

一种基于 TTCB 的容侵原子多播协议研究*

周 华, 孟相如, 张 立

(空军工程大学 电讯工程学院, 西安 710077)

摘 要: 对基于可信实时计算基(TTCB) 的分布式容侵系统的体系结构进行了研究, 针对分布式容侵系统中的一致性问 题, 实现了利用 TTCB 服务的容侵原子多播协议。在进程总数大于两倍恶意进程个数的条件下, 该协议可以满足正确结果的一致性要求, 达到了入侵容忍的目的。最后证明了该协议算法的正确性。

关键词: 入侵容忍; 原子多播协议; 一致性; 可信实时计算基

中图分类号: TP309. 2 文献标志码: A 文章编号: 1001-3695(2008) 07-2174-03

Study on atomic multicast protocol for intrusion-tolerance based on TTCB

ZHOU Hua, MENG Xiang-ru, ZHANG Li

(Telecommunication Engineering Institute, Air Force Engineering University, Xi 'an 710077, China)

Abstract: This paper studied the architecture of a distributed intrusion-tolerance system based on trusted timely computing base(TTCB) , and proposed an atomic multicast protocol for intrusion-tolerance using TTCB services to solve the consensus problem in distributed intrusion-tolerance system. When the total of processes is more than twice the number of malicious processes, the protocol can satisfy the consensus and achieve the intrusion-tolerance. It proved the Correctness of the protocol.

Key words: intrusion-tolerance; atomic multicast protocol; consensus; trusted timely computing base(TTCB)

世界范围内系统的广泛互连在计算机服务行业产生了重要的社会经济价值, 然而这种价值却受到了网络恶意攻击的影响^[1]。传统的网络安全技术已不能完全保护网络系统不受攻击, 因此产生了一种新的安全措施——入侵容忍。入侵容忍的主要思想就是承认系统中存在可以被攻击者利用的脆弱点, 并构建一种可以容忍一定数量故障(包括攻击和入侵) 的系统^[2]。目前许多容侵系统均采用了分布式的体系结构。然而, 分布式容侵系统的一个困难就是一致性问题^[3]。一致性协议要求所有复制品的进程均提议一个值, 然后对最终决定某个值达成一致。但是系统中因入侵而产生的任意行为的恶意进程可能会影响结果的一致性。假定分布式系统中进程总数为 n , 最大恶意进程个数为 f 。文献[4] 需要 $n \geq 3f + 1$ 才能保证结果的一致性; 文献[5] 要求 $n \geq 2f + 1$, 但其体系结构中客户端与服务器直接交互, 存在恶意客户端对服务器进行攻击的缺点; 文献[6] 中进程总数减少到 $n \geq f + 2$, 但是没有解决 sender 是恶意进程以及各个进程提交消息的全序(total order) 问题。

本文采用了基于 TTCB 的容侵分布式体系结构, 并在客户端与服务器之间引入了代理服务器, 可以有效阻止恶意客户端对服务器的直接攻击。系统中实现了容侵原子多播协议, 在最大恶意进程个数为 f 的条件下, 该协议只需要 $n \geq 2f + 1$ 就能保证结果的一致性。该协议还克服了 sender 是恶意进程的问题, 并可以保证各个进程以相同的顺序提交消息。

1 体系结构与 TTCB

本文的容侵系统主要由客户端、代理服务器和服务器组成。主代理与服务器以及服务器之间通过负载网络进行连接。

笔者认为负载网络是可靠通道, 即如果发送方与接收方均是正确的, 它满足两个条件: a) 消息最终被接收; b) 传输通道中的消息内容不会被篡改。系统的时间模型总体上是异步的, 除了服务器的本地安全内核——可信实时计算基(TTCB)^[7], 所有的 TTCB 通过专有控制通道进行连接。容侵系统的体系结构如图 1 所示。

