第七章

分布式系统

分布式系统是这样一种系统,在这种系统中,您甚至不知道其存在的计算机发生故障可能会导致您自己的计算机无法使用。

– 莱斯利·兰波特 [1123]

名字里有什么?我们称之为玫瑰的任何其他名字都会闻起来很甜 威廉·莎士比亚

7.1 简介

我们需要的不仅仅是身份验证、访问控制和密码学来构建任何规模的健壮的分布式系统。有些事情需要快速发生,或者以正确的顺序发生,一旦我们拥有具有复杂弹性安排的超大规模数据中心,对于几台机器来说处理起来微不足道的事情就会变成一件大事。大家一定注意到了,当你通过在线服务提供商更新你的地址簿时,更新可能会在另一台设备上出现一秒钟,或者可能只有几个小时。

在过去的 50 年里,随着我们构建从电话系统和支付网络到 Internet 本身的各种事物,我们学到了很多有关并发、故障恢复和命名等问题的知识。我们有扎实的理论,也有很多来之不易的经验。这些问题是稳健安全系统设计的核心,但通常处理得相当糟糕。我已经描述了作为并发故障出现的对协议的攻击。如果我们复制数据以使系统具有容错能力,那么我们可能会增加数据被盗的风险。最后,命名可能是一个棘手的问题。人和物与帐户、会话、文档、文件、指针、密钥和其他命名方式之间存在着复杂的交互。许多组织正在尝试构建更大、更扁平的命名空间 无论是使用身份证来跟踪公民还是使用设备 ID 来跟踪对象 但我们实际上可以做的事情是有限的。大数据意味着要处理大量的标识符,其中许多标识符是模棱两可的,甚至是不断变化的,而且很多事情都可能出错。

7.2 并发

如果进程可以同时运行,则称为并发,这对性能至关重要;现代计算机有许多内核,一次运行许多程序,通常供许多用户使用。然而,并发很难做到稳健,尤其是当进程可以对相同的数据进行操作时。进程可能会使用旧数据;他们可以进行不一致的更新;更新的顺序可能重要也可能不重要;系统可能会死锁;不同系统中的数据可能永远不会收敛到一致的值;当让事情按正确的顺序发生很重要,甚至知道确切的时间很重要时,这可能比您想象的要棘手。这些问题在整个堆栈中上下波动。

由于多种原因,系统变得越来越并发。首先是规模:谷歌可能一开始只有四台机器,但他们的车队在 2011 年超过了 100 万台。其次是设备的复杂性;一辆豪华汽车现在可以包含数十个到数百个不同的处理器。您的笔记本电脑和手机也是如此。在每个 CPU 的深处,指令是并行执行的,这种复杂性导致了我们在访问控制一章中讨论的 Spectre 攻击。除此之外,Xen等虚拟化技术是构建现代云服务的平台,它们可以将服务器中的少数几个真实CPU变成数百个甚至数千个虚拟CPU。然后是交互的复杂性:上升到应用程序层,预订租车等日常交易可能会调用其他系统来检查您的信用卡、您的信用评级机构评分、您的保险索赔历史等等,而这些系统反过来可能取决于他人。

并发系统编程很难,标准的教科书示例来自操作系统内部和性能测量领域。计算机科学家被教导阿姆达尔定律:如果可以并行化的比例是 p,s 是额外资源的加速比,那么整体加速比是 (1p+p/s)1。因此,如果您的程序的四分之三可以并行化,但剩余的四分之一不能并行化,那么您可以获得的最大加速是四倍;如果你投入八个核心,实际加速不是三倍1。但是现实世界中的并发控制也是一个安全问题。与访问控制一样,需要防止用户无意或有意地相互干扰。并发问题可能发生在系统的许多层面,从硬件一直到业务逻辑。在下文中,我提供了一些具体的例子;它们绝不是详尽无遗的。

7.2.1 使用旧数据与付费传播状态

我已经描述了两种并发问题:对协议的重放攻击,攻击者设法传递 o 过时的凭据;和竞争条件,其中两个程序可以竞争更新某些安全状态。

例如,我提到了来自 Unix 的“mkdir”漏洞,其中分两个阶段执行的特权指令可能会通过重命名其作用的对象而在中途受到攻击。另一个例子可以追溯到

1(1 $\frac{3}{4}$ + $\frac{3}{4.8}$)1 =(0.25 + 0.09375)1=(0.34375)1=2.909

1960 年代,在最早的多用户操作系统之一 IBM 的 OS/360 中,尝试打开文件会导致读取文件并检查其权限;如果用户被授权访问它,它会被再次阅读。用户可以安排事情,以便文件在 [1129] 之间被更改。

这些是检查时间到使用时间 (TOCTTOU) 攻击的示例。
我们有系统的方法在文件系统中发现此类攻击 [251],但攻击仍然会在较低级别(例如虚拟化环境中的系统调用)和较高级别(例如业务逻辑)中出现。阻止它们并不总是经济的,因为传播安全状态的变化可能代价高昂。

一个很好的案例研究是信用卡欺诈。自 1970 年代信用卡和借记卡流行以来,银行业不得不管理热卡列表(无论是被盗还是被滥用),并且随着信用卡网络走向国际,这个问题在 1980 年代变得越来越严重。不可能在每个商户终端上都保留一份完整的热卡列表,因为我们必须立即向数千万台设备广播所有挂失报告,即使我们试图与发卡银行核实所有交易卡,我们将无法在没有网络的地方使用卡(例如在偏远的村庄和飞机上),并且我们会在其他地方强加不可接受的成本和延误。相反,有多个级别的替代处理,利用了大多数支付是本地支付或低价值支付或两者兼而有之的事实。

允许商户终端在线处理一定限额(下限)内的交易;较大的交易需要与商户银行进行在线验证,这将了解所有本地热卡以及被积极滥用的外国卡;超过另一个限制,它可能会将交易转至具有合理最新国际列表的网络,例如 VISA;而最大的交易需要参考发卡银行。实际上,在使用前立即检查的唯一交易是本地或大型交易。

经验告诉我们,更集中的方法可以更好地应对不良终端。世界上大约一半的 ATM 交易使用一种服务,当有人试图在 ATM 上使用被盗的卡或猜错 PIN 码时,该服务会从订阅银行收到警报。 FICO观察到,犯罪分子将一把偷来的卡拿到提款机前,一张一张地试;他们保留了一份全球 40 台 ATM 的清单,这些 ATM 最近被用于企图欺诈,订阅他们服务的银行拒绝在这些机器上进行所有交易 这些机器可能在半小时内无法被这些银行的卡使用。大多数小偷不明白这一点,只是把它们扔掉了。

直到大约 2010 年,支付卡网络拥有最大的系统来管理安全状态的全球传播,他们的经验告诉我们,在全球范围内快速撤销受损凭证的成本很高。

其他地方也吸取了教训;例如,美国国防部在 1999 年至 2005 年期间向军事人员颁发了 1600 万张证书,到那时它每天必须向所有安全服务器下载 1000 万张已撤销的证书,有些系统需要半小时才能完成当他们被解雇时 [1299]。

传播安全状态的成本会导致中心化。谷歌、Facebook 和微软等大型服务公司无论如何都必须为数十亿用户维护凭证,因此它们将登录作为服务提供给其他网站。

其他公司,例如认证机构,也提供在线证书。
但是,尽管集中化可以降低成本,但中央服务的妥协可能会造成破坏。例如,2011 年,使用伊朗 IP 地址的黑客入侵了荷兰认证机构 Diginotar。 7 月 9 日,他们生成假证书并对伊朗激进分子的 gmail 进行中间人攻击。Diginotar 于 19 日注意到证书被错误颁发,但只是召集了其审核员。黑客攻击于 29 日公开,Google 的反应是在 9 月 3 日从 Chrome 中删除所有 Diginotar 证书,并让 Mozilla 也这样做。这立即导致公司倒闭,并且由于各部争先恐后地从其他供应商那里获得其网络服务证书 [471],荷兰公共服务在线多日无法使用。

7.2.2 锁定以防止不一致的更新

当人们同时处理一份文档时,他们可能会使用版本控制系统来确保在任何时候只有一个人对文档的任何给定部分具有写入权限,或者至少警告争用并标记任何不一致的编辑。锁定是一种管理资源争用（例如文件系统）并降低更新冲突可能性的通用方法。另一种方法是回调;服务器可能会保留所有依赖于它的客户端的列表以确保安全状态,并在状态更改时通知它们。

信用卡再次提供了如何将其应用于安全性的示例。如果我拥有一家酒店,并且客户在入住时出示了信用卡,我会向信用卡公司索要预授权,这会记录我将在不久的将来进行扣款;我可能会提出“最高 500 美元”的索赔。这是通过分离授权和结算系统来实现的。处理故障模式可能很棘手。如果第二天卡被注销,我的银行可以打电话给我,让我联系警察,或者让她支付现金2。这是发布-注册-通知模型的一个例子,说明如何在分布式系统中进行健壮的授权（在 [152] 中有更一般的描述）。

