 Scapy^[7]程序,Scapy 是一个 Python 程序,包含精心构造分组的各种定义。使用该程序,用户可以发送、嗅探和构造网络分组。从 Scapy 2.0 版本以后,该程序实现了对 IPv6 的支持,可以生

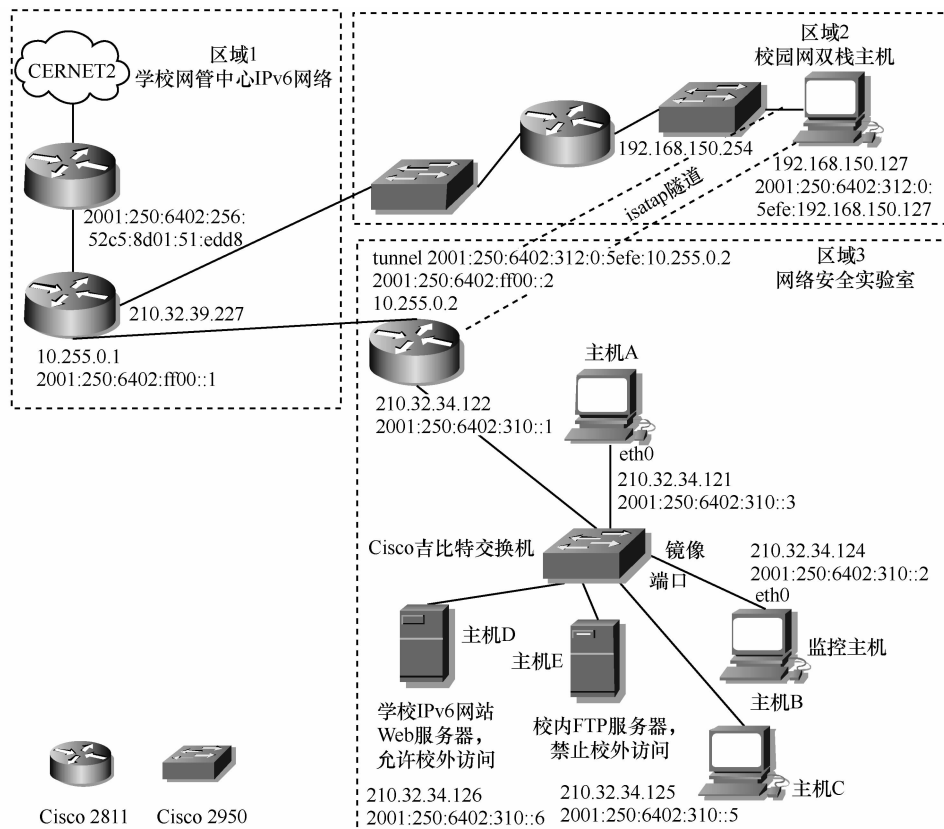


图10 实验网络拓扑

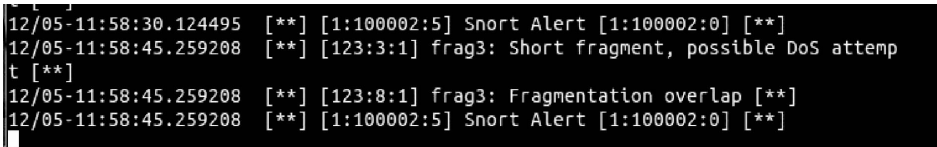


图 11 IPv6 过小分段分组告警提示

成具有特定伪造首部的定制分组。

4.2 IPv6 过小分段分组的检测

使用 Scapy 生成的分段分组,并将其发往目的地。该 IPv6 分组的下一首部是分段分组首部,但该分组存在一个参数问题。从 M 的值为 1 可知,该分组不是最后分段分组,而该分组的大小小于 1 280 bit,而 IPv6 链路定义必须具有大于或等于 1 280 bit 自己的 MTU,除非是 IPv6 分组的最后一个分段。所以对该种类的非合法分组应该产生报警,其记录信息如图 11 所示。

检测过程中对检测到的 IPv6 分段分组进行统计。检测到的分段分组的比例如图 12 所示。

TCP:	28	(9.459%)
IP6:	33	(11.149%)
IP6 Ext:	33	(11.149%)
IP6 Opts:	24	(8.108%)
Frag6:	24	(8.108%)
ICMP6:	3	(1.014%)

图 12 检测到的分段分组的比例

5 结束语

实验证明,本文所设计的入侵检测机制能够检测相关异常的 IPv6 分段分组,对 Snort 支持 IPv6 功能进行了改进和优化。进一步的研究思路是,对 IPv6 分段扩展首部中的两个保留字段进行优化设计,可以用于区分不同的分段安全隐患,另一种想法是,与 IPv6 中其他扩展首部类似,在分段扩展首部中增加选项字段,通过对选项字段的设计,可以提高分段扩展首部在安全性、功能性方面的性能。

参考文献

1 Deering S, Hinden R. Internet Protocol Version 6 (IPv6) Specification. RFC2460, 1998
2 王相林. IPv6 核心技术. 北京:科学出版社, 2009
3 Target-based fragmentation reassembly. [http://www.snort.org/assts/165/target based frag.pdf](http://www.snort.org/assts/165/target%20based%20frag.pdf)
4 RFC5722. Handling of Overlapping IPv6 Fragments, 2009
5 RFC6145. IP/ICMP Translation Algorithm,2011
6 THC IPv6 attack tool kit. <http://freeworld.thc.org/thc-ipv6/>
7 Scapy. <https://www.secdev.org/projects/scapy/>

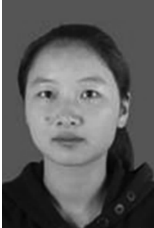
[作者简介]



王相林,男,杭州电子科技大学教授、硕士研究生导师,主要研究方向为网络安全、IPv6 技术、网络性能评价。



朱晨,女,杭州电子科技大学硕士研究生,主要研究方向为网络安全、IPv6 技术。



孙冬梅,女,杭州电子科技大学硕士研究生,主要研究方向为网络安全。



李明月,男,杭州电子科技大学硕士研究生,主要研究方向为网络安全。



沈清姿,女,杭州电子科技大学硕士研究生,主要研究方向为网络安全、IPv6 技术。

(收稿日期:2013-04-23)