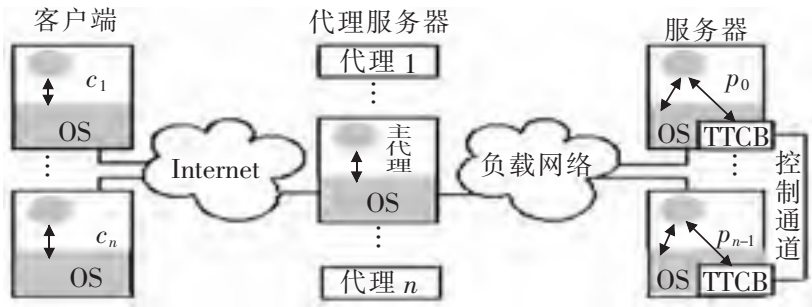


图 1 容侵系统的体系结构

1.1 TTCB

TTCB 是一种安全的实时分布式模块, 它植入在主机内部并与操作系统隔离, 协助分布式协议的正确执行。将每个主机内部的 TTCB 叫做本地 TTCB, 所有本地 TTCB 通过控制通道进行连接。本地 TTCB 是 fail-silent 的, 它们只会因为崩溃才失效。任何应用进程均不能破坏 TTCB 的正确运行, TTCB 之间的交互不会发生故障, 也不会产生错误结果。每个本地 TTCB 拥有一个时钟, 并且所有的时钟均是同步的。

TTCB 通过接口向进程提供有限的服务, 这些服务主要分为两类, 即安全服务和时间服务^[7]。本文中的原子组播协议使用了 TTCB 的三个服务:

a) 本地认证服务(LA)。该服务允许进程与本地 TTCB 安全通信。服务在进程运行之前认证本地 TTCB, 并为 TTCB 与

收稿日期: 2007-06-20 修回日期: 2007-09-04; 基金项目: 陕西省自然科学基金资助项目(2005517)

作者简介: 周华(1981-), 男, 博士研究生, 主要研究方向为计算机网络安全(zhoumiaomiao_2005@ 126. com); 孟相如(1963-), 男, 教授, 博导, 主要研究方向为宽带网络、信息处理; 张立(1981-), 男, 博士研究生, 主要研究方向为计算机网络故障诊断。

进程建立一个共享的对称密钥。这个密钥可以作为两者之间的安全通道,用来加密和认证它们之间的通信。

b) 可信块协商服务(TBA)。该服务是安全协议的主要组成部分。在对一组进程提议的值协商之后,该服务就将达成一致的某一个值提交给进程。TBA 服务只应用于协议中的重要步骤。

c) 可信绝对时间戳服务(TAT)。该服务提供了具有全局意义的时间戳。

1.2 代理服务器

任意选择一个代理服务器作为主代理,用来接收客户端的服务请求,并将请求转发给服务器中的 sender;接收所有服务器返回的结果,将 $f+1$ 个不同服务器返回的相同值作为最终结果返回给客户端。因为系统中最大恶意进程个数为 f ,所以 $f+1$ 个值中至少有一个是正确进程返回的,保证了结果的正确性。主代理转发给 sender 的消息格式为 REQUEST, addr, num, cmd, vec 。其中: REQUEST 表示消息的类型; addr 表示客户端的地址; num 表示请求消息的序列号,在主代理中设置一个计数器,以连续递增的方式给同一个客户端的请求加上序列号; cmd 表示需要执行的请求命令; vec 是消息认证码(MAC)^[5,8] 向量。服务器向主代理返回的消息格式为 REPLY, addr, num, rst 。其中: REPLY 表示消息类型; addr 表示服务器的地址; num 表示请求消息的序列号; rst 表示返回的结果。

若主代理在转发请求消息之后,在时间 T_{timeout} 内没有收到 $f+1$ 个不同服务器返回的相同值,那么它就将请求消息发送给另外 f 个不同的服务器。这样可以保证至少有一个正确的服务器进程接收到请求消息,因而在服务器之间转发该消息。

为了防止恶意的 sender 篡改消息,在请求消息中加入了 MAC 向量。主代理通过 MAC 算法与它和各个服务器之间的共享密钥计算出 MAC 值并加入到请求消息中,每个服务器通过验证自身的 MAC 值来判定消息是否被 sender 篡改。代理服务器中存在安全检测机制,用于检测客户端的恶意请求或攻击行为,以及主代理是否出现故障或被攻克。若主代理出现故障或被攻克,则随机选择另一个代理服务器作为主代理。

