基于MapReduce模型的分布式Word文档破解①

陈 勤 方海英 褚一平 黄剑军 (杭州电子科技大学 计算机学院 浙江 杭州 310018)

摘 要: 利用 Word 加密机制中存在的漏洞，实现了与加密密钥长度无关的常量时间破解，同时提出了基于MapReduce 架构实现 Word 文档破解的方案，简化了分布式程序设计。实验表明了该方案的有效性，分布式计算中节点之间的并行消耗低。

关键词: Word 破解；密钥搜索空间；分布式计算

Distributed Decryption of Word Document Based on MapReduce Model

CHEN Qin, FANG Hai-Ying, CHU Yi-Ping, HUANG Jian-Jun

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

Abstract: This paper makes use of the leak in Word encryption mechanism to decrypt word document in constant time, which has nothing to do with the key length. At the same time, an approach based on MapReduce paradigm is presented to to achieve Word document decryption, which simplifies the design of distributed application. The experimental result proves the validity of the proposed approach and the low consumption among distributed computing nodes.

Keywords: word decryption; key search space; distributed computing

1 引言

Microsoft Word 是微软公司的一个文字编辑软件，是目前使用最为广泛的办公软件之一。Word 提供了多种方法限制访问用户文档，加密被编辑文档的内容，以免未经授权者查看和更改，给用户的数据提供了必要的安全保障。但在信息化的今天，随着密码应用范围的扩大，遗忘密码的情况也在与日俱增，一旦忘记密码，用户将无法打开或访问该文档，给用户造成很大的损失。在忘记密码之后如何破解这些密码，尽可能减少损失就成为用户所关心的一个话题。

Word 漏洞方面的研究，岳彩松等人[1]介绍了一种基于Fuzz测试的智能Office应用程序漏洞挖掘方法，研究方向集中在 Office 应用程序的缓冲区溢出漏洞上，同时，他也致力于 MS Office 漏洞利用技术的研究[2]。Hongjun Wu[3]指出了 RC4 算法在 Word 和 Excel 中误用的漏洞。破解 Office 系列文档密码的软件，如 Accent Office Password Recovery[4]、Advanced Office Password Recovery[5]之类的软件，采用穷举法对所有可能的口令字进行测试，一旦被加密文件口令大于 7 到 8 个字符，因其搜索的密钥空间大就基本上不能破解出加密的信息。

本文结合 Word 文档加密漏洞，提出了一种转换搜索空间实现破解的方案，该方案排除了加密文件口令长度对搜索密钥空间的影响。通过分析 Word 加密算法的细节，基于 Hadoop，利用 MapReduce 架构实现了 Word 文档的暴力破解，并对暴力破解的速度进行了测试。测试结果表明，利用 Word 文档加密机制存在的漏洞，Word 加密文档在抵抗本文提出的破解方案时是十分脆弱的。

2 Word加密机制和密钥空间搜索

2.1 Word 加密机制

Office 系列软件所产生的文档均采用基于OLE2的微软复合文档结构，它由若干个 stream和 storage 组

① 基金项目:现代通信国家重点实验室基金(9140C1102060703);杭州电子科技大学校科学研究基金(KYF071506005)
收稿时间:2009-06-24

成。Word 文档包含 Data stream，1Table stream，Word Document stream，Summary Information stream，Document Summary Information stream 等。其中 Data stream 中是图片数据，Word Document stream 中是文本数据，Summary Information stream 和 Document Summary Information stream 中是摘要信息，等等。

