Mercury: Bandwidth-Effective Prevention of Rollback Attacks against Community Repositories

1. 背景：

1.1 问题

目前一些流行的社区存储（community repository）Docker Hub、PyPI、RubyGems上存在很多的软件以共享给大量的用户，这些数量庞大的软件和用户群使得他们容易成为攻击的目标。比如：在一次存储破坏后，恶意的一方能够对无辜的用户发起一系列的攻击，这些攻击中就包括了回滚攻击--利用软件的过时的版本或者有漏洞的版本，从而有针对地对这些软件的漏洞和缺陷而进行攻击。

1.2 当前解决方法

通过高频率的更新包来防止回滚攻击，这样会导致平均每个用户每个月要下载的文件大小是平均包大小的2-3倍，这几乎是不可能完成的。

2. Mercury 的提出

防止回滚攻击的解决方案需要满足几个重要的特点：没有管理开销、简单的客户端通信、低开销。

Mercury 使用了一些技术从而能够限制用户遭受回滚攻击,这个技术的关键就是总是去比较由存储签署的当前和之前的版本信息目录，从而这些攻击就会很容易的被侦查到。Mercury 对于回滚攻击是带宽有效的，是因为它只下载了所有项目的版本号，而不是所有的文件包。Mercury 压缩版本信息，带宽有效技术，使用增量压

缩来传递之前和当前版本列表信息的差异，如果 Mercury 被部署在 PyPI 上，每个用户每个月下载的元数据只有平均包大小的 3.5%。

3. 贡献

1、发现了现存系统在防止回滚攻击的时会导致过高的带宽代价。

2、设计和完成了一个带宽有效的系统，这个系统依靠存储持续地指向最新的项目版本号来阻止回滚攻击。

3、使用对 PyPI 请求来评估 Mercury 的效率，Mercury 通过让每个用户下载大小为平均包大小的 3.5%的元数据来防止回滚攻击，另外新用户需要下载平均包大小的 48%的元数据，而其他两个系统的开销则是 1152%和 3092%。

4. Mercury 是一个新的安全系统

1、把对开发者的信任转移到存储上来，当前存在的系统代价是昂贵的，因为它们是基于在当前存储上是没有能够正确地指示工程元数据文件的版本号的这个假设来设计的。

2、安全分析，Mercury 一个主要强大的地方就是攻击者不能回滚攻击在最近一次用户访问过且之前就存在的工程。这是因为无论用户什么时候安装一个包，包管理者总是会去对当前的快照元数据文件和之前的副本进行比较。

3、从存储破坏中恢复，通过代替包管理者的快照元数据副本来解决这个问题。所以管理员必须使用线下备份来恢复所有的项目元数据和包到被破坏之前的一个点，然后在线 keys 用来签署可以被新 keys 唤醒和替代的快照元数据，这借鉴了 TUF。

4、确保过时的包管理者安全，Mercury 用户的安全依赖于包管理者所拥有的相关的版本数量。

5、不允许在快照元数据中删除工程。如果包管理者从快照元数据文件中删除了版本信息，想要控制存储库的攻击者可以重置一个已知的版本号，所以更好的保护使用 Mercury 的存储库，工程应该不允许从快照元数据中删除。Docker 和 Flynn 都是使用这种方式。

6、针对恶意镜像的保护：这些镜像不会去篡改快照元数据，但是他们能够用带有恶意的工程元数据文件来代替一些原始的工程元数据文件。使用 Mercury-hash（一个 Mercury 的变体）来解决这个问题，这会导致代价是 Mercury 的 7 倍，但是这是可以被接受的，因为防止恶意镜像破坏是很重要的。

5. 带宽成本的评估

1、从 PyPI 上面获得了一个月的用户下载包的日志，从而把 Mercury 与其它 4 个安全系统（GPG/RSA、Mercury-hash、TUF-version、TUF）进行带宽成本比较。

2、工程数量对带宽的影响，主要比较了一个存储上相对于 PyPI 拥有较多或者较少的工程数量对带宽代价变化的影响。一方面，我们研究了新用户的初始成本是如何随着最大工程数量的变化而变化的；另一方面，如果研究项目的数量大于上一个版本，我们使用线性回归来推断成本。

3、工程更新频率对带宽的影响。主要比较了一个存储上相对于 PyPI 拥有较高的或者较低的工程更新速率对带宽代价变化的影响。一方面我们研究了返回用户的循环成本是如何变化的。为了研究当工程更新的速率降低时候的成本，我们通过增加第一次和其它任意接下来版本的时间间隔来降低工程更新的速率。另一方面，如果研究这个比率增加的话,我们用线性回归来推断基于较小速率的更大速率的成本。

6. 相关工作

1、负责任的系统，与 PeerReview、CATS、Cloud-Proof 不同，它们只能在回滚攻击发生后才能侦查到，而 Mercury 被设计成在回滚攻击之前就可以侦查到。

2、软件存储安全系统。之前工作表明软件的更新者更容易遭到像回滚攻击这样的安全问题，著名的 Linux 包管理者使用一个安全框架来防止恶意镜像或 CDNs。但是，与 Mercury 不同，它不一定能够承受得住对原始存储库的破坏。Revere 使用了一个自组织、P2P 重叠网络来提供安全更新。然而，一个 P2P 的安装将会增加部署社区存储库的复杂度，因此，管理人员认为这是不切实际的。由于 Mercury 不需要 p2p 安装，所以它是一个更容易实现的系统。

3、不可信存储服务器的文件系统。我们讨论了一些文件系统（如 ECFS、TCFS），这些系统本质上是为了检测攻击者是否已对包进行了篡改。最大的区别是，Mercury 不是一个文件系统，这意味着存储库可以自由使用他们喜欢的任何文件系统。Mercury 在现有文件系统的基础上工作，并要求存储库只添加一层已签名的元数据，并在安装包之前修改包管理器来验证这些元数据。

4、具有不同信任假设的安全系统。SUNDR 是一个为软件存储而设计的一个文件系统。不同于 Mercury，尽管 SUNDR 使用一个不受信任的服务器，但是可以防止回滚攻击以及检测分叉攻击。然而，这个代价是 SUNDR 要求客户相信其他客户会诚实地报告存储库是否进行了分叉攻击。这会有一个问题，就是单个故障或恶意客户机可能意外或有意地构建一个诚实的存储库。

5、Byzantine 容错安全系统。Byzantine 容错（BFT）系统使用多个副本而不是一个单一的服务器执行的操作。不幸的是，PBFT 需要管理员管理 3 f + 1 个独立的副本，而不是一个单一的服务器（其中 f 是能够容忍破坏的存储库的最大数量）。这

大大增加了管理负担。Mercury 仅仅使用一台服务器，因此它不那么昂贵，而且更容易部署。

7. 结论

随着社区存储库的不断普及，对可靠和经济上可行的安全系统的需求也会保护用户免受许多可能的攻击。要求开发人员指出最新版本号的解决方案在实际使用中成本太高。在本文中，我们介绍了一个安全系统，它使用社区存储库来指示项目的最新版本号。虽然攻击者可以破坏存储库，但 Mercury 始终可以防止回滚攻击，而且其恢复机制有助于用户从快进攻击中恢复。由于使用存储库中的一个键来为每个项目签名版本号，这使得 Mercury 能够有效地利用带宽来防止回滚攻击。