不过,回调机制并未提供通用解决方案。凭证发行者可能不想运行回调服务,并且客户可能出于隐私原因反对向发行者告知她的所有来往。

考虑护照作为另一个例子。在许多国家,许多交易都需要政府提供的身份证件,但政府不会提供任何担保,如果政府对每次出示身份证件都进行记录,大多数公民会反对。事实上,印度政府要求在越来越多的交易中使用 Aadhar 生物识别 ID 系统的要求经常遭到反对,其中之一是在所有重要交易中检查公民的指纹或虹膜代码,从而创建对他们所做的所有地方的审计跟踪业务,官方人员和任何想贿赂他们的人都可以使用。

2我的银行可能会也可能不会为我提供这笔钱的担保;这完全取决于我与它签订的合同类型。当骗子想出如何冒充商店并取消授权以便一张卡可以用于多次大额购买时,也有一段时间的攻击。发卡银行可能需要一到三天的时间才能将警报传播到商家的银行。深入研究这一切本身就是一本书的章节！

对发行人产生某种义务的凭据（例如信用卡)和其他凭据（例如护照)之间存在一般区别。不同之处在于凭证的使用是否改变了重要状态,除了可能添加到日志文件或其他监视系统之外。

这与更新的顺序是否重要有关。

7.2.3 更新顺序

如果两笔交易到达政府的银行账户 比如贷方 500,000 美元和借方 400,000 美元 那么应用它们的顺序可能并不重要。但如果他们到达我的银行账户,订单将对结果产生巨大影响!事实上,决定事务应用顺序的问题没有干净的解决方案。它与如何并行化计算的问题密切相关,构建高效分布式系统的大部分艺术在于安排事务,以便进程要么简单地顺序进行,要么完全并行化。

传统的银行算法是在一夜之间对交易进行批处理,并在应用所有借记之前应用每个账户的所有贷记。来自 ATM 和支票分拣机等设备的输入在隔夜核对之前首先被分批放入日志中。退回的付款必须撤回 在 ATM 和借记卡交易中,现金已经用完,您最终可能会遇到客户未经授权就借钱的情况。在实践中,失败的支付链终止。近年来,一个又一个国家引入了实时全额结算 (RTGS) 系统,其中交易按到达顺序记账。有几个微妙的缺点。首先,在许多机构中,面向零售客户的实时系统是覆盖在一个平台上的,该平台仍然可以通过夜间更新运行。

其次,结果可能取决于交易顺序,这可能取决于人为、系统和网络的变幻莫测,当金融机构之间进行许多非常大的支付时,这可能是一个问题。信用卡采用混合策略,在结算时实时运行信用额度,就像在老式支票账户中一样。

在 2010 年代后期,对加密货币的兴趣浪潮让一些企业家相信区块链可能会解决更新不一致的问题,简化供应链管理等应用程序。对于大多数应用程序,能源成本排除了基于工作量证明的区块链;但其他类型的只能追加的公共分类账是否会找到一个杀手级应用程序?我们将不得不拭目以待。同时,加密货币社区广泛使用经常让人想起支票账户方法的 o-chain 机制:断开连接的应用程序提出临时更新,这些更新稍后会被协调并应用于主链。经验表明,在一般情况下并没有什么神奇的解决方案,或许除了拥有少数非常擅长技术的超大型银行之外。我们将在关于银行业务的章节中进一步讨论这个问题。

在其他系统中,交易到达的顺序不那么重要。护照就是一个很好的例子。护照签发者只担心他们的创建和到期日期,而不是签证盖章的顺序

在他们 3。

7.2.4 死锁

另一个问题是死锁,两个系统都在等待对方先行动。 Edsger Dijkstra 通过用餐哲学家问题著名地解释了这个问题及其可能的解决方案。许多哲学家围坐在一张桌子旁,每人之间夹着一根筷子。哲学家只有在能拿起两边的筷子时才能吃饭。因此,如果他们所有人都试图同时吃东西,并且每个人都拿起右边的筷子,他们就会被卡住 [560]。

当您有跨系统分布的多个锁层次结构时,这会变得非常复杂,其中一些会失败(并且失败可能意味着锁不可靠)[151]。僵局不仅仅与技术有关; “第 22 条军规”一词已流行用于描述官僚流程中的僵局4。在流程是手动的情况下,可能会发现一些软糖来绕过陷阱,但当一切都变成软件时,此选项可能不再可用。

在一个众所周知的商业问题中 表格之战 一家公司发出一份附有自己合同条款的订单,另一家公司根据自己的条款接受订单,并且在没有任何进一步协议的情况下进行交易。

在过去,事情可能只有在出现问题并且公司最终诉诸法庭时才能得到解决;即便如此,一家公司的条款可能会指定美国法院,而另一家公司的条款可能会指定英国法院。随着交易变得更加电子化,赢家通常是能够迫使输家使用其网站进行交易并因此接受其条款和条件的公司。公司越来越多地试图确保事情以对他们有利的方式失败。由此产生的责任游戏可能对安全和安全产生相当负面的结果;我们将在经济学章节中进一步讨论它们。

7.2.5 非收敛状态

在设计更新分布式系统状态的协议时,“母亲和苹果派”是 ACID 事务应该是原子的、一致的、隔离的和持久的。如果您“全部执行或根本不执行”,则事务是原子的 这使得在失败后更容易恢复。如果保留了一些不变量,例如账簿必须仍然平衡,那么它就是一致的。这在银行系统中很常见,并且通过坚持每笔交易的贷方和借方总和为零来实现(我将在有关银行业务和簿记的章节中对此进行更多讨论)。如果事务是可序列化的,那么它们就是隔离的;如果事务一旦完成就无法撤消,那么它们就是持久的。

这些属性可能太多,也可能不够,或者两者兼而有之。一方面,它们中的每一个都可能以许多晦涩的方式失败或受到攻击;在

3许多阿拉伯国家不会让你的护照上有以色列的印章,但大多数纯粹的身份识别系统基本上是无国籍的。

4约瑟夫·海勒 (Joseph Heller) 1961 年的同名小说描述了第二次世界大战军事官僚机构中不一致和疯狂规则的多个实例。

另一方面,将系统设计为收敛的通常就足够了。这意味着,如果交易量尾随 o ,那么最终整个 [1353] 都会有一致的状态。通常使用时间戳和版本号等语义技巧来实现收敛;在事务被附加到文件而不是被覆盖的情况下,这通常就足够了。

在现实生活中,您还需要一些方法来应对出现问题且无法完全恢复的情况。安全或审计经理的生活可能是一场与熵的持续斗争:明显的赤字(和盈余)总是出现,有时根本无法解释。例如,不同的国家系统对于银行交易记录中的哪些字段是强制性的或可选的有不同的想法,因此支付网关通常必须猜测数据才能使事情正常进行。有时他们猜错了;有时人们会看到并利用直到很久以后(如果有的话)才被理解的漏洞。

最后,通过添加一个校正因子并设定一个目标使其保持在某个年度阈值以下,事情可能会变得模糊。

持久性是事务处理中争论的主题。网络钓鱼和键盘记录攻击的出现意味着一小部分银行账户随时会被不法分子控制;钱从他们那里转移,也通过他们转移。当检测到帐户泄露时,银行会冻结它,并可能撤销最近从中进行的付款。网络钓鱼者自然会尝试在不进行交易逆转的机构或司法管辖区内转移资金,或者充其量只能缓慢而勉强地进行 [75]。这一方面在可恢复性和支付系统的弹性与另一方面交易的持久性和最终性之间造成了紧张关系5。

7.2.6 安全时间

安全工程师特别感兴趣的最后一个并发问题是提供准确的时间。由于可以通过引入时钟错误来攻击 Kerberos 等身份验证协议,因此仅仅信任随机的外部时间源是不够的。一种可能性是灰姑娘攻击:如果防火墙等安全关键程序拥有带时间锁的许可证,攻击者可能会把你的时钟调快“并导致你的防火墙变成南瓜”。鉴于物联网设备的传播可能对安全至关重要,并且以人们知之甚少的方式使用时间,现在人们担心可能发生大规模的服务拒绝攻击。时间比看起来难得多:即使你有原子钟,也无法预测闰秒,但需要以某种方式广播;有些分钟有 61 秒甚至 62 秒;奇数时间效应可能是一个安全问题6;世界上大部分地区都不使用公历。

5这个问题可以追溯到几个世纪前,关于善意行事的人是否可以获得对赃物或被盗资金的良好所有权的法律繁多。 1882 年的《汇票法》为善意购买汇票的人提供了良好的权利,即使这些汇票被盗。类似的东西曾经用来存放在公开市场上购买的赃物,但最终被废除了。在电子支付的情况下,银行充当卡特尔,通过卡网络规则和通过支付服务指令游说欧洲机构来更快地完成支付。至于比特币的情况,它仍在不断变化;参见第 20.7.5 节。