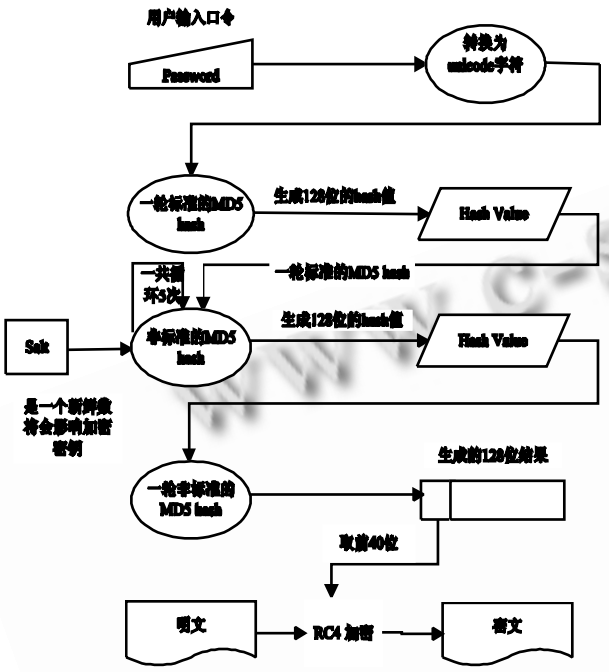

图 1　Word 文档加密算法

当 Word 文档被加密时，文档中只有 1Table、WordDocument 等带有文本、图片等数据的 stream 被加密。1Table stream 中存放验证口令是否正确的信息，它由 3 个 16 字节的域组成，其中第一个域中存放的是一个新鲜数 Salt，用户输入的口令在 hash 计算过程中加入 Salt 值生成 40 位数，该 40 位数为决定 RC4 初始化向量的值，通过该 40 位数可以生成 RC4 加/解密的密钥，计算过程如图 1 所示。第二个域中存放的是系统随机产生的一个 128 位的新鲜数 B 被 RC4 加密后的结果。扩展新鲜数 B 为一个 64 字节的字节串，然后对所得的 64 位字节串计算 MD5 hash，计算出来的 hash 值被 RC4 加密后存放在第三个域中。

测试用户输入的口令正确与否，主要依据 1Table 第二个域和第三个域中存放的数据。用户输入口令打开一个被加密的 Word 文档时，口令加上第一个域中的 Salt 值经过 hash 计算等生成 RC4 加/解密的密钥流。Word 程序读取 1Table 中第二个域和第三个域的值，经 RC4 解密得到值 b 和 c。如果 c 为 b 的 MD5 Hash 值，那么口令正确，否则口令错误。过程如图 2 所示。

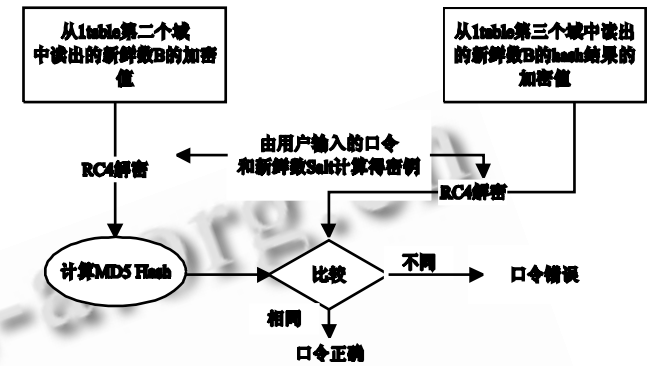

图 2　1Table 校验过程

2.2 密钥空间搜索

Word 的各个版本为了保持向下兼容，默认情况下均使用 97/2000 兼容的加密算法。该加密方法在使用时存在一个漏洞，加密强度在最好的情况下也仅相当于 40 位密钥的加密强度，利用该漏洞可以在常量时间内破解 Word 文档。传统的暴力破解方法是猜测用户口令，每猜一个口令校验程序就加上 1Table 中存储的 Salt 按图 1 所示的流程计算出决定 RC4 初始化向量的 40 位数，再用它产生 RC4 初始化向量，继而使用 RC4 解出第二个域和第三个域中存放的新鲜数 B 及其 hash 的明文，校验程序再次计算新鲜数的 MD5 hash，将其结果与第三个域中存放的结果比较，如果二者相同，则说明用户输入的口令是正确的，反之，则说明用户输入的口令不正确。如果正确，则停止攻击返回正确的口令字；如果不正确，就继续猜测下一个口令。

绕开猜测用户口令字的传统模式，采用直接猜测 40 位决定 RC4 初始化向量的数的方法，省去由用户口令字到生成决定 RC4 初始化向量的 40 位数之间多次计算 MD5 hash 的时间开销，每猜一个，按前一节中提到的 1Table 的校验流程逐个测试直至正确停止攻击。然后用此 40 位数产生的 RC4 解密密钥流，破解出整个文档的明文。

分析比较密钥空间搜索可以发现，与传统方法相比，当口令长度大于一定数目时，直接猜测 40 位的

效果是非常明显的。采用传统的破解方法，假设用户的口令只含数字，猜测一个最大长度为 n 位的口令字，搜索的空间等于 $10^n+10^{n-1}+...+10$。而采用直接猜测 40 位时，搜索的空间只为决定 RC4 初始化向量的那个 40 位数的空间，即 2^{40}(近似于 10^{12})。当用户选择了一个强度较大的口令时，如 n>=12，采用直接猜测 40 位的方法更高效。实际情况中，当用户口令包含字母，大小写以及特殊字符时，传统方法能够破解成功的口令长度将进一步减小。