1.3 服务器

将每个服务器抽象为一个进程 p_i , 则 $P = \{ p_0, p_1, \dots, p_{n-1} \}$ 表示所有服务器的集合。任意选择 $p_k \in P$ 为 sender,用来接收主代理的请求消息,并采用多播方式将请求消息转发给其他进程。每个进程接收到消息后,首先验证其有效性,然后采用同样的方式转发给除 sender 外的所有进程。每个进程发送 $od+1$ 次以保证所有正确进程均能接收到请求消息。其中 od 称为冗长度^[6]。如果一个进程在发送者发送 $od+1$ 次后还没有收到消息,那么就认为这个进程出现了故障或受到攻击。 od 的具体大小需要根据实际情况确定。关于进程之间转发消息的过程将在协议算法实现中具体论述。

2 可信块协商服务(TBA)

可信块协商服务由 TTCB_propose, TTCB_decide 和 decision 函数实现。现在详细介绍一下服务中的前两个接口函数。

```
outp ← TTCB_propose( eid, elist, tstart, decision, value)
outd ← TTCB_decide( eid, tag)
```

进程在提议一个值时调用 TTCB_propose。其中: eid 是进程在调用之间通过本地认证服务获取的惟一标志符; elist 是所有

参与该服务的进程标志符列表; tstart 是通过时间服务获取的时间戳(理想情况下服务应该在 elist 中所有进程均提议一个值之后才能执行,但是恶意进程可能无限期地延迟提议时间导致服务不能执行。为了避免这种情况,在 tstart 时间后服务立即执行并不再接受任何其他提议); decision 表示如何计算即将决定的值; value 表示提议的值。TTCB_propose 返回一个含有两个字段的结构体 outp: outp. error 表示返回的错误码; outp. tag 表示服务执行的惟一标志符。TTCB 通过 (elist, tstart, decision) 可知道不同进程调用的 TTCB_propose 是否属于同一个服务执行。

进程通过调用 TTCB_decide 函数获取服务的结果值。Tag 是由 TTCB_propose 返回的惟一标志符,用于标志一个服务执行实例。Outd 是由四个字段组成的记录: outd. error 是返回的错误码; out. value 是决定的值; outd. proposed-ok 是由 $N_{\text{elist}}(N_{\text{elist}})$ 表示 elist 中进程总数) 位组成的掩码,每一位表示 elist 中其相应进程提议的值是否就是最后决定的值; out. proposed-any 也是掩码,但它表示的是哪些进程提议的是任意值。

3 容侵原子多播协议的实现

3.1 容侵原子多播协议

服务器进程通过原子多播协议提交消息。该协议具有以下四个特性:

a) 有效性。如果一个正确进程以多播方式发送消息 M , 所有的 MAC 均为有效,那么某一个正确进程最终将提议 M 。

b) 一致性。如果一个正确进程提议消息 M , 那么所有正确进程最终将提交 M 。

c) 完整性。对于任意一个标志符 ID, 每个正确进程最多只能提交一次标志符是 ID 的消息 M 。如果 M 的发送者是正确的, 那么 M 先前就是由该发送者转发的。

d) 全序性。如果两个正确进程均提交消息 M_1 和 M_2 , 那么这两个进程将以相同的次序提交这两条消息。

协议算法主要由三部分组成,即初始化、sender 协议以及接收方协议。

```
算法 1 原子多播协议(对于所有  $p_i \in P, i = (0, 1, \dots, n-1)$ )
Initialization:
1: elist ← all eids of processes in P
2: msg_id ←          //set of  $M_{\text{REQ}}$ . num
3: prepare_deliv ←          //set of M preparing for delivery
4: minnum ← 1
Sender:
5: tstart ← TTCB_getTimestamp( ) +  $T_0$ 
6:  $M \leftarrow ( \text{elist}, \text{tstart}, M_{\text{REQ}} )$  //construct message M
7: propose ← TTCB_propose( elist, tstart, TTCB_TBA_RMULTI-
  CAST, H( M) ) //propose M
8: multicast M for  $od+1$  times to elist except sender
9: Atomic_deliver( M)
Recipients:
10: Get_msg( M) //get message M
11: if verify_mac(  $M_{\text{REQ}}$ . vec[  $p_i$  ] ) then propose ≠ TTCB_propose
  ( M. elist, M. tstart, TTCB_TBA_RMULTICAST, )
12: do decide ← TTCB_decide( propose. tag) //decide the result
13: while ( decide. error ≠ TTCB_TBA_ENDED)
14: while ( H( M) ≠ decide. value) do Get_msg( M) od
15: multicast M for  $od+1$  times to elist except sender
16: Atomic_deliver( M)
When Atomic_deliver( M) is called do:
17: msg_id ← msg_id ∪ {  $M_{\text{REQ}}$ . num}
18: prepare_deliv ← prepare_deliv ∪ {  $M_{\text{REQ}}$ }
```

```
19: minnum ← min( msg_id)
20: while ( MREQ. num = minnum) and ( msg_id ≠ ) do
21: msg_id ← msg_id-{ MREQ. num}
22: prepare_deliv ← prepare_deliv-{ MREQ}
23: minnum ← minnum + 1
24: deliver( M) od
```

