---------------------------------------------------------------------

# 文档声明

## 合规性提示

本文档仅用于SSL VPN技术原理及应用场景的知识科普，不涉及任何翻墙工具的推荐、使用指导或技术破解内容。根据中国法律法规，未经许可私自使用VPN访问境外网络属于违法行为，相关使用需严格遵守国家网络安全管理规定及企业合规要求。

## 版权与使用说明

本文档所有内容（包括技术原理、分类说明、应用场景等）均为作者原创整理，仅供个人学习、研究参考使用，禁止用于任何商业用途（如付费课程、商业出版物、盈利性内容分发等），禁止以任何形式（包括完整转载、节选、截图、二次编辑后转发等）在其他平台传播。任何获取、使用本文档的用户，均需完整保留作者信息及本声明内容，不得抹去或修改原创标识。

## 内容免责

本文档内容仅为技术知识梳理，不构成专业指导建议。因引用本文档内容导致的任何直接或间接后果（包括但不限于信息误差、合规风险、第三方投诉等），作者不承担法律责任。文档中提及的技术方案（如SSL VPN协议、实现流程）仅作原理说明，具体使用需结合当地法律法规及企业内部规范。

---------------------------------------------------------------------

# 1 SSL VPN工作流程

SSL VPN的工作过程是一个包含“发起访问→安全握手→身份认证→隧道建立→数据传输→会话终止”的完整流程，涉及用户设备、SSL VPN网关、企业内部资源服务器等多个角色，核心是通过TLS协议构建加密隧道并实现精细化资源访问。

## 阶段 1：准备阶段

用户需要访问企业内部资源（如OA系统、文件服务器）时，首先通过两种方式发起访问：

**浏览器方式**：在浏览器中输入SSL VPN网关的公网地址，依赖浏览器内置的 TLS协议栈，无需安装额外客户端。

**专用客户端方式**：通过企业部署的SSL VPN客户端发起连接，客户端会主动向网关发起请求。

此时，用户设备通过DNS解析获取SSL VPN网关的公网IP地址，建立与网关的TCP连接（通常使用443端口，与HTTPS默认端口一致，可规避部分网络防火墙限制）。

## 阶段 2：TLS 握手

TLS握手是SSL VPN安全的基础，目的是让用户设备（客户端）与SSL VPN网关协商加密规则、验证网关身份，并生成后续数据传输的对称会话密钥。现代 SSL VPN均使用TLS协议（SSL正在逐步淘汰），以TLS 1.3为例，握手过程如下：

1.客户端Hello（Client Hello）

用户设备向网关发送消息，包含：

支持的TLS版本（如TLS 1.3）；

支持的加密套件列表（如TLS\_AES\_256\_GCM\_SHA384，包含密钥交换算法、对称加密算法、哈希算法）；

一个随机数（Client Random），用于后续密钥生成；

会话ID（若需复用旧会话，可减少握手步骤）。

2.服务器Hello（Server Hello）

网关从客户端提供的选项中选择最优配置，回复消息包含：

确认使用的TLS版本和加密套件（如选定TLS\_AES\_256\_GCM\_SHA384）；

网关生成的随机数（Server Random）；

会话ID（确认是否复用会话）。

3.服务器证书与密钥交换

网关向客户端发送：

服务器证书：包含网关的公钥、证书颁发机构（CA）签名等信息，用于客户端验证网关身份（防止中间人伪造网关）。

密钥交换信息：若使用ECDHE（椭圆曲线 Diffie-Hellman 临时密钥交换），网关会发送“椭圆曲线公钥”和“签名”（用网关私钥对交换信息签名，客户端用证书中的公钥验证签名，确保密钥交换未被篡改）。

