**安全补丁模式分析报告**

# 研究问题与重要性

软件供应链的蓬勃发展带动了计算机软件开发的空前进步，使用商业和开源第三方软件已经是软件开发的关键策略，2021年VDC的研究报告指出，商业第三方二进制文件占软件产品的33%。第三方组件的引用使得第三方二进制中的漏洞随供应链传播。基于深度神经网络的二进制代码相似性检测（BCSD）方法在供应链的漏洞检测方面，发挥了强大的潜力。然而，漏洞函数与其补丁后版本往往存在高度相似性，这导致了大量误报（将补丁后的安全函数判断为存在漏洞的函数），这影响到了BCSD方法在软件供应链威胁检测中的应用。为了降低误报，提升基于深度学习的BCSD的实用性，首先面对的问题就是判定疑似漏洞函数是否进行了安全修复。

为了解决上述问题，需要识别漏洞二进制和目标二进制之间的差异，并生成补丁签名或描述，最后进行判断补丁引入的差异是否与安全补丁相关。

上述工作不仅可以降低漏洞检测中的误报，还可以用于识别隐蔽的安全补丁，进而挖掘潜在的漏洞。

# 难点与挑战

识别二进制补丁是否为安全补丁面临以下三个难点：

**补丁差异难定位：**以基本块为粒度难以精准定位补丁引入的差异。现有的差异提取方法基于基本块相似性匹配，补丁引入的基本块拆分和重组影响匹配的准确性。此外，基本块中可能包含大量补丁无关语义，定位到的差异存在大量补丁不相关内容，影响对补丁语义的分析理解。

**补丁签名难提取：**现有方法难以兼顾细粒度的变量级别的更改和粗粒度的结构层面的更改。补丁存在性检测相关工作关注具体函数、具体漏洞、具体的补丁，保留了大量的“具体”特征，关注补丁点附近的函数行为，而不关注补丁究竟干了什么。无法应用于通用的安全补丁检测。

**安全补丁模式难定义：**难以定义、描述安全补丁区别于非安全补丁的特征与模式。

# 安全补丁模式

针对上述难点与挑战，本项目对大量安全补丁进行了广泛的分析与调研，意图提取安全补丁不同于功能性补丁的补丁模式。发现每个二进制函数，都具有双重身份，既有可能是调用函数，也可能是被调用函数。所以对于每个函数，我们要在单一函数视角的基础上，补充调用者和被调用者两种视角来分析补丁引入的差异。另外，现有工作对安全补丁做了什么、如何做进行了归纳，但并未分析安全补丁的意图，本项目认为从意图入手可以更好区分安全补丁与非安全补丁。

从意图和角色入手，本项目对安全补丁进行以下归纳与分析：

单一函数的意图：

**P1：严格准入。**通过条件检查的变化(增、删、改、顺序变化)，所见可以进入关键功能块的状态。

e.g CVE-2016-1833，A和A’，B和B’分别为补丁前后未引入变化的基本块组，引入的安全补丁通过条件检查的新增（check1）和条件检查顺序和条件的变更，减少了准许进入两部分的状态。

