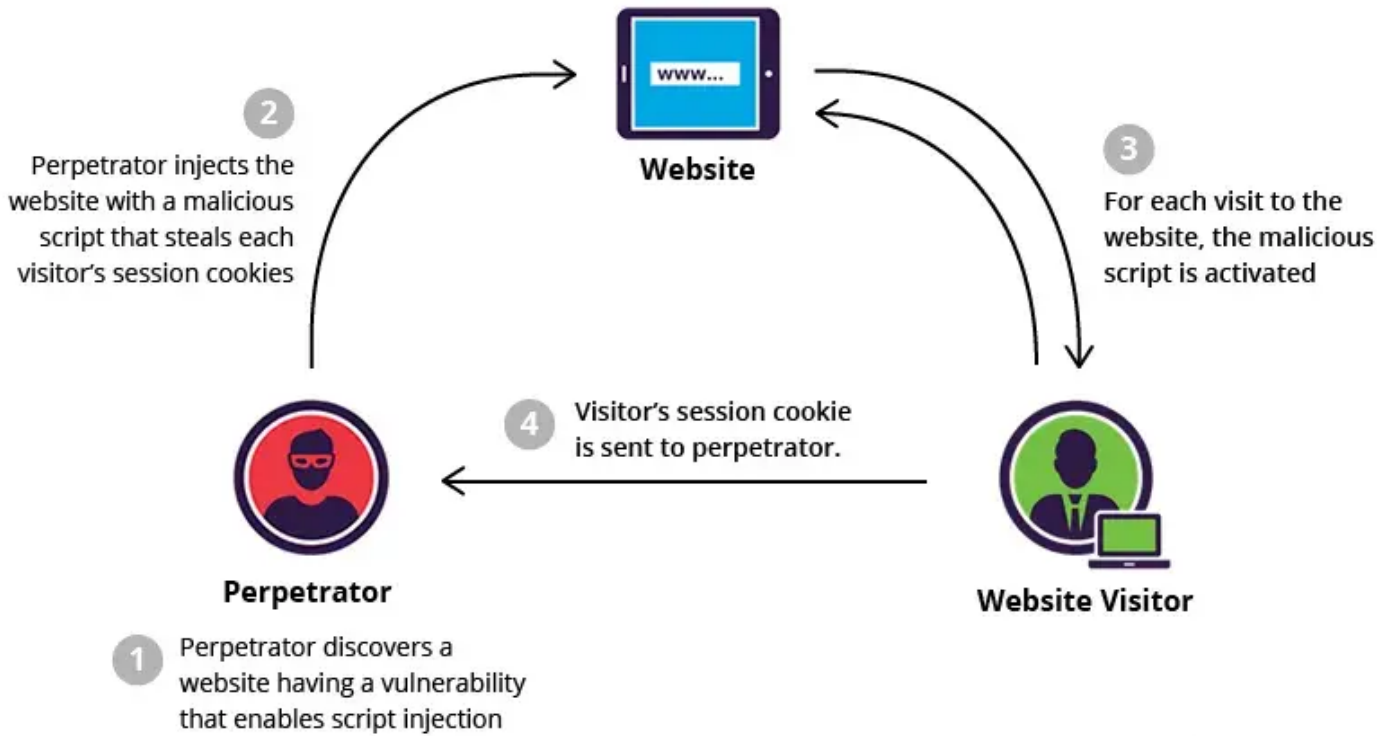
七大最常见的前端安全攻击

如果你正在构建 web 应用，那么你不仅要关注应用的开发，还要关注其安全性。

> 事实上，由于 web 应用程序设计不当，每天发生的网络攻击超过 2,200 起。

因此，你必须了解 web 应用中可能发生的不同类型的攻击，以及如何防范这些攻击。

1. **跨站脚本攻击 (Cross-Site Scripting：XSS)**



1. 攻击者发现网站存在漏洞，然后注入脚本。
2. 攻击者在网站上注入恶意脚本，目的是窃取每个访问者的会话 cookie。
3. 每次访问网站，恶意脚本都会被执行。
4. 访问者的会话 cookie 会被发送给攻击者。

跨站脚本攻击（XSS）是最常见的攻击之一。在 XSS 攻击中，攻击者将恶意脚本注入可信网站，然后在用户浏览器中执行。

什么会导致 XSS 攻击?