4.客户端验证网关身份

用户设备收到证书后，通过内置的根CA证书（如操作系统或浏览器预装的可信CA）验证证书有效性：

检查证书是否在有效期内；

检查证书签名是否由可信CA颁发；

检查证书中的域名是否与网关公网地址一致（防止域名劫持）。

若验证失败，浏览器会提示“证书错误”，阻断连接（核心防中间人攻击手段）。

### 5.客户端密钥交换与握手完成

客户端生成“椭圆曲线私钥”，结合网关的“椭圆曲线公钥”计算出“预主密钥（Pre-Master Secret）”；

客户端用Client Random、Server Random、Pre-Master Secret生成“主密钥（Master Secret）”，再衍生出会话密钥（用于后续数据加密的对称密钥，如AES密钥、HMAC密钥）；

客户端发送“Finished消息”（用会话密钥加密的握手摘要，证明已正确生成密钥）。

### 6.服务器确认握手完成

网关用同样的方式生成会话密钥，解密客户端的Finished消息，验证通过后，发送自己的“Finished消息”（用会话密钥加密）。

客户端解密并验证网关的Finished消息，至此TLS握手完成，双方已持有相同的会话密钥，后续数据将通过对称加密传输。

## 阶段 3：用户身份认证

TLS握手仅验证了网关的身份（防止连接到假网关），但网关还需确认用户是否有权访问，即用户身份认证。这一步是SSL VPN的“访问控制第一道关”。

### 1.网关推送认证页面/请求

握手完成后，网关向用户设备发送认证请求（如登录页面），提示输入身份凭证。

### 2.用户提交认证信息

用户输入凭证，常见方式包括：

单因素认证：用户名 + 密码（需通过TLS加密传输，防止明文泄露）；

双因素认证（2FA）：用户名 + 密码 + 动态口令（如OTP令牌、手机验证码）、用户名 + 密码 + 生物识别（指纹/面部）；

证书认证：用户设备安装个人证书，网关验证证书有效性（适合高安全场景，如金融、政务）。

### 3.网关验证身份

网关将用户凭证发送至企业认证服务器（如 LDAP、Active Directory、RADIUS 服务器）验证：

检查用户名密码是否匹配；

验证动态口令的时效性（如OTP是否在30秒有效期内）；

若为证书认证，检查用户证书是否由企业CA颁发、是否吊销。

### 4.授权资源访问范围

身份验证通过后，网关根据用户所属角色（如“普通员工”“管理员”）查询预配置的权限策略，确定该用户可访问的内部资源列表（如仅允许访问192.168.1.100的OA系统，禁止访问数据库服务器）。

## 阶段 4：资源访问与数据传输

身份认证和授权通过后，SSL VPN网关与用户设备之间建立“加密隧道”，用户开始访问内部资源，数据传输流程如下：

### 1.用户发起资源请求

用户在浏览器/客户端中请求具体资源（如访问http://oa.internal或下载共享文件），请求被封装后发送至SSL VPN网关（目标地址为网关，而非直接访问内部资源）。

### 2.网关代理转发请求

网关接收加密的请求后，用会话密钥解密，根据用户授权策略判断是否允许访问该资源：

若允许，网关作为“代理”，将请求转发至企业内部资源服务器（如OA服务器的内网IP 192.168.1.100）；

若不允许，网关返回“权限不足”提示。

### 3.内部资源响应处理

内部资源服务器处理请求后，将响应数据（如网页内容、文件数据）返回给 SSL VPN网关。

### 4.加密传输至用户设备

网关用会话密钥加密响应数据，通过隧道发送给用户设备；用户设备解密后，在浏览器/客户端中展示内容（如网页渲染、文件保存）。

注：整个过程中，用户设备与内部资源服务器不直接通信，所有数据均通过网关中转，且全程用TLS会话密钥加密，公共网络中仅能看到加密后的密文，无法窃取或篡改内容。

## 阶段 5：会话维护与终止

会话维护：SSL VPN网关会设置会话超时时间（如30分钟无操作），期间定期发送“心跳包”维持连接；用户可持续访问授权资源，无需重复认证。

会话终止：

主动终止：用户点击“断开连接”，网关与用户设备协商关闭TLS会话，销毁会话密钥；

被动终止：超时未操作、网络中断或网关检测到异常（如多次密码错误），强制关闭连接并清理会话信息。

# 2 SSL VPN相比于IPsec VPN

SSL VPN工作在传输层或者应用层，设计的目的是让远程用户安全的访问内网特定资源，因此，SSL VPN必须设置VPN网关进行代理转发以控制用户的行为。在整个过程中，内网服务器与用户均不能知晓对方的IP地址，由VPN网关进行代理双方的会话。更适合向用户提供服务。【我们常用VPN访问谷歌学术基本如此，使用SSL VPN进行通信】