6有些ATM在Y2K之后几天不查客户余额,导致unau
一旦消息传开,就会透支透支

无论如何,有几种可能的方法来提供安全时间。
你可以给每台电脑一个无线电时钟,而且你的智能手机确实有 GPS 但它可能会被路过的卡车司机干扰。您可以放弃绝对时间,而改用 Lamport 时间,其中您只关心事件 A 是否发生在事件 B 之前,而不是它是什么日期 [1122]。

出于稳健性原因,Google 不在其内部证书中使用时间,而是使用与撤销机制相结合的序列号范围 [23]。

在许多应用程序中,您最终可能会使用网络时间协议 (NTP)。这具有中等程度的保护,具有时钟投票和时间服务器的身份验证,并且对于许多用途来说足够可靠。但是,您仍然需要保重。例如,Netgear 将他们的家用路由器硬连接到威斯康星大学麦迪逊分校的 NTP 服务器,该服务器每秒被数十万个数据包淹没; Netgear 最终不得不向他们支付 375,000 美元,以维持三年的定时服务。

不久之后,D-Link 又犯了同样的错误 [445]。其次,从 2016 年开始,出现了使用 NTP 服务器作为力量倍增器的拒绝服务攻击;数以百万计的服务器被证明是可滥用的,因此许多 ISP 甚至 IXP 开始阻止它们。因此,如果您计划在公司网络外部部署大量依赖 NTP 的设备,您最好仔细考虑您要信任哪些服务器,并注意 CERT [1797] 的最新指南。

7.3 容错和故障恢复

故障恢复通常是安全工程中最重要的方面,但也是最容易被忽视的方面之一。多年来,大多数关于计算机安全的研究论文都涉及机密性,其余大部分涉及真实性和完整性;可用性几乎被忽略了。然而,现代信息业务(无论是银行还是搜索引擎)的实际支出却恰恰相反。在可用性和恢复机制(例如多个处理站点和冗余网络)上的花费远远多于在完整性机制(例如代码审查和内部审计)上的花费,而这又远远超过在加密上的花费。在阅读本书时,您会发现许多其他应用程序,从防盗警报到电子战,再到保护公司免受 DDoS 攻击,从根本上讲都与可用性有关。

容错和故障恢复往往是安全工程师工作的核心。

经典的容错通常是建立在冗余的基础上,使用日志、锁等机制加固的,当要经受住对这些机制的恶意攻击时,就变得非常复杂。容错以多种方式与安全相互作用:故障模型、弹性的性质、用于提供它的冗余位置,以及对拒绝服务攻击的防御。我将使用以下定义:故障可能导致错误,这是一种不正确的状态;这可能会导致失败,这是与系统指定行为的偏差。我们构建到系统中以容忍故障并从故障中恢复的弹性将包含许多组件,例如故障检测、错误恢复以及必要时的故障恢复。意义

平均故障前时间 (MTBF) 和平均修复时间 (MTTR) 应该是显而易见的。

7.3.1 故障模型

为了决定我们需要什么样的弹性,我们必须知道会发生什么样的攻击。其中大部分将来自对我们系统操作环境特定威胁的分析,但一些一般性问题值得一提。

7.3.1.1 拜占庭失败

首先,我们关注的失败可能是正常的,也可能是恶意的,我们通常将后者建模为拜占庭。拜占庭失败的灵感来自于这样一个想法,即有 n 个将军保卫拜占庭,其中 t 个将军被进攻的土耳其人贿赂以造成尽可能多的混乱。将军可以通过信使传递口头消息,信使是值得信赖的,所以每个将军都可以与其他将军交换机密和真实的通信(我们可以想象他们在每条消息上加密和计算一个 MAC)。最多可以容忍多少叛徒?

关键的观察是,如果我们只有三个将军,比如 Anthony、Basil 和 Charalampos,而 Anthony 是叛徒,那么他可以告诉 Basil “让我们进攻”和 Charalampos “让我们撤退”。 Basil 现在可以对 Charalampos 说“Anthony 说让我们进攻”,但这并没有让 Charalampos 断定 Anthony 是叛徒。它很可能就是巴兹尔;安东尼本可以对他们俩说“让我们撤退”,但巴兹尔在说“安东尼说让我们进攻”时撒了谎。

这一绝妙的见解归功于 Leslie Lamport、Robert Shostak 和 Marshall Pease,他们证明了当且仅当 n 3t+1 [1124] 时问题有解。
当然,如果将军们能够在他们的消息上签名,那么就没有将军敢对两个不同的同事说不同的话了 。这特别说明了数字签名和一般端到端安全机制的强大功能。现在有大量关于拜占庭容错的文献 能够承受这种故障的系统的详细设计;例如,参见 Miguel Castro 和 Barbara Liskov [394] 的算法。

另一个教训是,如果一个失败的组件(或可能被对手诱导失败)给出了错误的答案,而不是仅仅没有答案,那么使用它来构建一个有弹性的系统就会困难得多。如果失败的组件只是停止,或者如果它们至少可以被快速识别并列入黑名单,这将很有用。

7.3.1.2 与容错的交互

所以我们可以通过多种方式来限制故障率。两个最明显的方法是使用冗余和故障停止过程。后者处理纠错信息和数据,并在检测到不一致时停止;例如,如果在处理任务后检测到余额不足情况,银行交易处理通常会停止。两者可以结合;汽车和飞机中某些安全关键功能中使用的处理器通常

有两个或多个核心。 1970 年代,在航天飞机项目的推动下,容错多处理器 (FTMP) 有了开创性的工作;这探讨了哪些组件应该是冗余的,以及围绕错误检测发生的位置以及一切同步的紧密程度的相关设计权衡[920]。此类研究最终推动了用于各种潜艇和航天器的容错处理器的设计,以及波音和空客使用的架构。FTMP 的想法也被 Tan dem 和 Stratus 商业化,后者销售用于支付处理的机器。 Stratus 有两个磁盘、两条总线,甚至还有两个 CPU,如果检测到错误,每个 CPU 都会停止;故障停止 CPU 是通过在同一张卡上安装两个 CPU 芯片并比较它们的输出而构建的。如果他们不同意,输出就会开路。替换卡将邮寄到;你会把它带到机房,注意到卡 5 有一个闪烁的红灯,把它拔出来换上新的 这时机器每秒处理几十笔交易。如今,大型服务公司的数据中心有更复杂的协议来确保如果一台机器发生故障,另一台机器会接管;如果一个机架发生故障,另一个机架将接管;即使一个数据中心发生故障,它的工作负载也会很快由其他数据中心恢复。谷歌是开发相关软件堆栈的领导者,它在 2000 年代初发现,使用商品 PC 和智能软件构建大型系统比从专业供应商处购买更大的服务器要便宜得多。

虽然冗余可以使系统更具弹性,但它也有成本。首先,我们必须处理更复杂的软件堆栈和工具链。Banks 最终放弃了 Stratus,因为他们发现 Stratus 总体上不如传统大型机可靠:虽然硬件故障导致的停机时间更少,但这并不能弥补不熟悉的开发环境导致的额外软件故障。其次,如果我有多个站点有备份数据,那么如果其中任何一个遭到破坏,机密性可能会失败 7;如果我有一些我有责任销毁的数据,那么从多个备份磁带中清除它可能会让人头疼。在冗余云服务之上的容器中开发软件的现代问题与其说是编程语言,不如说是通过数据中心的妥协;这是因为开发人员不熟悉云服务提供商的访问控制工具,并且经常让敏感数据在全球范围内可读。

粗心的人还有其他陷阱。在我被称为专家的一个案例中,我的客户在商店使用信用卡时被逮捕,被指控拥有伪造的卡,并被警察殴打。他坚信这张卡是真的。很久以后,我们让 VISA 检查了这张卡,VISA 确认它确实是真的。发生了什么,以及我们可以重建它,是这样的。信用卡在磁条上有两种类型的冗余 通过使用异或将磁道上的所有字节组合在一起获得的简单校验和,以及我们将在后面的 12.5.1 节中详细描述的加密校验和。前者用于检测错误,后者用于检测伪造。似乎在这种特殊情况下,商家的读卡器未对准,导致偶数位错误在简单校验和中偶然相互抵消,同时导致

7或者您的数据中心之间的通信被窃听;我们在 2.1 节讨论过 GCHQ 是如何对谷歌这样做的。

加密校验和失败。结果是一场虚惊,严重扰乱了我客户的生活。

冗余在机械系统中很难处理。例如,训练飞行员处理多引擎飞机包括对他们进行引擎故障程序的训练,首先是在模拟器中,然后是在真正的飞机上与教练一起进行。事实上,新手飞行员在多引擎飞机上比在单引擎飞机上更有可能死于发动机故障;在最近的场地着陆对他们来说比应对突然的不对称推力更危险。仪器故障也是如此;如果在压力下你依赖坏掉的那个,那么在驾驶舱里放三个人造地平线也无济于事。飞机比许多现代信息系统简单得多 但当飞行员未能管理本应确保他们安全的冗余时,仍然会发生空难。

还有一些复杂的故障,例如两架波音 737 Max 飞机因单个传感器故障而坠毁,当飞机有两个但软件无法读取它们时,飞行员没有接受过如何诊断问题和管理后果。很多时候,系统设计者会设置多种保护机制,但没有仔细考虑后果。许多其他安全故障是可用性故障,这同样适用于安全性,正如我们在第 3 章中讨论的那样;冗余不是糟糕设计的解毒剂。

7.3.2 弹性有什么用?