XSS 攻击的主要原因之一是在页面上渲染用户输入内容之前，对这些输入未处理或处理不当。例如，攻击者可以使用 JavaScript 注入恶意代码，并在应用渲染 DOM 时执行。

这些恶意代码最终会访问并窃取用户会话令牌、cookie 和其他存储在浏览器中的敏感信息。

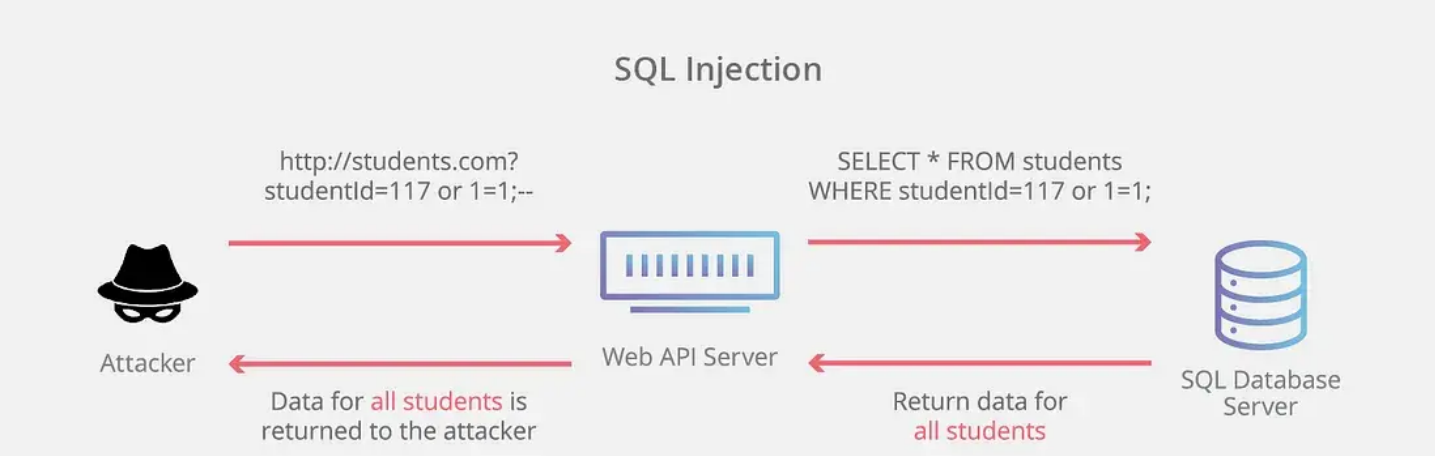
> 笔者举例：比如用户输入 `<script>alert('foolish!')</script>`，未进行处理的话，在展示这段内容时可能会在浏览器提示 foolish。

如何防止 XSS 攻击?

防止跨站脚本攻击并不难，对用户输入内容的验证是核心。

确保对用户插入的数据进行过滤，内容进行编码。此外，考虑使用内容安全策略（CSP）来限制加载的资源和脚本。或者，只需使用 Angular、Vue 和 React 等框架，这些框架都有针对跨站脚本攻击的内置预防机制。

5. **SQL 注入**



1. 攻击者访问 `http://student.com?studentId=117 or 1=1;` 接口。
2. 服务器生成 SQL（`SELECT *FROM students WHERE studentId=117 or 1=1;`） 访问数据库。
3. 数据库返回所有学生数据给服务端。
4. 服务端接口把所有学生数据返回给攻击者。

SQL 注入是一种存在已久的致命攻击。攻击会操纵数据库查询以获得未经授权的数据库访问权限，从而执行恶意活动，如破坏数据库或窃取敏感数据。

简单地说，SQL 注入可让攻击者从你的前端执行 SQL 查询。这可能会导致破坏性操作，使你的数据库宕机!

例如，2020 年对爱沙尼亚中央健康数据库的攻击导致几乎所有爱沙尼亚公民的健康记录泄露，就是近年来发生的大规模 SQL 注入事件的一个令人痛心的例子。

如何防止 SQL 注入?