IPsec VPN工作在网络层，其设计的核心目的是构建跨公网的虚拟局域网络，让不同网络能够在一逻辑网络上通信。在这个模式下，没有所谓的VPN网关，路由器仅转发数据包。类似局域网中主机间的直接通信，所以更适合需要整合不同私有网络的情景。（点到点实现网段虚拟互联通信实际上很依赖NAT技术，而且要注意直连路由器下没有对方的私有网段IP号，否则会触发混乱，要使用NAT技术或者修改网段设备IP。而实现网关到网关的虚拟互联通信将会大大减小NAT技术依赖）【点到点的连接侧重于设备工作，即设备自行获取NAT提供的IP并主动与对方建立连接，所有的加密、认证之类的让设备自己干，网关只负责转发。网关到网关的连接则侧重于路由器工作，即配置路由器间的IPsec VPN，让数据包以明文形式从设备上传至路由器，由路由器启动IPsec VPN工作】

IPsec VPN的隧道模式因为加密整个IP数据包使得其能够隐藏真实的私网IP，在公网上只能仅会显示VPN网关的IP【请注意，即便如此，VPN服务提供方依旧能够知道你的IP，你并不能实现完全“隐身”】。

SSL VPN的隧道模式也可以加密整个IP层数据包（非隧道模式加密应用层数据包），非隧道模式下，可以通过NAT技术、证书零知识隐匿性认证、URL改写（依赖于资源IP地址向URL的映射建立）等进行IP隐藏。

再次强调，任何IP隐藏技术都只是面向公网或其他网络的，并不是面向整个世界的，你的真实IP乃至MAC地址都会被路由器记录下来（路由器不只有内存也有储存）。

# 3 SSL/TLS协议

## 发展历史

SSL 1.0：1994年，由Netscape开发，为HTTP提供加密支持。由于设计缺陷未正式发布并使用。

SSL 2.0：1995年，首个商用版本，支持加密与身份认证。由于加密算法弱、未认证服务器证书、协议降级漏洞导致缺点明显。2011年被全面禁用。

SSL 3.0：1996年，引入SHA1和RSA密钥交换，将协议进行分层设计，支持灵活的加密套件组合。然而，在2014年发生的**POODLE攻击**彻底暴露CBC模式下的填充漏洞；**CRIME攻击**能够获取敏感数据，如Cookie；依赖的**RC4算法**存在统计偏差问题，容易泄露密钥进而破解明文。2015年被全面禁用。

TLS 1.0：基于SSL 3.0改进，增强算法灵活性，支持AES等更安全的对称加密算法；能够通过版本协商彻底阻止强制使用低版本协议的漏洞。然而，其**CBC模式下的IV复用**导致的密钥泄露问题依旧存在（这是CBC模式固有的问题）；依赖太短的哈希算法SHA1和MD5导致签名可伪造；**私钥一旦泄密就会导致历史所有通信被解密**。2020年后被全面禁止使用。（参考RFC 2246）

TLS 1.1：显示初始化向量IV，彻底解决CBC模式的填充问题，能够抵御填充语言攻击，禁用了DES等弱加密算法。然而**BEAST漏洞**仍未解决，需要结合TLS扩展FREAK防御；协议的复杂度太高，仍旧支持过时的加密套件。（参考RFC 4346）