在系统中引入冗余或其他弹性机制时,我们需要了解它们的用途以及各个参与者面临的激励措施。因此,复原力是本地的还是跨越地理或组织边界很重要。

在第一种情况下,复制可以成为服务器的内部功能,以使其更值得信赖。我已经提到了 1980 年代的系统,例如 Stratus 和 Tandem;然后我们在组件级别复制了标准硬件,例如廉价磁盘冗余阵列 (RAID)。自 20 世纪 90 年代末以来,大量投资用于开发机架级系统,让多台廉价 PC 完成昂贵服务器的工作,并采用机制确保发生故障的单个服务器的工作负载将由另一台迅速接管,实际上是机架失败的也可以在热备份上恢复。这些现在是云服务架构的标准组件:任何运营数十万台服务器的公司都会遇到如此多的故障,以至于恢复必须在很大程度上实现自动化。

但事情往往要复杂得多。一项服务可能不得不假设它的某些客户正试图欺骗它,并且还可能不得不依赖许多服务,但没有一个是完全准确的。在开设银行账户或签发护照时,我们可能希望对照从选民名册到信用咨询机构再到驾照数据库的服务进行核对,而结果通常可能不一致。信任决策可能涉及复杂的逻辑,与电子战中使用的试图找出哪些输入被干扰的系统并不完全不同。 (我将在有关电子和信息战的章节中进一步讨论这些内容。)

不信任的方向对协议设计有影响。面对多个不可信客户端的服务器和依赖多个可能无能、不可用或恶意的服务器的客户端都希望控制协议中的消息流，以遏制服务拒绝的影响。很难为每个人都不可靠且所有人都相互怀疑的现实世界设计系统。

有时重点是安全可更新性。银行卡就是一个明显的例子：银行可以不时地通过邮寄更新版本的卡来升级安全性，无论是从磁条升级到芯片还是从廉价芯片升级到更复杂的芯片；它可以通过不定期向受影响的客户邮寄卡片来从妥协中恢复过来。付费电视和手机有些相似。

7.3.3 冗余在什么级别？

可以使系统在多个级别上对错误、攻击和设备故障具有弹性。与访问控制一样，随着我们在系统中上升到更高层，这些变得越来越复杂且越来越不可靠。

一些计算机在硬件层面已经构建了冗余，例如我之前提到的 Stratus 系统和 RAID 磁盘。但是简单的复制不能防御恶意软件，也不能防御利用有缺陷的软件的入侵者。

在下一层，存在进程组冗余。在这里，我们可以在不同位置的多个服务器上运行一个系统的多个副本，并比较它们的输出。这可以阻止对手获得对机器的物理访问并破坏它的攻击类型，无论是通过机械破坏还是通过插入未经授权的软件。它无法抵御授权用户的攻击或不良授权软件的破坏，这些软件可以简单地命令删除关键文件。

下一个级别是备份，我们通常会定期获取系统副本（检查点）。副本通常保存在无法覆盖的介质上，例如写保护磁带或带有特殊软件的光盘。

我们还可能保留检查点之间应用的所有交易的日志。
无论细节如何，备份和恢复机制不仅使我们能够从物理资产破坏中恢复，它们还确保如果我们确实在逻辑层面受到攻击，我们有恢复的希望。 1980 年代的经典示例是一个定时炸弹，它会在特定日期删除客户数据库；自加密货币问世以来，时尚一直是

勒索软件。

具有关键服务需求的企业（例如银行和零售商）多年来一直拥有备份数据中心。这个想法是，如果主中心出现故障，该服务将故障转移到第二个设施。维护此类设施占用了典型银行的大部分信息安全预算。

备份与回退不同。回退系统通常是功能较弱的系统，当主系统不可用时，处理会恢复到该系统。

一个例子是使用手动压印机采集信用卡

当电子终端出现故障时,从卡上进行交易。回退系统是应用层冗余的一个例子 我们可以把它放在最高层。

重要的是要认识到这些是不同的机制,它们做不同的事情。冗余磁盘无法防止恶意程序员删除您所有的帐户文件,如果他不只是删除文件而是编写缓慢插入越来越多错误的代码,备份也不会阻止他。两者都不会提供太多保护以防止对数据机密性的攻击。另一方面,如果您的数据处理中心烧毁,世界上最好的加密也无济于事。现实世界的恢复计划和机制涉及上述所有方面的混合。

我之前所说的关于冗余的困难以及正确计划和培训的绝对必要性,完全适用于系统备份。当我在 1980 年代在银行业工作时,我们估计我们可能会在我们的主处理中心被摧毁后的一个小时左右让我们的备份系统工作,但测试受到限制,因为我们不想冒险在工作时间处理:我们将在每年的一个星期六在我们的备份数据中心恢复主要生产系统。到 1990 年代初期,英国超市乐购 (Tesco) 已经进行了现场演习:他们每年在不通知操作员的情况下拔下主处理中心的插头,以确保备份在 40 秒内启动。到 2011 年,Netflix 开发了“混沌猴子” 随机破坏一台机器、一个机架,甚至整个数据中心的系统,以不断测试弹性。到 2019 年,大型服务公司的规模已经达到不需要这个的程度。如果您在 30 个数据中心拥有 300 万台机器,那么您将不断丢失机器,频繁丢失机架,并且经常丢失整个数据中心,以至于您必须设计一些东西才能继续运行。所以现在,你只需付钱,云服务提供商就会为你操心很多细节。但你需要真正了解亚马逊、谷歌或微软可以为你处理什么样的故障,以及你必须自己处理什么。主要供应商的标准服务水平协议允许他们每月中断你的服务好几个小时,如果你使用较小的云服务(甚至是政府云),它会有容量限制,你必须仔细考虑.

值得尝试弄清楚您依赖哪些服务是在您的直接供应链之外的。例如,英国在 2001 年发生了油罐车司机罢工,一些医院因人手短缺而不得不关闭,这本来是不可能发生的。政府已将汽油口粮分配给医生和护士,但没有分配给学校教师。所以学校关闭了,护士们不得不呆在家里照顾他们的孩子,这也导致医院也关闭了。

这帮助罢工者击败了首相托尼·布莱尔:他放弃了稳步提高燃油税的标志性环境政策。随着我们越来越依赖彼此,应急计划变得越来越困难。

8如今,真正严重的勒索软件运营商会入侵您的系统,暗中添加文件加密并等待他们突袭 – 因此他们不仅劫持您当前的数据,还劫持数周的备份

7.3.4 拒绝服务攻击

我们希望安全服务具有容错能力的原因之一是降低拒绝服务攻击的吸引力和/或有效性。此类攻击通常用作更大计划的一部分。例如,有人可能会关闭安全服务器以强制其他服务器使用缓存的凭据副本,或者淹没 Web 服务器以使其暂时离线,然后让另一台机器为受害者尝试下载的页面提供服务。

防止服务拒绝的有力防御措施是防止对手发起选择性攻击。如果委托人是匿名的 比如负载均衡器后面有几个等效的服务,而对手不知道要攻击哪一个 那么他可能是无效的。我将在防盗警报器和电子战的背景下进一步讨论这个问题。

如果这是不可能的,并且对手知道在哪里攻击,那么有些类型的服务拒绝攻击可以通过冗余和弹性机制来阻止,而其他类型则不能。例如,TCP/IP 协议几乎没有让主机保护自己免受网络泛滥的有效机制,而网络泛滥有多种形式。防御这种攻击往往涉及将您的网站转移到具有专业数据包清洗硬件的更强大的托管服务 - 或者追踪和逮捕肇事者。

分布式拒绝服务 (DDoS) 攻击在 1996 年被用于使纽约 ISP Panix 瘫痪数日后引起了公众的注意。

在 1990 年代后期,脚本小子偶尔会使用它们来关闭聊天服务器。 2001 年,我在本书的第一版中顺便提到了它们。
在接下来的三年里,勒索者开始使用它们;他们会组装一个僵尸网络,一个由受感染的 PC 组成的网络,它会用数据包流量淹没目标网络服务器,直到它的所有者付钱让他们停止。典型的目标是在线博彩公司,通常需要 10,000 至 50,000 美元的金额才能让他们置之不理,而典型的博彩公司在第一次发生这种情况时就付清了。