防止 SQL 注入的策略分为两个部分：

1. 首先，需要确保前端输入的字段经过正确过滤和编码。你需要防止用户在输入的字段中插入恶意代码。

2. 前端对内容验证后，后端接口对接收到的参数进行验证和转义同样重要。不要相信你的接口参数，因为任何人都可以获取你的接口地址并开始发送恶意输入。因此，后端也要确保对参数进行验证。此外，利用 Burp Scanner、sqlmap、jSQL Injection 和 Invicti 等工具来检测应用中潜在的 SQL 攻击和相关漏洞。
3. **跨站请求伪造（Cross Site Request Forgery: CSRF）**

1. 攻击者伪造一个向网站转账的请求。
2. 攻击者会将请求嵌入在一个超链接中，并将其发送给可能已登录网站的访问者。
3. 访问者点击链接，无意中将请求发送至网站。
4. 网站验证完请求并从访问者账户向攻击者转账。

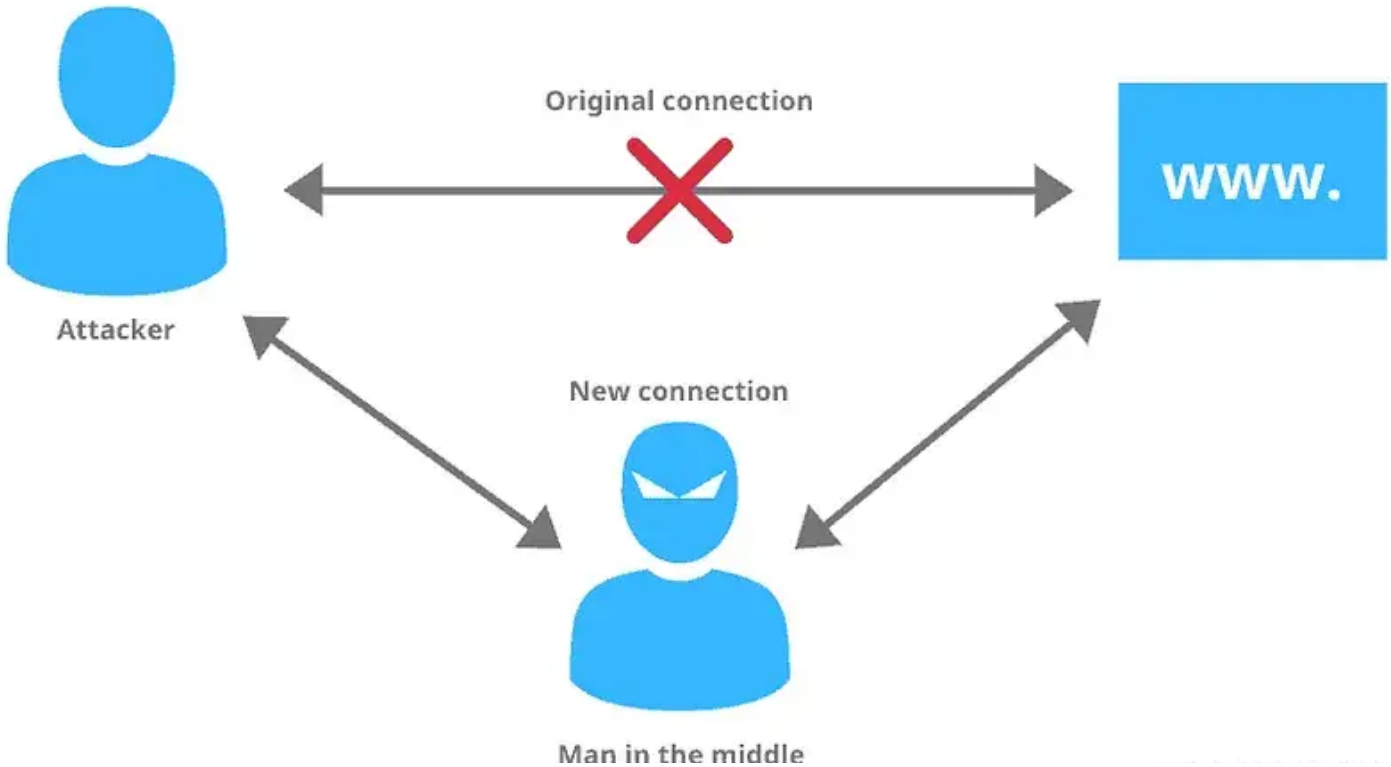
> 跨站请求伪造（CSRF）是一种前端安全攻击，它会欺骗特定应用上的认证用户，让他们执行他们不希望执行的请求。
>
> 这可能是一个伪装过的表单、链接或按钮，在用户发出请求时会更改用户凭据、删除或篡改敏感数据，或无意中从用户的银行账户中转移资金。
>
> 笔者举例：`<img src="http://bank.example/withdraw?account=bob&amount=1000000&for=Mallory" />`，登录的用户访问到这个 img 标签时，会发起转账请求并自动带上登录相关的 cookie。

如何防止 CSRF 攻击?