TLS 1.2：强制使用AEAD加密（AES-GCM、ChaCha20-Poly1305等），完全弃用MD5、SHA1，仅支持SHA256及以上，使用ECDHE/RSA等密钥交换算法确保私钥泄露不影响历史通信。2016年，**DROWN攻击**利用服务器同时支持SSL 2.0与TLS 1.2时共享RSA私钥的漏洞破解了TLS连接；2020年，**Raccoon攻击**利用TLS 1.2握手流程的漏洞获取密钥。（参考RFC 5246）

TLS 1.3：简化握手流程1-RTT甚至0-RTT完成密钥交换，大幅度减小延迟，仅仅保留AES-GCM、ChaCha20等安全的算法，禁用RSA、DH等非**向前保密**的密钥交换算法。其兼容性较差，还面临着**RTT滥用**风险，攻击者可以利用泄露的会话票证伪造请求进行重放攻击。

## 与HTTP/HTTPS的关系

HTTPS是经历TCP握手、SSL/TSL握手再传输消息的，而HTTP仅仅只需要经历TCP握手就开始传输消息，这导致其消息以明文传输。

## 补充

### POODLE攻击详解

POODLE（Padding Oracle On Downgraded Legacy Encryption，降级加密中的填充预言攻击）是针对SSL 3.0协议的中间人攻击手段，利用协议在填充机制上的设计缺陷，使得攻击者能够窃取HTTPS通信里的敏感信息。

POODLE漏洞由Google安全团队于2014年发现，编号为CVE-2014-3566。当时的许多服务器为了兼容旧版客户端而保留SSL 3.0，攻击者通过将协议强制降级未SSL 3.0并利用该协议CBC加密模式下的缺陷实施攻击。该漏洞是高危漏洞，影响全球99.9%以上的网站。

破解一个字节至少需要256次请求，破解的过程大体如下：

1.协议降级：攻击者通过中间人拦截客户端与服务器的TLS握手过程，伪造服务器不支持TLS的响应来诱导客户端使用SSL 3.0。

2.CBC加密模式下的填充漏洞：由于SSL 3.0使用PKCS#5填充，那么使得攻击者可以以一种高效的方法破解明文。（请注意，在有MAC值校验时破解难度将会增大）

PKCS#5：

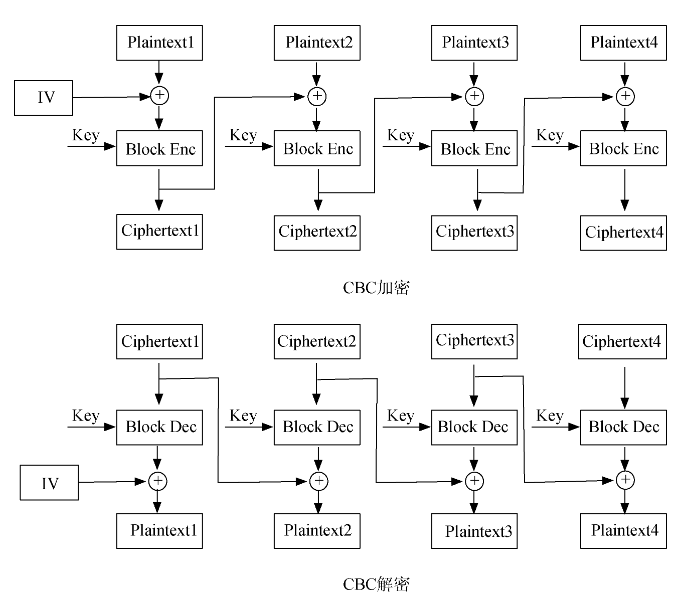
将明文填充至块大小的整数倍：当最后一个分组小于块大小时，剩余m个空字节就每个空字节用值为m来填充；当最后一个分组等于块大小时，直接填充一整个块，规则与之气前的相同。请注意PKCS#5规定块大小为8字节，PKCS#7支持大小为1-255字节的块。**PKCS#5要求检查整个块的填充是否正确.即最后一个字节是0x0p，就要检查最后的p个字节是否都为0x0p**。