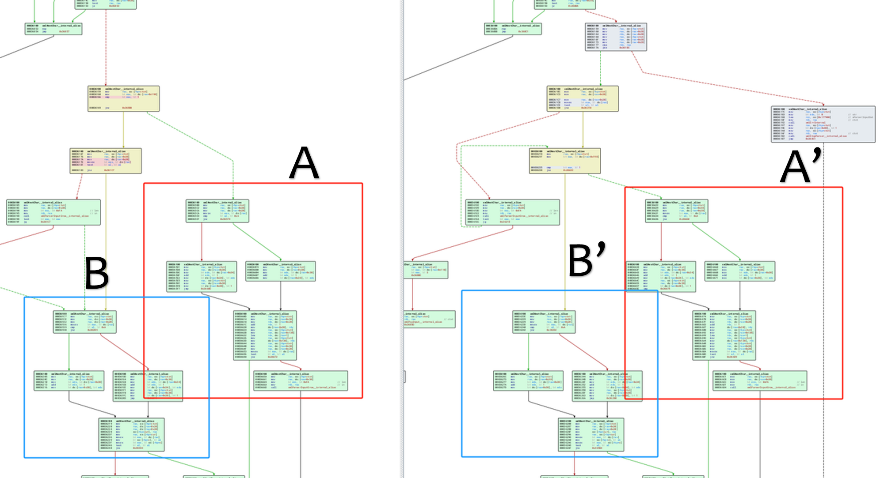


图1 CVE-2016-1833安全补丁引入的局部变更

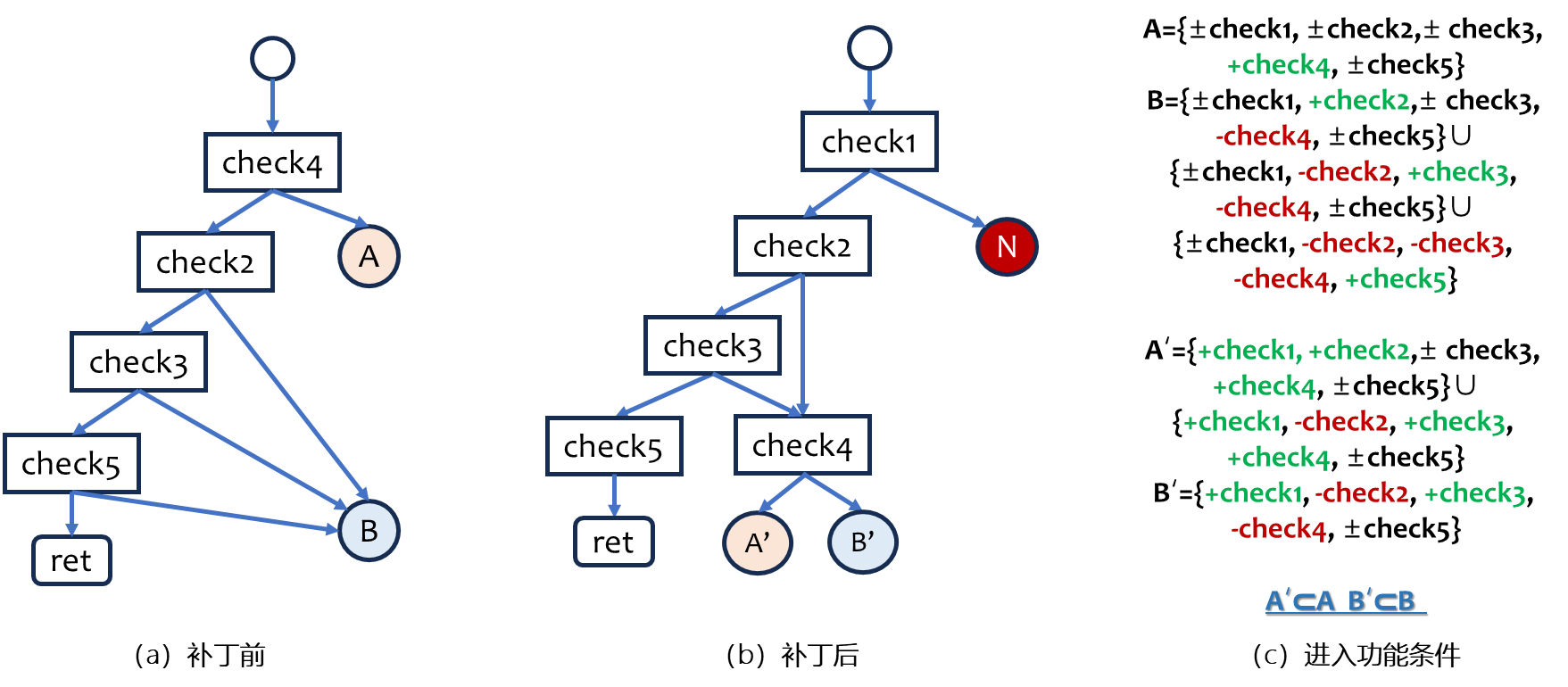


图2 严格准入安全补丁示例

**P2：控制循环&状态保持。**关键路径上（关键块前后），对变量进行+1/-1，且该变量仅进行这种操作。

**P3：“敏感值”检查。**条件分支的更改往往被视作一种重要的补丁特征（PatchDB：70%的安全补丁都与条件分支的更改有关）。然而，功能性补丁也存在大量的条件分支。所以可以以条件分支为锚点，结合上下文语义推测补丁意图。安全补丁往往涉及一些“敏感值”的检查，这里的“敏感值”包括，参与比较的参数/变量敏感、比较方法（是否包含“=”、无符号比较还是有符号比较）、参数比较的数值敏感（如“0”、“0xff”等）。

被调用函数的意图：

**P4：返回状态可更好区分不一致状态。**

一些变量级的修改在二进制代码中难以识别，其改动极小，在二进制中仅改变了几个变量，而而剥离的二进制本身也不包含变量名称。但从意图来看，一些变量改动目的是使返回值更好的区分，函数的执行路径。所以，返回状态可以看作这类补丁的一个特征。

e.g CVE-2022-1343--CWE-295 证书验证不当，引入err变量存储X509\_STORE\_CTX\_get\_error(ctx)的返回值。