防止 CSRF 攻击的最简单方法之一就是使用从服务器生成的 CSRF 令牌。你可以与客户端共享这些令牌，这样服务端就可以在收到的每个请求中检查令牌并验证其真实性。如果客户端未能提供准确的令牌，服务器就可以拒绝所请求的操作。

此外，.NET、Joomla、Spring（Spring Security）和 Ruby on Rails 等框架都内置了 CSRF 支持，可防止此类攻击。

5. **中间人攻击**



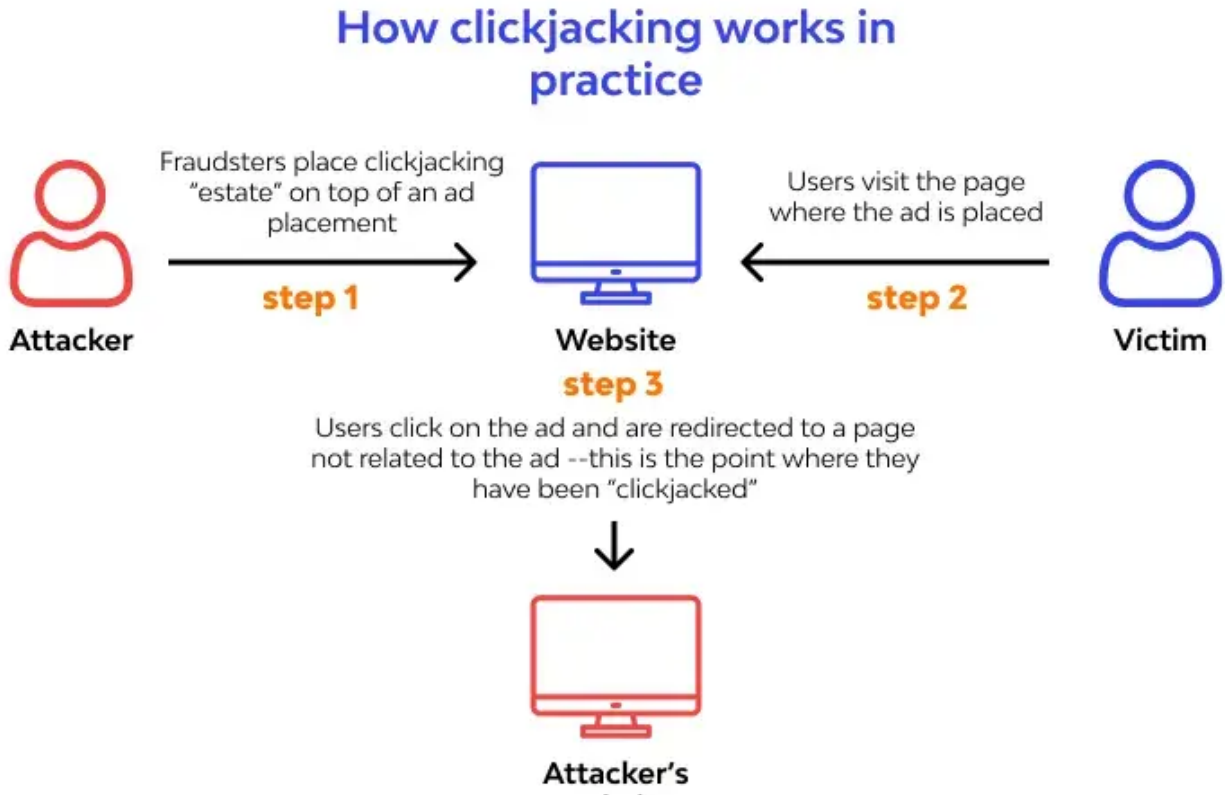
中间人（MitM）攻击指攻击者截获并操纵双方之间传输的信息。

> 例如，攻击者可以拦截你与 `Facebook.com` 的连接，窃取你的凭据，然后将你的请求转发给 Facebook。

当攻击者利用不安全的通信渠道（通常通过公共 WiFi）时，就可以发生此类攻击。这种攻击的受害者并不觉得自己受到了攻击，因为他们认为自己正在与服务器进行非常正常和安全的对话，而他们正在共享的信息却在途中被窥探或篡改。

如何防止中间人攻击?