服务器响应：

**我们先不考虑MAC校验问题**，只要填充检测不通过，就会返回填充失败；否则，一定不会返回填充失败。

CBC模式加密过程：

每个明文块都要和前面邻近的密文快做异或操作，即ci=mi xor ri，其中ri = Fk(ri-1)，r0 = IV，F是一个带密钥的加密模块，不同的算法有着不同的加密方式，相当于F拥有加密功能。F等价于Enc，F-1等价于Dec。加密过程与解密过程如下图所示：



破解方法（标准检测填充）：

假设一个明文与填充以及其密文为

M1 = m11m12m13m14m15m16m17m18 ，M2 =m210x070x070x070x070x070x070x07

C1 = c11c12c13c14c15c16c17c18  ,C2 = c21c22c23c24c25c26c27c28

IV = C0

1.记D2 = Deck(C2)，D1 = Deck(C1)。根据CBC解密过程，有M2 = D2 xor C1，M1 = D1 xor IV。D长度为8字节，记D[i]为D的第i个字节。需要注意的是，Dm[i]不等于对Cmi进行解密。

2.首先破击D2[8]：D2[8] = C18 xor M28。利用服务器响应，通过修改C18（遍历0x00至0xFF）使得服务器响应的不是填充错误，那么能够确定M28 = 0x01，此时由于已知C18就可以直接计算出D2[8]。

3.之后破解D2[7]：由2得到当M28 = 0x02时，可以一次就计算出C18现在的值。D2[7] = C17 xor M27。利用服务器响应，通过修改C17（遍历0x00至0xFF）使得服务器响应的不是填充错误，那么能够确定M27 = 0x02，此时由于已知C17就可以直接计算出D2[7]。

4.如此一直下去能够轻易得到整个D2，那么根据M2 = D2 xor C1获得M2。

**如此能够高效的破解含有填充的块。**对于无填充的块，可以构造成有填充的块，方法很简单，直接删掉已经被破解的块就行依旧依照如上的逻辑逐字节破解D1，并借助D1获得M1。细心的读者能够借此破解C1 C2 ... Cn了。

#### 破解方法（不严格检查填充）：

攻击需依赖原始明文的填充结构（如示例中P2[2-8] = 0x07）验证结果，而非多字节填充的一致性。

1.记D2 = Deck(C2)，D1 = Deck(C1)。根据CBC解密过程，有M2 = D2 xor C1，M1 = D1 xor IV。D长度为8字节，记D[i]为D的第i个字节。需要注意的是，Dm[i]不等于对Cmi进行解密。

2.首先破击D2[8]：D2[8] = C18 xor M28。利用服务器响应，通过修改C18（遍历0x00至0xFF）使得服务器响应的不是填充错误，那么我们得到两个集合，一个时填充的集合M28 bt {0x01 - 0x08}-number is 8，C18 bt {0x00 - 0xFF}-number is 8。直观上看，我们没有破解出D2[8]，但是我们忽略了一个事实：D2[8]是一个常数，那么我们计算合法填充M28与密文C18异或的值并记录映射关系，很容易发现八组一样的值，这个值就是是D2[8]，同时也知道了填充值0x07（很直观的知道，如果0x0p与c的异或是K，那么0x0q与c的异或就不可能是K。这六十四个结果中肯定会出现八个一样的值，那就是D2[8]，不可能再出现八个一样的值但不是D2[8]，结合前一句话和容斥原理即可证明）。

3.然后破解D2[2-7]：因为填充值是一样的，均为0x07，所以简单与密文异或就行。

4.接着破解D2[1]：如果填充值是0x08就不用这一步了。实际上，如果仅仅只校验最后一个字节是否处于0x01-0x08之间，就卡在这一步了，无法继续在纯数学的角度进行深入破解了。