当攻击持续存在时,第一个解决方案是复制:运营商将他们的网站转移到 Akamai 等托管服务,Akamai 的服务器数量如此之多(而且离客户如此之近),以至于他们可以对普通僵尸网络可能向他们抛出的任何东西不屑一顾。最终,当博彩公司开会并同意不再支付勒索钱时,勒索问题得到了解决,乌克兰警方被迫逮捕了肇事团伙。

到 2018 年,我们又回到了原点,大约有 50 名坏人在运营 DDoS 即服务,主要是针对想要摧毁对手的 teamspeak 服务器的游戏玩家。这些服务作为“引导程序”在网上出售,可以引导你的对手退出游戏;几美元就能获得大约 100Gbit/sec 的流量。服务运营商还更委婉地称它们为“压力源” 您可以使用它们来测试您自己网站的稳健性。这并没有骗过任何人,就在 2018 年圣诞节前夕,FBI 关闭了其中 15 个网站,逮捕了他们的一些运营商,并导致 DDoS 流量在几个月内显着下降 [1445]。

最后,如果存在更易受攻击的回退系统,一种常见的技术是使用拒绝服务攻击迫使受害者进入回退模式。典型的例子是支付卡。智能卡通常比

磁条卡,但由于静电和触点磨损,每年可能有 1% 的卡出现故障。此外,一些游客仍然使用磁条卡。

所以大多数卡支付系统仍然有使用磁条的后备模式。一种简单的攻击是使用假终端,或将窃听器插入到真正终端的电缆中,以捕获卡的详细信息,然后将它们写入带有坏芯片的卡的磁条中。

7.4 命名

命名对于普通的分布式系统来说是一个次要的问题,但在安全工程中却变得异常困难。在 20 世纪 90 年代的互联网繁荣时期,当 SSL 被发明并且我们开始建立公钥认证机构时,我们遇到了在证书上放什么名字的问题。

简单地写着“允许名为 Ross Anderson 的人管理机器 X”的证书几乎没有用处。我曾经是我所知道的唯一罗斯·安德森;但是当第一个搜索引擎出现时,我发现我们有几十个。我也以不同的名字为数十种不同的系统所熟知。名称存在于上下文中,在安全系统中命名主体变得越来越重要和困难。

从概念上讲,命名空间可以是分层的或平面的。您可以将我称为“在英国剑桥教授计算机科学的 Ross Anderson”或“rossjanderson@gmail.com 的 Ross Anderson”,甚至是“拥有某某护照号码的 Ross Anderson”。但这些不是一回事,将它们联系起来会导致各种各样的问题。

通常,使用更多名称会增加复杂性。简单地说“这是管理机器 X 的密钥”的公钥证书是不记名令牌,就像金属门钥匙一样;控制该证书私钥的人就是管理员,就像根密码在银行金库的信封中一样。

但是一旦涉及到我的名字,并且我必须出示某种护照或身份证来证明我是谁,系统就会进一步依赖。如果我的护照被泄露,后果可能是深远的,我真的不想让政府有动力以我的名义向其中一名代理人签发假护照。

9/11 之后,政府开始强制企业要求政府在以前认为没有必要的地方签发带照片的身份证件。例如,在英国,您不能再仅使用购票时使用的信用卡登上国内航班;您必须出示护照或驾驶执照 您还需要在分行进行超过 1000 英镑的银行转账、租公寓、聘请律师甚至找工作。

这些措施不仅不方便,而且会在各种系统中引入新的故障模式。

世界正朝着更大、更扁平的名称空间发展的第二个原因:大型服务公司在在线身份验证领域的主导地位日益增强。
你的名字越来越国际化;它是您的 Gmail 或 Hotmail 地址、您的 Twitter 句柄或您的 Facebook 帐户。这些公司不仅受益于技术外部性,我们在

身份验证和业务外部性,我们将在经济学章节中讨论,它们将在一定程度上解决一些命名问题。但我们不能自满,因为还有许多其他问题。因此,了解一代计算机科学研究人员在分布式系统中的命名知识是很有用的。

7.4.1李约瑟命名原则

在 20 世纪的最后 25 年,构建分布式系统的工程师遇到了许多命名问题。用于将名称绑定到地址的基本算法被称为会合:导出名称的委托人在某处公布它,寻求导入和使用它的委托人搜索它。明显的例子包括电话簿和文件系统目录。

构建分布式系统的人们很快意识到命名会很快变得复杂,Needham [1424] 的一篇经典文章中列出了这些教训。以下是他的十项原则。

1、名字的作用是方便分享。这继续存在:我的银行帐号存在是为了与我本周试图从其取款的出纳员共享我上周存入资金的信息。一般情况下,当要共享的数据是可变的时,就需要命名。如果我只想提取与我存入的金额完全相同的金额,那么不记名存款证明就可以了。

相反,名称不需要共享 或链接 数据不会共享;除非我要从该帐户支付电话费,否则无需将我的银行帐号链接到我的电话号码。

2.命名信息可能不会全部放在一个地方,因此解析名称带来了分布式系统的所有一般问题。这非常有效。银行账户和电话号码之间的联系假定它们都将保持稳定。所以每个系统都相互依赖,对一个系统的攻击可以影响另一个系统。许多银行使用双通道授权来打击网络钓鱼 如果你在线订购付款,你会在手机上收到一条短信,说“如果你想向账户 Y 支付 X 美元,请在你的浏览器中输入以下四位数代码 .

标准的攻击是骗子向电话公司声称是您并报告您的手机丢失。所以他们给了他一个新的 SIM 卡,可以用你的电话号码,他用你的钱赚了 o 。

电话公司可以阻止这种情况,但它不太关心身份验证,因为它所损失的只是一些通话时间,其边际成本为零。最新的攻击是使用 Android 恶意软件窃取身份验证码。谷歌可以通过像苹果一样严格锁定 Android 平台来阻止这种情况 但它缺乏这样做的动力。

3. 假设只需要那么多名字是不好的。 IP 地址短缺激发了 IP 版本 6 (IPv6) 的发展,这一问题已得到充分讨论。鲜为人知的是,信用卡行业有史以来最昂贵的升级是从 13 位信用卡号码变为 16 位。发行人原先假设

13 位数字就足够了,但该系统最终涉及数万家银行 许多银行有数十种产品 因此需要一个 6 位数字的银行识别码。一些发行人拥有数百万客户,因此九位数的帐号是常态。还有一个校验位来检测错误。

4. 全球知名品牌给您带来的收益比您想象的要少。比如IPv6中的128位地址,理论上可以让宇宙中的每一个物体都有一个唯一的名字。但是,为了我们开展业务,必须将我端的本地名称解析为该唯一名称,然后再解析为您端的本地名称。在中间调用一个唯一的名称可能不会给我们带来任何好处;如果唯一的命名服务需要时间、成本或偶尔失败(这是肯定会发生的),它甚至可能会成为障碍。事实上,名称服务本身通常必须是一个分布式系统,与我们试图保护的系统具有相同的规模(和安全级别)。因此,我们可以期待本季度不会出现灵丹妙药。添加额外的名称或采用更复杂的名称可能会增加额外的成本和故障模式。

5. 名称意味着承诺,因此要使方案足够灵活以应对组织变化。在英国政府的安全电子邮件密钥管理系统的设计中忽略了这个合理的原则 [115]。

   在那里,委托人的私钥是从他们的电子邮件地址生成的。
   因此,频繁的重组意味着每次都必须重建安全基础设施 而且必须花更多的钱来解决次要问题,例如人们如何访问旧材料。

6. 名称可以兼作访问权证或能力。我们已经在第 2 章和第 3 章中看到了很多这样的例子。一般来说,假设今天的名字不会成为明天的密码或能力是一个坏主意 记住我们在 4.5 节中讨论的 Utrecht 欺诈。例如,挪威过去认为公民的身份证号码是公开的,但它最终在许多应用程序中被用作一种密码,以至于他们不得不让步并将其设为私有。美国社会安全号码 (SSN) 也存在类似问题。因此,国防部创建了一个称为 EDIPI 的替代编号,该编号本应不敏感;但是,果然,人们开始将其用作身份验证器而不是标识符。

   我已经给出了一些例子,说明当一个名字开始被用作密码时,事情是如何出错的。但有时名称和密码的作用是不明确的。为了进入我以前在大学使用的停车场,我必须对着屏障处的麦克风说出我的姓氏和停车证号码。那么如果我说,“Anderson, 123”,其中哪一个是密码?实际上是“安德森”,因为任何人都可以穿过停车场并记下汽车挡风玻璃上停车证上的有效徽章号码。