1. 使用安全的互联网连接，并在不使用应用时及时退出登录。
2. 不要连接不熟悉的网络。例如，不要连接咖啡馆里的免费 WiFi。
3. 使用安全通信协议，如 HTTPS 和 TLS，对传输中的所有数据进行加密。
4. **点击劫持**



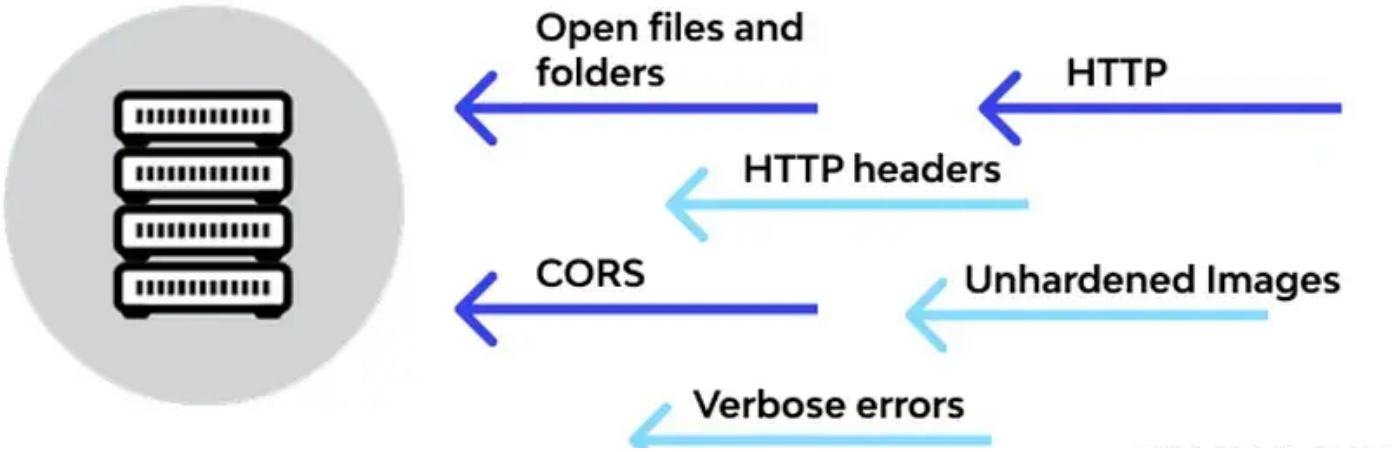
1. 攻击者将点击劫持的 “内容” 放在广告位置的顶部。
2. 用户访问广告所在页面。
3. 用户点击广告后会被重定向到与广告无关的页面 - 这时用户就被 “点击劫持” 了。

> 点击劫持（A.K.A - UI 纠错攻击）是一种欺骗机制，它欺骗用户点击了不是他们想要访问的东西。
>
> 它将隐藏元素覆盖在网站上一些可点击的内容之上。在这种情况下，用户实际上是在点击一个非预期元素，而该元素可能会在未经用户同意的情况下触发资金转移等意外操作。
>
> 笔者举例：攻击者开发了一个页面，里面用 iframe 加载平铺了百度页面，并在百度搜索的按钮上覆盖一个元素，用户点击搜索时，实际点击的是搜索之上的那个元素。