#### 有MAC校验时的漏洞

以上两种破解方法都是没有MAC值校验的，在真实场景下是不现实的。

以下提供的环境可能没有MAC，但是已经是在公开平台找的最好的了。【大家有好的资源可以dis我，邮箱2035334606@qq.com，我会把资源挂在github上的】

真实场景的POODLE模拟，请点击[Padding Oracle Attack - CTF Wiki](https://ctf-wiki.org/crypto/blockcipher/mode/padding-oracle-attack/)、[Padding Oracle Attack(填充提示攻击)详解及验证 - 简书](https://www.jianshu.com/p/833582b2f560)或参考《Security flaws induced by CBC Padding》（路径为..Articles/assets/document/Security flaws induced by CBC Padding.pdf）

### CRIME攻击介绍

CRIME（Compression Ratio Info-leak Made Easy，压缩率信息泄露攻击）是 2012 年发现的针对 TLS 协议的中间人攻击手段，利用 TLS 压缩机制的设计缺陷窃取敏感信息（如会话 Cookie、认证令牌等）。

CRIME由Juliano Rizzo和Thai Duong在2012年披露（CVE-2012-4929），其核心在TLS压缩与加密顺序的缺陷。当数据在TLS层被压缩后再加密时，攻击者可通过观察压缩后数据的长度变化，逆向推断明文内容。例如，若攻击者能控制请求中的部分内容（如URL参数），并观察到压缩后密文长度的细微差异，即可逐字节破解敏感信息。

TLS 1.0/1.1协议允许在加密前对数据进行压缩（如DEFLATE算法），以减少传输带宽。但压缩过程会引入统计特征：若明文包含重复或可预测的内容（如 Cookie中的固定前缀），压缩后的长度会显著缩短。攻击者通过构造包含猜测内容的请求，对比压缩后密文长度，即可判断猜测是否正确。

### CBC模式下的IV复用介绍

若IV被复用（如两次加密使用相同的IV和密钥），则从开头开始的连续相同的明文块会生成相同的密文块，泄露明文之间的关联性。攻击者可通过观察密文块的重复，推断出明文的重复模式，尤其是在协议存在固定格式（如 HTTP 头、Cookie）时。攻击者可修改密文块并利用 IV 复用来伪造有效数据。

BEAST漏洞是2011年由安全研究员Moxie Marlinspike发现的针对SSL/TLS 协议的中间人攻击漏洞，主要影响使用CBC（密码块链）模式加密的HTTPS连接，其核心就是CBC加密模式下的IV复用。

### DROWN攻击介绍

DROWN的核心在于利用SSLv2的RSA加密缺陷作为旁路通道，攻击与之共享私钥的TLS连接。具体过程如下：

SSLv2的致命缺陷：

SSLv2协议在处理RSA加密时存在两个关键问题：

1.填充预言机（Padding Oracle）：SSLv2服务器在解密RSA密文时，若发现格式错误（如 PKCS#1 v1.5填充不正确），会返回不同的错误响应。攻击者可通过反复发送伪造的密文，利用这些响应逐步逼近正确的解密结果（即 Bleichenbacher攻击）。

2.无会话隔离：SSLv2服务器与TLS服务器若共享同一RSA私钥，攻击者可通过SSLv2连接获取的解密信息，直接破解TLS流量。

攻击流程：

1.攻击者先与目标服务器建 SSLv2连接，发送精心构造的密文，触发服务器的填充预言机响应。

2.通过多次交互，攻击者利用响应中的“有效/无效”反馈，逐步缩小可能的明文范围，最终恢复TLS会话密钥。

3.一旦获得密钥，攻击者即可解密TLS连接的所有数据，包括用户名、密码、信用卡信息等。

### Raccoon攻击介绍

Raccoon攻击是传输层安全性（TLS）协议中的一个定时漏洞攻击，利用TLS 1.2及更低版本加密协议中的旁通道，对TLS握手期间的Diffie - Hellman（DH）密钥交换过程采用定时攻击策略。由于TLS标准鼓励非恒定时间处理DH秘密，若服务器重用临时密钥，带有前导零的DH机密会使服务器KDF计算更快，导致服务器响应时间缩短。攻击者通过记录客户端和服务器之间的握手消息，发起新握手并测量服务器响应时间，识别出这种极端情况，进而破译原始握手的秘密密钥，最终解密TLS流量。