初始化过程将所有进程的 eid 存放在 elist 中, msg_id 是存储请求消息 M_{REQ} 序号 $M_{REQ}.num$ 的数组; prepare_deliv 是存储准备提交的消息数组; minnum 表示 msg_id 中序号的最小值。

进程之间转发的消息 M 由(elist, tstart, M_{REQ}) 组成。其中: elist 是符合 TTCB 服务格式的 eid 列表, 列表中第一个元素是 sender 的 eid, 其他元素为接收方的 eid; tstart 是服务的时间戳; M_{REQ} 表示请求消息。协议的每一次执行实例由惟一的标志符(elist, tstart) 表示。Get_msg() 从消息 M 中读取(elist, tstart, M_{REQ})。假定存在一个垃圾消息处理器, 其作用就是销毁已经提交过的相同消息。

Sender 协议首先计算 tstart, 并根据主代理的请求消息 M_{REQ} 以及 elist, 构建消息 M (5、6 行)。一个服务执行由 elist、tstart 和 decision 函数决定 (7 行)。Sender 和所有接收方均使用相同的 decision 函数 TTCB_TBA_RMULTICAST。这个函数表示将选择由 elist 中第一个进程提议的值作为结果值(即 sender 提议的值)。Sender 提议的值是消息 M 的哈希值 $H(M)$, 并通过控制通道发送给接收方。哈希函数是一种单向加密函数, 根据哈希函数的性质^[8], 攻击者是不能够通过哈希值获取原始消息的。Sender 将消息以多播方式转发给其他进程, 然后执行 atomic_deliver() 提交消息 M (8、9 行)。接收方调用 get_msg() 函数等待接收消息, 并通过 MAC 来验证消息的合法性(10、11 行); 然后调用函数 TTCB_propose 获取标志服务的 tag, 接着循环调用 TTCB_decide, 确保协议等待服务结束(12、13 行)。如果接收到的消息与 sender 转发的消息不同, 那么 $H(M)$ 就会与服务返回的值 decide. value 不同。在这种情况下, 协议需要等待接收到正确的消息(14 行); 然后采用多播方式将消息转发给除 sender 外的所有进程, 最后执行 atomic_deliver() 提交消息 M (15、16 行)。当执行 atomic_deliver() 提交消息 M 时, 先将 $M_{REQ}.num$ 和 M_{REQ} 分别存放在数组 msg_id 和 prepare_deliv 中, 然后找出 msg_id 中的最小值(17~19 行)。如果 $M_{REQ}.num$ 是最小值, 则从数组 msg_id 和 prepare_deliv 中移出 $M_{REQ}.num$ 和 M_{REQ} , 同时将 minnum 加 1, 最后提交消息 M (20~24 行)。

3.2 正确性证明

下面对算法 1 所描述的协议是否满足原子多播协议的四个性质进行证明。

定理 1 有效性。如果一个正确进程以多播方式发送消息 M , 其中所有的 MAC 都是有效的, 那么某一个正确进程最终将提议 M 。

证明 设消息 $M = (elist, tstart, M_{REQ})$, 任意一个正确进程 $p_i \in P$, p_i 以多播方式发送消息 M 给其他进程(8 行或 15 行)。设另一个任意的正确进程 $p_k \in P$, 根据可靠通道的性质, p_k 最终接收到消息 M 。 p_k 通过验证其 MAC 可以证明消息 M_{REQ} 没有被篡改(11 行), 又因为 p_i 是正确进程, 那么消息 M 是正确的, 所以可以执行 12~16 行。根据 23 行, 数组 msg_id 中 $M_{REQ}.num$ 最终会成为最小值, 所以消息 M 最终将会被 p_k 提交(24 行)。