3 设计与实现

3.1 MapReduce 模型

MapReduce 是 Google 公司首先提出的一种能在大型计算机集群上并发地处理海量数据的框架模型。它是一种简化的分布式编程模式，Map 和 Reduce 是该模型的两大基本操作，用户通过这个 Map 函数处理 key/ value(键/值)对，产生一系列的中间 key/value 对，并且使用 Reduce 函数将具有相同 key 值的中间键值对聚集起来，将结果输出。

MapReduce 的运行时系统会解决输入数据的分布细节，跨越机器集群的程序执行调度，处理机器的失效，并且管理机器之间的通讯请求。Hadoop[6]是 Apache 开源社区开发出的 MapReduce 的一个 Java 实现，提供了在由通用计算设备组成的大型集群上执行分布式应用的框架。

Word 破解时，搜索密钥空间的计算量大，实现速度较慢，实际应用时受到不少制约，不适合高速实现的应用场合。而 Hadoop 以一种可靠、高效、可伸缩的方式进行处理。它具有可靠性，假设计算元素和存储失败，仍有维护多个工作数据副本，确保能够针对失败的节点重新分布处理。Hadoop 以并行的方式工作，通过并行处理加快处理速度，可以保证高效性。Hadoop 还具有可伸缩的，能够处理 PB 级数据。本文破解任务即搜寻正确的决定 RC4 初始化向量的 40 位数，首先将含有 240 个密钥的密钥空间进行分割，而后将分割后的密钥块通过输入传递给并行的 Map 操作，各 Map 操作对各自的密钥块进行破解运算，在密钥块取值范围内进行搜索，顺序依次测试每个可能的 40 位数，对每一个取值调用校验函数 verify_pwd（）测试其是否就是搜索目标，最后将正确的 40 位数传递给主程序解密 Word 文档。系统主要的数据流程如图 3 所示。

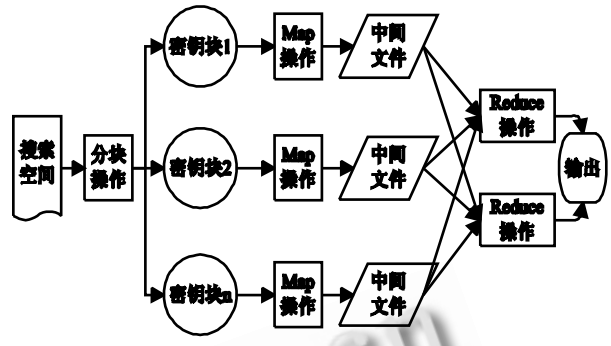

图 3 系统主要数据流程

3.2 软件实现

在 Hadoop 中一次计算任务称之为一个 job，可以通过 JobConf 对象设置运行 job，然后通过 JobClient 类的 runJob 静态方法来运行这个 job。首先根据 MapReduce 模型，确定所需 Map 任务总数。在配置文件中设定单机执行的 Map 数，在程序中进行读取，并乘以机器数即为所需 Map 总数。然后将搜索空间按 Map 数进行等分，实际产生的 Map 数根据密钥块数确定。每次 Map 操作都对一个密钥块进行破解运算。为此，为每个 Map 任务创建一个输入文件，将搜索密钥块起始值与结束值写入文件，作为 Map 操作的输入。通过 Hadoop 中的 JobConf 定制计算任务，进行初始化配置。另外，将读取 MS Word 文件的函数库解析文件写入分布式文件系统中，由所有 Map 操作共享。最后为 JobConf 设置各类配置信息，然后启动任务：

```
JobConf conf = new JobConf (Decrypt.class); //读取 hadoop 配置
conf.setMapperClass(DecryptMapper.class); //设置 Map 操作实现类
conf.setReducerClass(DecryptReducer.class); //设置 Reduce 操作实现类
conf.setInputKeyClass(BytesWritable.class); //设置输入 key 的类型为 BytesWritable
conf.setInputValueClass(BytesWritable.class);//设置输入 value 的类型为 BytesWritable
conf.setInputFormat(SequenceFileInputFormat.class);//设置输入文件格式
conf.setOutputFormat(SequenceFileInputFormat.class);//设置输出文件格式
conf.setNumMapTasks(numMaps);//设置 Map 操
```

```
作数
    conf.setNumReduceTasks(1); //设置 Reduce 操作数，写入一个文件
    conf.setOutputKeyClass(BytesWritable.class);//设置输出 key 的类型为 BytesWritable
    conf.setOutputValueClass(BooleanWritable.class);// 设置输出 value 的类型为 BytesWritable
    JobClient.runJob(conf);//启动任务
    // Map 操作前通过 configure 函数从 XML 配置文件中获得参数，参数包括 1Table 中的第二和第三个域的值，用于检验 40 位是否正确
    public void configure(JobConf conf) {
    // RC4 加密后的新鲜数
    String s_salt = conf.getString("Decrypt.salt", 0);

    String s_hashedSalt =conf.getString("Decrypt.hashedSalt", 0); //经过 RC4 加密及一轮 MD5 hash 后的新鲜数
    ……
    }
    //map 函数进行破解运算
    public void map(BytesWritable key, BytesWritable value,
    OutputCollector <BytesWritable,BooleanWritable > output, Reporter reporter)throws IOException{
    //从输入中取得密钥块始末值
    byte [] begin_pwd = key.getBytes();
    byte [] end_pwd = value.getBytes();
    while( compare( begin_pwd, end_pwd ) < = 0 )
    {//验证密钥
    if(verify_pwd(begin_pwd,s_salt,s_hashedSalt ) == 0 ) {//输出正确密钥
    output.collect(new BytesWritable(begin_pwd), new BooleanWritable(true) );
      break;
      }
      plus( begin_pwd, 1 ); //密钥递增
      }
    }
```