### RTT滥用介绍

**一、RTT 滥用的核心机制**

1. 协议缺陷与时序漏洞：

* TLS/SSL握手：在TLS 1.2及之前版本中，Diffie-Hellman密钥交换的握手过程存在时序差异。攻击者通过测量RTT波动，可推断密钥生成的中间状态（如 Raccoon 攻击）。
* QUIC协议：基于UDP的QUIC协议允许0-RTT握手，但攻击者可通过伪造源IP发起反射放大攻击，利用RTT优化攻击效率（单包放大率达100倍）。

2. 隐蔽信道与数据泄露：

* RTP协议隐蔽信道：攻击者通过修改 RTP 头部的时间戳字段，以 350 bps 的速率传输秘密数据。由于时间戳的自然波动，此类攻击极难检测。
* 跨层RTT差异：代理流量中，传输层（客户端 - 代理）与应用层（客户端 - 服务器）的 TT差异可被攻击者利用，通过统计分析识别代理存在。

3. 拥塞控制操纵：

* BBR算法的 RTT不公平性：在TCP BBR拥塞控制中，长RTT流会抢占更多带宽。攻击者通过人为增加RTT（如引入延迟节点），可窃取90%以上的链路带宽，导致其他流量被饿死。

**二、典型攻击场景与案例**

1. DDoS攻击中的RTT优化

* 反射放大攻击：攻击者伪造源IP向DNS服务器发送查询请求，利用RTT短的服务器作为反射器，使目标服务器收到百倍于请求量的响应流量。例如，2023年某短视频平台遭受的QUIC反射攻击峰值达60Gbps。
* ICMP洪水攻击：发送大量ICMP Echo请求（ping），通过测量RTT筛选响应快的节点集中攻击。2022年俄罗斯RT电视台网站因DDoS攻击瘫痪，27%的攻击流量来自美国。

2. 网络拓扑探测与渗透

* DNS负载均衡破坏：攻击者通过发送恶意域名解析请求，使 DNS 服务器对无响应节点进行惩罚（RTT值重置），导致负载均衡失效。2025年某金融机构因此类攻击导致DNS解析延迟增加500%。
* 无线 Mesh网络虫洞攻击：恶意节点通过伪造短RTT路径建立隧道，诱骗数据流量经过其控制的节点，窃取敏感信息。NS-3仿真显示，此类攻击可使网络吞吐量下降70%。

3. 加密通信破解

* TLS 1.3 0-RTT重放攻击：攻击者拦截0-RTT握手消息并重复发送，导致支付接口重复扣款或用户会话劫持。2025年某电商平台通过添加一次性令牌（有效期 5 秒）将重放攻击拦截率提升至99.2%。
* 量子侧信道攻击：利用量子计算的高精度时序测量，通过分析TLS握手的RTT波动，可在10分钟内破解ECDHE-256密钥。

### 向前保密介绍

要求即使长期的私钥泄露也不会暴露历史通信。

前提是要有一把由CA签过名的公钥证书，证明这把公钥是可信的。

对于会话时交换临时密钥，服务器端需要使用CA签过名的公钥相适配的私钥对密钥交换算法里的临时公钥签名。这样中间人即使能够伪造成客户端却不能伪造成服务器端，因为CA证书是不可逆的，即攻击者没办法获得CA的私钥（CA签名用私钥，验证证书真假用公钥）。即使服务器的长期私钥泄露了，也不影响历史通信，除非攻击者能够逆向破解签名算法（离散对数难题，目前计算复杂度是亚指数级的，除非用量子计算机，否则无法在可接受时间内破解，或者找到更先进的算法，当然，这样攻击者就创造历史了）

**本文章归属 github 用户 WhatTheFuck-cyber**