7. 如果错误的名字很明显,事情就会变得简单得多。在标准的分布式系统中,这使我们能够对缓存采取自由的态度。在支付系统中,只要信用卡号码出现,信用卡号码就会在终端离线时被接受

有效（即最后一位是前十五位的正确校验位)并且它不在热卡列表中。现代芯片卡上的证书为相同的基本概念提供了更高质量的实现;加密和安全打印等身份验证机制可以提供额外的好处,使名称能够抵御欺骗。作为仍然可能出错的一个例子,爱尔兰警方为“Prawo Jazdy”先生创建了 50 多个案卷，因为他们未能支付超过 50 张交通罚单　直到他们意识到这是波兰语的“驾驶执照”[192]。

8. 一致性很难,而且经常被捏造。如果目录被复制,那么您可能会发现自己无法读取或写入,具体取决于可用目录是太多还是太少。命名一致性会以多种方式给企业带来问题,其中最臭名昭著的可能是条形码系统。虽然这在理论上很简单　每个产品都有一个唯一的数字代码　但在实践中,不同的制造商、分销商和零售商在他们的数据库中对条形码附加了相当不同的描述。因此,根据是否插入撇号,搜索“Kellogg　s”的产品将产生截然不同的结果,这可能会导致供应链混乱。解决这个问题的建议可能非常复杂 [914]。

   还有上面讨论的融合问题;即使在理论上,数据也可能在整个系统中不一致。还有时效性的问题,比如产品是否召回过。

9. 不要太聪明。电话号码比计算机地址更可靠。早期的安全消息系统　从 PGP 到政府系统　试图将密钥链接到电子邮件地址,但这些随着人们的工作而改变。 Signal 和 WhatsApp 等更现代的系统改用手机号码。同样,早期在 SET 等协议中用公钥证书替换银行帐号和信用卡号的尝试失败了,尽管在某些移动支付系统中,例如肯尼亚的 M-Pesa,它们已被电话号码替换。 （我将在 21.6 节中进一步讨论公钥基础设施的具体问题。）

10. 有的名字绑定早,有的不绑定;一般来说,如果可以避免的话,尽早绑定是一件坏事。谨慎的程序员通常会避免编码绝对地址或文件名,因为这会使升级或更换机器变得困难。通常最好将其留给配置文件或 DNS 等外部服务。然而,安全系统通常需要稳定且可靠的名称,因为用于最后一刻解决方案的任何第三方服务都可能成为攻击点。因此,设计人员需要注意命名信息的去向、设备如何使用它进行个性化以及它们如何升级　包括安全性可能依赖的服务名称,例如上面第 7.2.6 节中讨论的 NTP 服务.

7.4.2 还有什么问题

Needham 原则是为 1990 年代初期的世界制定的,在这个世界中,命名系统可以在系统所有者方便的时候强加。一旦我们

转向在全球范围内运营的现代基于网络（和相互关联的）服务行业的现实，我们发现还有更多内容需要补充。

到 2000 年代初期，我们了解到没有任何命名系统可以是全球唯一的、去中心化的和对人类有意义的。事实上，这是一个经典的三难困境：你只能拥有其中两个属性（Zooko 的三角形）[37]。过去，工程师们寻求唯一且有意义的命名系统，例如 URL，或者唯一且分散的系统，例如 PGP 中的公钥或在 Android 中用作应用程序名称的自签名证书。人名是有意义的和本地的，但不能扩展到 Internet。我在上面提到过，当第一个搜索引擎出现时，我可以立即找到几十个叫 Ross Anderson 的人，但情况更糟；六个人在我也曾工作过的领域工作，例如软件工程和配电。

像 Facebook 这样的网站的创新是在非常大的范围内展示名称不必是唯一的。我们可以使用社会背景来构建既去中心化又有意义的系统 这正是我们的大脑进化来应对的。每个 Ross Anderson 都有一群不同的朋友，你可以这样区分我们。

我们如何才能理解这一切，并阻止它被用来绊倒人们？
有时详细分析名称的属性会很有帮助。

7.4.2.1 命名和身份

首先，安全协议中的主体通常以多种不同的名称为人所知 银行帐号、公司注册号、个人姓名加上出生日期或邮政地址、电话号码、护照号码、医疗服务患者编号，或计算机系统上的用户 ID。

一个常见的错误是将命名与身份混淆。同一性是指两个不同的名称（或相同名称的实例）对应于相同的主体（这被计算机科学家称为间接名称或符号链接）。一个典型的例子来自不动产的所有权登记。

希望出售房屋的人通常使用与购买房屋时不同的名字 ：他们可能在结婚或变性时更改了名字，或者开始使用中间名。土地登记系统必须处理很多这样的身份问题。

有两种类型的身份失败会导致妥协：一种是我乐于冒充任何人，另一种是我想冒充特定的个人。前者包括设立账户以清洗网络犯罪所得，而后者的一个例子是更换 SIM 卡（我想克隆 CEO 的手机，这样我就可以盗用公司的银行账户）。如果银行（或电话公司）只要求人们提供两份地址证明，例如水电费账单，那很容易。要求政府签发带照片的身份证件可能需要我们分析诸如“拥有银行帐号 12345678 的 Aaron Bell 是 Aaron James Bell，护照号码为 98765432，出生日期为 3/4/56”之类的陈述。

这可能被视为两个独立系统（银行系统和护照办公室）之间的象征性链接。请注意，此“身份”的后半部分封装了进一步的声明，可能类似于“美国护照 oce s

档案号 98765432 对应于纽约出生登记册上的 3/4/56 一个 Aaron James Bell。”如果亚伦通常被称为吉姆,它会变得更加混乱。

通常,名称可能涉及多个递归步骤,这使攻击者可以选择目标。例如,很多护照欺诈都是预签发欺诈:不法分子以尚未申请护照且出生证明复印件很容易获得的真实公民的名义申请护照。尸检应用也很常见。 “第二人生”的运营商 Linden Labs 推出了一项计划,您可以通过提供驾照号码或某人的社会安全号码来证明自己已年满 18 岁。现在,网络搜索可以迅速为许多人找到此类数据,例如说唱歌手 Tupac Amaru Shakur;是的,Linden Labs 确实接受了 Shakur 先生的驾照号码 即使驾照已经过期并且他已经死了。

也可能存在制度上的失败。例如,阿拉伯联合酋长国开始对所有访客进行虹膜扫描,因为曾因卖淫罪被驱逐到巴基斯坦的妇女几周后出现,并持有不同姓名的真实巴基斯坦护照,并附有不同的“丈夫”。类似的问题导致许多国家签发生物识别签证,这样他们就不必依赖他们不想信任的国家的护照签发者。

除了腐败之外,普遍的失败是原始记录的丢失。在出生登记、婚姻登记和死亡登记都保存在本地并记录在纸上的国家,有些登记丢失了,聪明的冒名顶替者会利用这些登记。你可能认为数字化正在解决这个问题,但数字记录的长期保存即使对富裕国家来说也是一个难题;文档格式发生变化,软件和硬件变得过时,您要么必须模拟旧机器,要么必须翻译旧数据,这两者都不是理想的。各个州都对必须永久保存的电子文件进行了试点项目,例如民事登记,但我们仍然缺乏可信的标准。明智的发达国家仍然保留纸质原件作为长期记录文件。在欠发达国家,您可能不得不在不稳定的政府 IT 的 Scylla 和自然灾害的 Carybdis 之间游刃有余9。

7.4.2.2 文化假设

名称背后的假设因国家而异。在英语世界,人们通常可以随心所欲地使用多个名字;名字就是你所知道的。但一些国家禁止使用别名,而另一些国家则要求他们必须注册。出生、婚姻、民事伴侣关系、性别转变和死亡的民事登记是一项极其复杂的登记,通常被政治化,在许多国家与宗教和身份证件问题联系在一起。国家之间不兼容的规则给移民、游客以及拥有海外客户的公司带来了真正的问题。

在本书的早期版本中,我举了一个例子,即改变的作家

9一边听着开发顾问的警笛声说 把它放在区块链上!