定理 2 一致性。如果一个正确进程提议消息 M , 那么所有正确进程最终将提交 M 。

证明 任意一个正确进程 $p_i \in P$, 若 p_i 提议一条消息 M , 那么 p_i 在提议之前以多播方式发送了消息 M 。根据定理 1, 存在一个正确进程 $p_k \in P$ 最终提交 M , 而 p_k 在提交之前同样会转发 M , 因此所有正确进程都会接收到来自其他正确进程的消息 M , 并且最终提交 M 。

定理 3 完整性。对于任意一个标志符 ID, 每个正确进程最多只能提交一次标志符是 ID 的消息 M 。如果 M 的发送者是正确的, 那么 M 先前就是由该发送者转发的。

证明 消息 M 的标志符 ID 是(elist, tstart, M_{REQ}), 一个 TBA 服务执行实例由惟一的标志符(elist, tstart) 表示, 所以标志符是 ID 的消息 M 只能由 TBA 执行一次, 正确进程只能提交一次消息 M 。根据可靠通道的性质可知, 定理后半部分是正确的。

定理 4 全序性。如果两个正确进程均提交消息 M_1 和 M_2 , 那么这两个进程以相同的次序提交这两条消息。

证明 因为所有进程都提交 msg_id 中最小序号所对应的消息, 若 $M_2.M_{REQ}.num > M_1.M_{REQ}.num$, 那么这两个进程将均先提交 M_1 然后提交 M_2 , 所以它们以相同次序提交这两条消息。

4 结束语

入侵容忍作为信息安全领域的前沿研究课题, 正越来越受到各方面的重视。入侵容忍的基本体系结构是分布式的, 但是在存在入侵的情况下, 如何对分布式结果达成一致是一个重要问题。本文在分布式容侵系统中利用 TTCB 服务实现了容侵原子多播协议, 有效地解决了一致性问题。在最大恶意进程个数为 f 的情况下, 该协议只需分布式系统中进程总数 $n \geq 2f + 1$ 就可以满足正确结果的一致性要求, 达到了入侵容忍的目的。

参考文献:

[1] VERSSIMO P, NEVES N F, CACHIN C. Intrusion-tolerant middleware: the road to automatic security[J]. IEEE Security & Privacy, 2006, 4(4): 54-62.

[2] CORREIA M, NEVES N F, LUNG L C, *et al.* Byzantine-resistant consensus based on a novel approach to intrusion tolerance[C] // Proc of the 10th Pacific Rim International Symposium on Dependable Computing. Tahiti, French Polynesia: [s. n.], 2004.

[3] MONIZ H, NEVES N F, CORREIA M. Randomized intrusion-tolerant asynchronous services[C] // Proc of International Conference on Dependable Systems and Networks. 2006: 568-577.

[4] REITER M K. A secure group membership protocol[J]. IEEE Trans on Software Engineering, 1996, 22(1): 31-42.

[5] CORREIA M, NEVES N F, VERSSIMO P. How to tolerate half less one Byzantine nodes in practical distributed systems[C] // Proc of the 23rd IEEE Symposium on Reliable Distributed Systems. 2004: 174-183.

[6] LUNG L C, CORREIA M, NEVES N F, *et al.* A simple intrusion-tolerant reliable multicast protocol using the TTCB [EB/OL]. (2003). <http://www.di.fc.ul.pt/~nuno/PAPERS/SBRC03.pdf>.

[7] CORREIA M, VERSSIMO P, NEVES N F. The design of a COTS real-time distributed security kernel[C] // Proc of the 4th European Dependable Computing Conference. 2002: 234-252.

[8] MENEZES A J, OORSCHOT P C, VANSTONE S A. Handbook of applied cryptography[M]. [S. l.] : CRC Press, 1996.