由于破解运算并没有归约过程，所以程序中省略了 Reduce 操作。DecryptReducer 类实际上实现的是一个空操作。此外，由于 Map 操作最终的结果是得到正确的 40 位数，中间过程也没有必要进行合并，同样省略了 Combiner 的设置。当某一 Map 操作得到的正确的 40 位数，宣布任务计算完毕，主程序取得该 40 位数解密 Word 文档。

4 实验结果与分析

为检测软件性能，在局域网内用 16 台装有 Linux 系统的电脑配置 Hadoop 分布式集群，测试所用电脑 CPU 均是 Pentium 4，主频 2.8GHz，内存 1G。由于 Word 文档的破解速度较慢，而密钥空间大，因此本文测试了部分密钥块，3200 块，每个密钥块含 2^{24} 个密钥，配置单机运行 200 个 Map 操作。测试结果与先前实现的局域网内 16 台机器分布式测试结果相比较，结果见表 1。

表 1　比较结果数据表

次数	原来多机方案	Hadoop 框架方案	节约时间
1	9 小时 32 分 3 秒	8 小时 24 分 18 秒	1 小时 7 分 45 秒
2	9 小时 40 分 3 秒	8 小时 26 分 47 秒	1 小时 13 分 16 秒
3	9 小时 47 分 25 秒	8 小时 46 分 12 秒	1 小时 1 分 13 秒
4	9 小时 43 分 26 秒	8 小时 42 分 2 秒	1 小时 1 分 24 秒

那么，从以上对比测试数据中，排除机器性能，机器上其他程序的运行，网络偶然性等影响，我们可以得出这样的结论：在破解中引入 Hadoop 框架，确实性能优于原有的简单多机处理模式。另外，设定不同的 Map 值进行测试，随着 Map 数量增加，在不同处理能力的机器上的任务日益均衡，从更大程度上开发了并行度。处理的复杂度以及集群机器的数量，这些都是需要以后考虑的情况，做到任务的最优配置。

根据测试结果，分析每个密钥块破解时间，Hadoop 框架方案平均每台电脑的计算速度大约是每秒搜索 108629 个密钥，若有 100 台电脑持续计算的话，完成所有的密钥块破解，破解出正确的决定 RC4 初始化向量的 40 位数仅仅需要一天多一点的时间，从而成功解密 Word 文档。但由于破解与正确 40 位数所在的密钥块有关，平均情况下只需搜索一半的密钥块即可找到正确的 40 位数从而破解 Word 文档，

(下转第 193 页)

(上接第 182 页)

仅仅需要不到半天的时间。考虑到 Hadoop 集群，如果能利用社区局域网内几百台乃至几千台联网机器的闲置 CPU 资源，破解尤为乐观。

5 结语

本文通过对 Word 加密算法的研究分析，利用该加密机制存在的漏洞转换破解思路，提出了基于 Hadoop，采用 MapReduce 技术提高 Word 文档破解效率的方案。实验证明 Hadoop 框架方案为在低端计算机组成的机群上实现分布式计算提供了方便灵活的平台。

参考文献

1 岳彩松,李建华,银鹰.基于 Fuzz 的 MS Office 漏洞检测.信息安全与通信保密, 2007,(9):111－113.

2 岳彩松.MS Office 漏洞挖掘与利用技术研究[硕士学位论文].上海:上海交通大学, 2008.

3 Hongjun wu.The misuse of RC4 in Microsoft Word and Excel.http://packetstorm.setnine.com/papers/cryptography/007.pdf.

4 AccentSoft Utilities. http://www.accentsoft.com/.

5 Elcomsoft. http://www.elcomsoft.com/aop.

6 Http://hadoop.apache.org/.