如何防止点击劫持攻击？

为了降低点击劫持攻击的潜在风险，可以使用一种机制，即使用 X-Frame-Options 标头，以确保你的网站没有嵌入到其他网站或 IFrames 中。

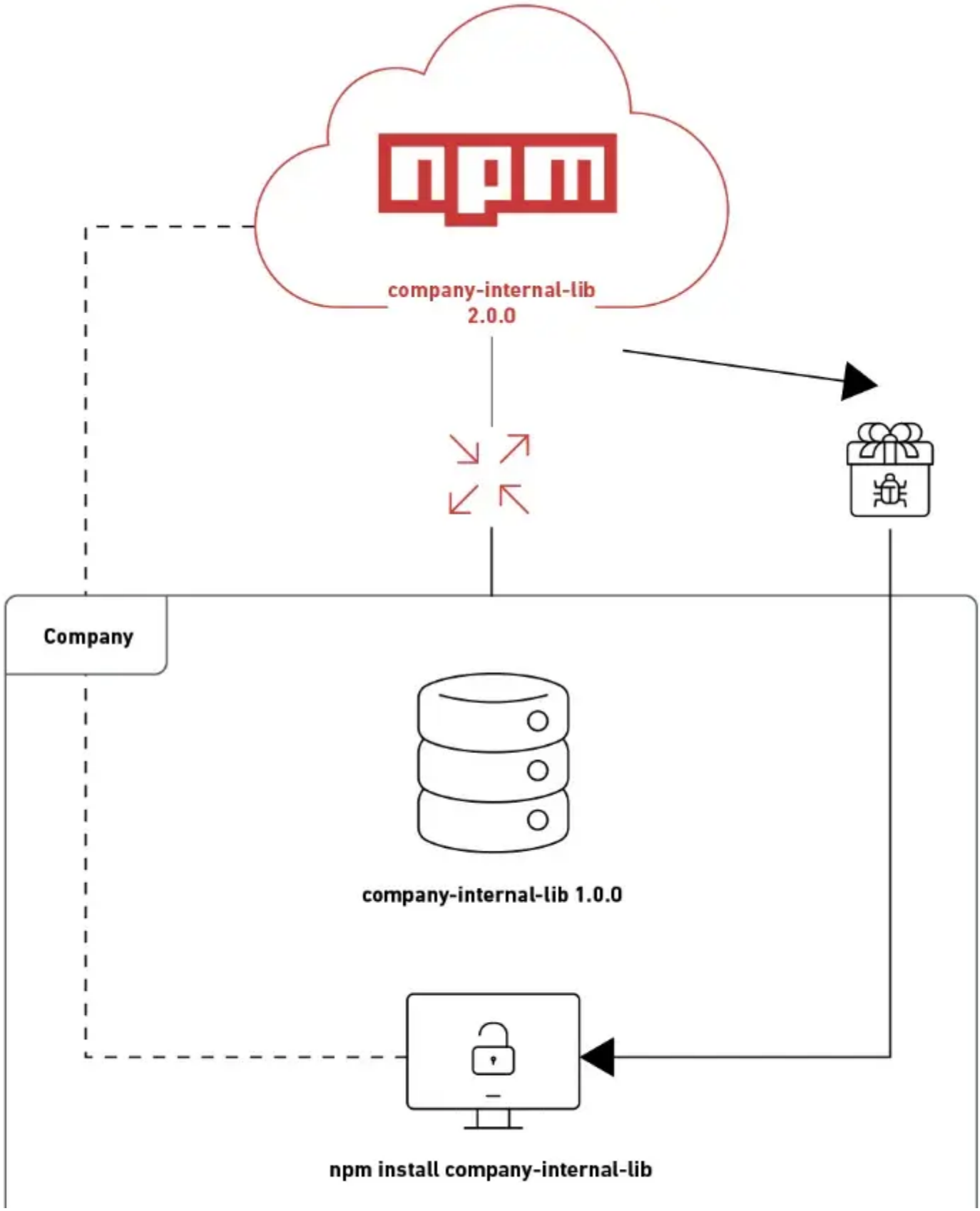
4. **安全配置错误攻击**



> 不恰当的设置、默认值和过时的配置往往会导致应用出现安全配置错误问题，从而使网络犯罪分子有机可乘。
>
> 例如，可能会出现以下情况：启用目录列表可能会泄露敏感信息、密码和仍是默认值的密钥，以及暴露错误处理信息。

如何防止安全配置错误问题?

1. 始终确保已更新服务的默认密钥和密码，并定期进行配置审计。
2. 定期审查安全设置也有助于降低可能存在安全配置错误或过时配置漏洞的风险。
3. 在配置相似的生产、开发和测试环境中使用不同的凭据进行自动构建和部署流程，也有助于保护你的应用。
4. **依赖性利用**



笔者：上图中公司内部和公共 npm 上都有名为 company-interal-lib 的库，如果你安装时不指定公司的源，你可能会安装 npm 上的包，这个包有安全漏洞。

前端应用由大量第三方库组成，这些库的使用使开发人员的工作变得更加轻松。但开发人员经常疏忽的一点是，这些库有时可能存在安全漏洞。

> 例如，Log4j 存在一个巨大的漏洞，攻击者可以在 Java 环境中执行远程代码。因此，任何使用 Log4j 的应用程序都会成为这种攻击的受害者！

如何防止依赖性利用?

使用广泛使用、维护得当、可靠并经过社区测试的库。

除此之外，定期审计、更新依赖关系和使用漏洞扫描工具也能确保前端应用的安全。

总结

确保你构建的 web 应用保持高度安全非常重要。这不仅仅是为了你的应用有良好的用户体验，更是为了确保用户数据的安全。

阅读完这篇文章后，我建议你检查一下你的应用代码，看看你的应用是否容易出现上述问题，如果有，请立即采取相关策略!

- 原文地址：Top 7 Common Frontend Security Attacks