他们在婚姻中的法定姓名经常使用他们的旧名发布。所以我的实验室同事,已故的 Karen Sp¨arck Jones 教授,每年都会收到一封来自大学的信,询问她为什么没有发表任何东西(她作为 Karen Needham 在工资单上倒下了)。出版物跟踪系统无法处理人事系统所知道的一切。随着软件融入一切,系统相互连接,冲突可能会产生意想不到的远程影响 。例如,凯伦也是大英图书馆的受托人,当大英图书馆开始使用持有人所在大学图书馆卡上的名字发行自己的入场券时,凯伦并不以为然。当大学引入以工资单名称为键的 ID 卡系统以统一访问建筑物、图书馆和食堂时,这些问题造成了更大的摩擦。这些具有多个名称的问题现在是主流;使用多个名字的不仅仅是教授、音乐家和小说家。想要阻止公司使用以前性别的名字的跨性别者;分居或离婚时希望停止使用已婚姓名的女性,如果她们逃离虐待伴侣,可能需要这样做;在宗教皈依后使用新名字的人 冲突的根源是无穷无尽的。如果你正在构建一个你希望在全球范围内扩展的系统,你最终将不得不处理所有这些问题。

人类命名约定也因文化而异。如果来自香港,华人可能有英文名和中文名,英文名在姓氏之前,中文名在姓氏之后。南印度、印度尼西亚和蒙古的许多人只有一个名字 单名。印度惯例是添加两个首字母 代表你的出生地和你父亲的名字。所以“BK Rajan”可能指来自班加罗尔的 Kumar 的儿子 Rajan。南印度移民到美国的一个常见策略是使用父名(这里是 Kumar)作为姓氏;但是当西方计算机系统将 Rajan 误解为姓氏时,就会产生混淆。俄罗斯人以名字、父名和姓氏而闻名。冰岛人没有姓氏;如果他们是男性,他们的名字后面是父名,如果是女性,则后面是母名。在过去,当“Maria Trosttad´ottir”到达美国移民局时,移民官了解到“Trosttad´ottir”不是姓氏,甚至不是父名,他们的标准做法是强迫她采用父名作为姓氏(如果她的父亲叫卡尔,就说“卡尔森”)。很多在美国的印度人都有过类似的问题,都造成了不必要的冒犯。在某些文化中,您的名字会在您生完孩子后发生变化。

另一个文化鸿沟通常被认为是英语国家之间的差异,在这些国家,出于隐私原因,身份证是不可接受的10,而在被拿破仑或苏联征服的国家,身份证是常态。鲜为人知的是,大英帝国乐于将身份证强加给它的许多臣民,所以真正的鸿沟可能在于一个国家是否曾经被征服过。

ID 条件的本地历史有各种假设。我认识一些德国人,他们拒绝相信一个国家在没有适当的人口登记和身份证系统的情况下根本无法运作,但承认他们很少被要求提供身份证(例如,开设银行账户或结婚)。

他们的卡号不能用作姓名,因为它是证件号

10除非他们被称为驾驶执照或健康服务卡!

每次发行新卡时都会更改。但是,冰岛身份证号码是静态的;它只是公民的出生日期加上另外两位数字。

此外,法律要求银行帐号包含帐户持有人的身份证号码。这些可能是私人和公共 ID 编号的极端情况。

最后,在许多欠发达国家,为公民登记并向他们发放身份证的行为不仅效率低下,而且具有政治意义 [88]。统治部落可能会试图剥夺其他人的权利,方法是让在他们的领土上登记出生变得困难,或者让获得身份证变得不方便。有时会在选举前重新发行卡片,以更新或加强歧视。卡可以与营业执照和福利金挂钩;拖延可以用来索取贿赂。一些国家(如巴西)在州和联邦层面有单独的登记系统,而其他国家(如马拉维)则让大部分人口未登记。有许多被排斥的群体,例如在父母国籍国以外出生的难民儿童,以及因宗教或意识形态原因而成为无国籍人的群体。

联合国可持续发展目标的具体目标 16.9 是“为所有人提供合法身份,包括出生登记”;许多公司出售由发展援助资助的 ID 系统和投票系统。它们以各种复杂的方式与政府互动,并且有一个完整的研究团体在研究这个 [88]。哦,如果你认为这是第三世界的问题,美国有几个州使用繁琐的登记程序来让黑人更难投票;在 Windrush 丑闻中,英国政府驱逐了一些在外国出生的英国居民,他们自动有权获得公民身份,因为他们没有保持足够好的公民身份书面记录来满足日益仇外的部长。

简而言之,关于政府和人民姓名之间关系的隐藏假设各不相同,这些假设会限制系统设计,并在假设跨国界传播时导致意想不到的失败。工程师必须始终警惕这样一个事实,即面向服务的 ID 是一回事,合法身份或公民身份证明是另一回事。政府总是试图将两者纠缠在一起,但这会导致各种痛苦。

7.4.2.3 名称的语义内容

从一种名称更改为另一种名称可能很危险。一家银行在从按帐号存储客户数据转变为按姓名和地址存储客户数据后遭到起诉。他们编写了一个程序,将每个客户运营的所有账户联系起来,希望这能帮助他们更准确地锁定垃圾邮件。对一位顾客的影响很严重:他为情妇开的银行对帐单被寄给了与他离婚的妻子。

名称的语义会随着时间而改变。在许多交通系统中,车票和通行费标签可以用现金购买,这消除了对隐私的担忧,但将它们与银行账户联系起来更方便,而且这些联系会随着时间的推移而积累。英国养老金领取者用来获得免费巴士旅行的卡也开始是匿名的,但在实践中,巴士公司试图将卡号与其他乘客标识符联系起来。事实上,我曾经拿到过五金店会员卡

使用随机帐号（并且没有信用检查）。在商店被一家超市接管并且超市开设了一家银行之后，我有机会把它变成一张银行卡。

7.4.2.4 名称的唯一性

当我们生活在小社区中时，人类的名字就演变了。我们从 o 开始只是用名字，但到了中世纪晚期，旅行的增长导致政府胁迫人们采用姓氏。这一过程花费了一个世纪左右的时间，并与纸张作为羊皮纸的低成本和防篡改替代品引入欧洲有关；纸张使人们以前用来支付道路通行费等的徽章、印章和其他不记名代币可以用提到他们名字的字母代替。

人员、企业和管理人员大量涌入互联网的速度太快，无法适应社会。现在在线的人（和系统）比我们过去处理的要多得多。那么我们怎样才能使人类难忘的名字独一无二呢？正如我们上面所讨论的，Facebook 通过将每个人与他的一组朋友聚集在一起并添加一张照片来告诉另一个约翰史密斯。

也许另一个极端是加密名称。名称是公钥或被命名对象的其他稳定属性的哈希值。已经提出了各种机制来将真实世界的名称、地址甚至文档内容永久地映射到哈希函数的位串输出上（例如，参见 [845]）。您甚至可以使用生物识别的哈希值或物体的表面微观结构，再加上合适的纠错码。

加密货币和区块链的世界大量使用基于哈希的标识符。这样的机制可能无法重用名称；由于过期的域名经常被坏人购买和利用，这有时很重要。

这并不是全新的，因为它在事务处理中很常见，只是给每件事和每个人一个数字。但是，如果您没有在正确的位置放置足够的唯一性，这可能会导致失败。例如，一家英国银行通过将序列号打印在用于记录交易的文具上，为交易分配了唯一的序列号。有一次，当他们想向海外汇款 2000 万英镑时，接线员错误地输入了 1000 万英镑。下令支付 1000 万英镑的第二笔款项 但这从文书工作中获得了相同的交易序列号。因此，向 SWIFT 发送了两笔具有相同日期、收款人、金额和序列号的付款 第二笔作为重复付款被丢弃 [309]。

7.4.2.5 名称和地址的稳定性

许多名称包含某种地址，但地址会发生变化。虽然我们在剑桥还有一本电话簿，但每年大约有四分之一的地址发生变化；使用工作电子邮件，营业额可能更高。当我们在 1990 年代后期尝试开发一个使用加密电子邮件的人的目录以及他们的密钥时，我们发现更改条目的主要原因是更改了

电子邮件地址 [103]。 （有些人认为这会是钥匙丢失或被盗;来自这个来源的贡献恰好为零。）现在情况可能更稳定了。大多数人都试图保留他们的个人手机号码,因此他们往往会长寿,个人电子邮件地址也是如此。像谷歌和微软这样的大型服务提供商通常不会两次发布相同的电子邮件地址,但其他公司仍然这样做。

分布式系统先驱认为将地址放在名称中是一件坏事 [1353]。但是分层命名系统可以涉及多个抽象层,每层的一些地址信息构成上一层名称的一部分。此外,命名空间是否更平坦取决于应用程序。通常人们最终会在部门和组织级别使用不同的名称（例如 rja14@cam.ac.uk 和 ross.anderson@cl.cam.ac.uk 在我自己的例子中）。因此,名称和地址之间的清晰分界并不总是可能的。

授权具有地址的许多（但不是全部）属性。肯特定律告诉设计者,如果一个凭证包含它可能用于什么的列表,那么这个列表上的东西越多,它的有用期限就越短。一个类似的问题困扰着名字是复合的系统。例如,一些在线企业通过电子邮件地址和信用卡号的组合来识别我。这显然是不好的做法。除了我有多个电子邮件地址这一事实外,我还有几张信用卡。