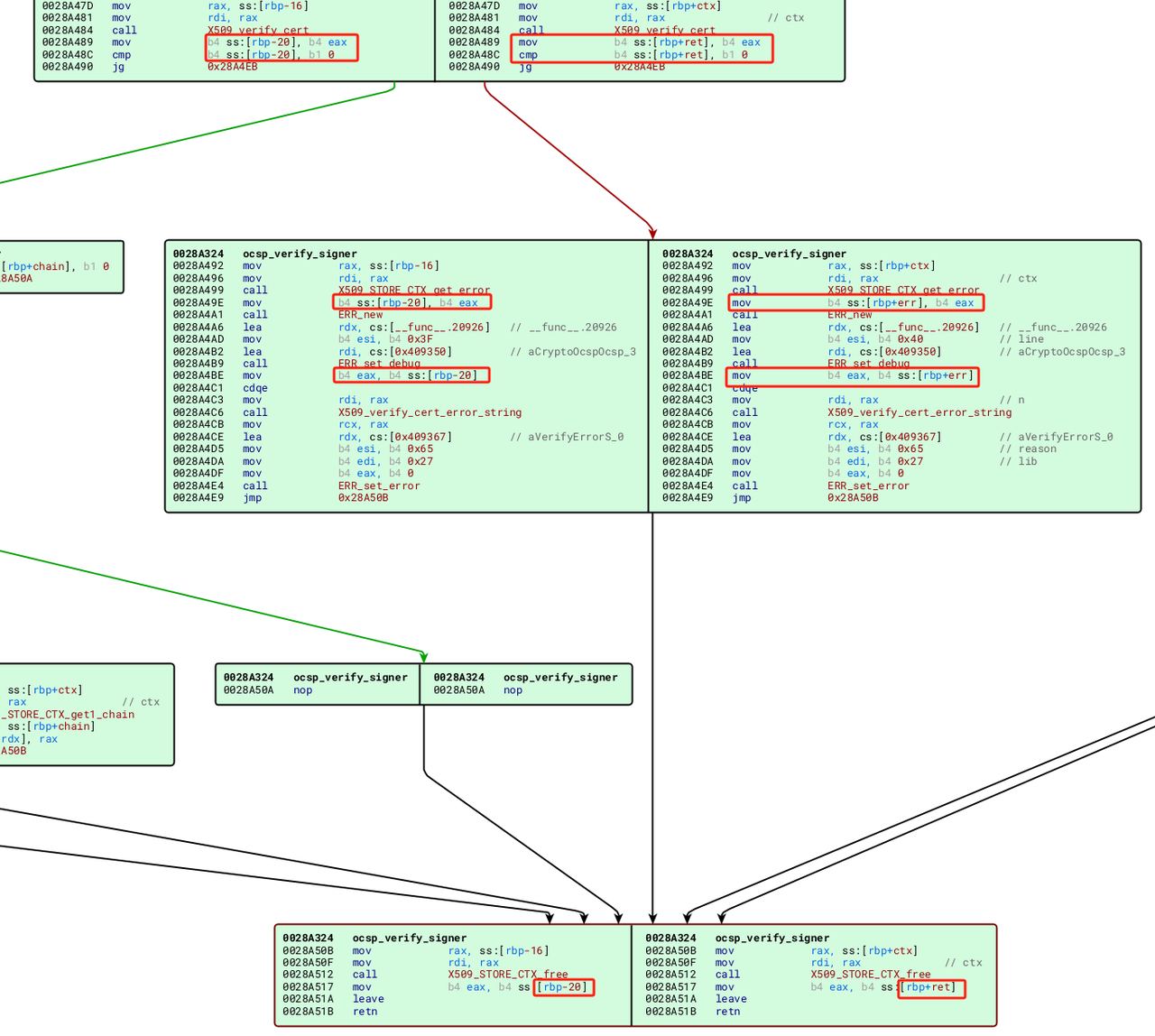


图3 CVE-2022-1343补丁前后二进制代码

从返回值分析来看，补丁前返回状态{-1；>0；不确定}，补丁后返回状态{-1；>0; <=0}。修改后，可以更好区分执行了错误处理和还是核心功能。

**P5：“错误状态”声明。**需要将错误状态，传递到调用函数。

一般情况下，新增状态与之前的多个状态一致，且返回值一般小于等于0。此外，其相应的路径与错误处理相关。

调用函数的意图：

**P6：保障函数调用的安全性。**对函数参数和函数返回值进行检查，对函数参数的来源和计算方法进行更新，将调用函数替换为等效的安全的安全函数等。