使用假名是有充分理由的。在 Facebook 出现之前,人们认为儿童和年轻人使用不易与他们的真实姓名和地址联系起来的在线名称是明智的。当您在 22 岁离开大学时找到第一份工作,或者在 45 岁时找到 CEO 的工作,您不希望搜索结果显示您所有的青少年咆哮。许多人还会不时更改电子邮件地址以避开垃圾邮件;我过去常常为我购物的每个网站提供不同的电子邮件地址。另一方面,一些警察和其他机构希望人们不要使用假名,这将我们带入了在线可追溯性的整个问题 我将在第二部分中讨论。

7.4.2.6 名称使用限制

命名和社会之间的相互作用给我们带来了一个更深层次的问题:一些名字只能在有限的情况下使用。这可能是法律规定的,就像美国社会安全号码及其在其他一些国家/地区的等效号码一样。有时这是营销问题:相当一部分客户会避开需要太多信息的网站。

受限制的命名系统以意想不到的方式相互作用。例如,医院使用患者编号作为医疗记录数据库的索引是相当普遍的,因为这可能允许研究人员出于某些目的使用假名记录。当健康维护组织的合并或政策变化迫使医院引入统一名称时,这会导致问题。

例如,关于哪些化名可以用于哪些目的,英国的卫生服务部门长期以来一直存在争论。

最后,当我们谈到法律和政策时,名称的定义会引发新的和意想不到的陷阱。例如,法规允许警察

收集通信数据 即谁打电话给谁以及何时打电话的记录 通常比管理电话窃听的规定宽松得多;在许多国家,警方只需询问电话公司即可获得通讯数据。这导致了对 URL 状态的争论,其中包含传递给搜索引擎的参数等数据。很明显,一些警察想要一份名单,列出所有点击 http://www.google.com/search?q=cannabis+cultivation; 等 URL 的人。同样清楚的是,许多人会认为如此大规模的拖网捕捞是对隐私的不可接受的侵犯。英国法律的决议是将 trac 数据定义为足以识别正在与之通信的机器的数据,或者用通俗的语言来说就是“直到第一个斜线的一切”。我稍后会在“监视还是隐私?”一章中更详细地讨论这个问题。

7.4.3 名称类型

不仅在各个层面命名都很复杂 从技术上到组织上再到政治上 而且一些真正棘手的问题还涉及各个层面。我在介绍中指出,名称不仅可以指代人(和代表他们行事的机器),还可以指代组织、角色(“守望者”)、群体和复合结构:角色中的委托人 爱丽丝作为经理;委托 爱丽丝代表鲍勃;合取 爱丽丝和鲍勃。连词通常表示隐含的访问规则:“Alice 作为分行经理,Bob 作为分行会计师组的成员”。

这仅仅是个开始。名称也适用于服务(例如 NFS 或公钥基础设施)和通道(可能意味着线路、端口或加密密钥)。相同的名称可能指代不同的角色:“作为计算机游戏玩家的爱丽丝”应该比“系统管理员爱丽丝”拥有更少的特权。安全文献中通常使用的抽象是将它们视为不同的不同主体。因此,名称和主体之间没有简单的映射,尤其是当人们带着自己的设备上班或将工作设备带回家时,因此在同一平台上可能有多个相互冲突的名称或角色。许多组织开始仔细区分“爱丽丝本人”、“爱丽丝作为在爱丽丝家用笔记本电脑上运行的程序”和“代表爱丽丝在公司云上运行的程序”,我们在本章讨论了一些可能的机制关于访问控制。

如果您弄清楚它们是如何由底层业务流程驱动的,那么通常更容易分析功能紧张。企业主要是想获得报酬,而政府则想唯一地识别人。实际上,企业需要您的信用卡号码,而政府需要您的护照号码。基于激励的分析有时可以表明命名系统是开放的还是封闭的、本地的还是全球的、有状态的还是无状态的 以及维护它的人是否是愿意为失败付出代价的人(经济学是其中之一)可靠性的关键问题,是下一章的主题)。

最后,虽然我已经说明了很多关于人的命名问题 因为这使问题更加直接和引人注目 但许多相同的问题以不同的方式出现在加密密钥、唯一产品代码、文档 ID、文件名、URL 等等。当我们深入现代企业网络的内部时,我们可能会发现 DNS

循环到多台机器,每台机器都有自己的 IP 地址,在一个名字后面;或 Anycast 到多台机器,每台机器都使用相同的 IP 地址,使用一个名称;或 Cisco 的 HSRP 协议,其中 IP 地址和以太网 MAC 地址从一个路由器移动到另一个路由器。 （我将在第 2 部分讨论网络安全的更多技术方面。）无论如何,随着系统规模的扩大,依赖简单、可互换和不可变的名称变得越来越不现实。您需要仔细界定命名范围,了解谁控制您所依赖的名称,弄清楚它们有多狡猾,并将您的系统设计成可靠的,尽管它们有局限性。

7.5 总结

许多安全的分布式系统已经产生了巨大的成本,或者产生了严重的漏洞,因为它们的设计者忽略了如何构建（以及如何不构建）分布式系统的基础知识。这些基础知识中的大部分已经出现在计算机科学教科书中了整整一代人的时间。

许多安全漏洞都是一种或另一种并发故障;系统使用旧数据,进行不一致的更新或以错误的顺序进行更新,或者假设数据是一致的,但实际上它们并不一致,甚至不可能一致。使用时间来安排交易可能会有所帮助,但知道正确的时间比这更难

似乎。

容错和故障恢复至关重要。提供从安全故障以及随机物理和软件故障中恢复的能力是许多组织保护预算的主要目的。在更技术层面上,保护和弹性机制之间存在重要的相互作用。拜占庭故障 有缺陷的进程合谋而不是随机失败 是一个问题,它与我们对加密工具的选择相互作用。

冗余有很多种不同的形式,我们必须使用正确的组合。我们不仅需要防止故障和企图操纵,还需要防止蓄意拒绝服务,这可能是更大攻击计划的一部分。

许多问题也源于试图让一个名字做太多的事情,或者对它做出在一个特定系统、文化或管辖范围之外不成立的假设。例如,应该可以通过取消用户名来撤销用户对系统的访问权限,而不会因为其他功能被撤销而被起诉。最简单的解决方案通常是为每个委托人分配一个唯一的标识符,不用于其他目的,例如银行帐号或系统登录名。但是当合并两个使用不兼容的命名方案的系统时会出现很多问题。有时这甚至可能是偶然发生的。

研究问题

我在本章中谈到了许多技术问题,从安全时间协议到命名的复杂性。但也许最重要的研究问题是弄清楚如何设计出在面对恶意时具有弹性的系统,能够优雅地降级,并且一旦攻击过去就可以恢复其安全性。过去曾推出各种补救措施,从让政府向每个人发放身份证到将其全部放在区块链上。

然而,这些魔法子弹似乎并没有杀死任何地精。

工程师研究失败总是一个好主意;我们从倒塌的一座桥中学到的东西比从几千座没有倒塌的桥中学到的更多。我们现在有越来越多的失败 ID 系统,例如英国政府的 Verify 计划 试图为公共服务创建联合登录系统,但已于 2019 年放弃 [1392]。有一个研究社区在研究欠发达国家的 ID 系统故障 [88]。然后是区块链未能实现其最初的承诺,我将在本书的第 2 部分中讨论。

也许我们需要更仔细地研究我们可以从损坏的安全状态中整齐地恢复的条件。恶意软件和网络钓鱼攻击意味着在任何给定时间都有一小部分（但非零)客户银行账户受到犯罪控制。然而,银行系统仍在继续。受感染的笔记本电脑和手机的比例因国家/地区而异,其影响可能值得更仔细地研究。

经典计算机科学理论将分布式系统中的收敛视为本质上的技术问题,其解决方案取决于技术特性（在一个层面上,原子性、一致性、隔离性和持久性;在另一个层面上,数字签名、双重控制和审计）。也许我们需要一个更高层次的观点,在这个观点中,我们会问我们如何就世界状况达成充分的共识,并且不仅要纳入技术弹性机制和保护技术,还要纳入欺诈受害者获得补救的机制。试图避免对强有力的补救措施的需求的纯技术机制实际上可能会使事情变得更糟。

延伸阅读

如果您不熟悉本章中的材料,您可能是从数学/加密货币背景或芯片/工程甚至法律/政策背景来看这个主题的。
计算机科学专业的学生有很多关于分布式系统的讲座;为了赶上进度,我建议使用 Saltzer 和 Kaashoek [1640]。多年来我们向学生推荐的其他书籍包括 Tanenbaum 和 van Steen [1860] 以及 Mul lender [1353]。美国国家研究委员会 2003 年的一份报告，“谁去了那里?从隐私的角度进行身份验证” ,讨论了身份验证和隐私之间的权衡,以及它们如何难以扩展 [1039]。

最后,Pat Helland [880] 最近对命名进行了讨论。