计算机安全的“坏态度”指南

Keith Winstein

Stanford University

https://cs.stanford.edu/~keithw

“坏态度”的建议

1.
2.
3.
4.
5.
6.
7.
8.

“坏态度”的建议

1. 一定要自己构建加密协议。
2.
3.
4.
5.
6.
7.
8.

Mosh（移动 shell）

(图片幻灯片)

Mosh 部署

动机: 适用于差 Wi-Fi 的 SSH

+ 间歇性连接

+ 漫游

+ 本地回显

+ 防伪造 RST 的安全性

首次发布: 2012年

今天: 约 200万 - 2000万用户

Mosh 协议

每个数据报用 AES-OCB 包装

每个数据报代表幂等操作

没有 TLS, 没有 DTLS, 没有公钥密码

没有时间戳, 没有重放缓存, 没有守护进程

没有密码协商, 没有文件 IO, 没有 root

漫游: 服务器回复编号最大的真实传入数据报的源地址

Hacker News

“从外表看是一个人的乐队...实现了自己的私有加密协议 (有没有经过重放攻击、填充攻击等方面的审查? [插入 20 年来困扰计算机科学最伟大头脑的费解难题]”

Slashdot

“欢迎来到又一个由学者设计的协议，这些学者二十年来（如果曾经有的话）从未接近过真实的网络。”

Twitter

Dan Kaminsky: “Mosh 在 SSH Transport 之外运行使得学术性能代码未经认证...喜欢 Mosh, 如果它在 SSH 的通道内运行会更喜欢。”

Q: “有什么特别的原因吗? 由于从丢包中快速恢复是其主要目标之一, 需要 UDP+OCB。”

Kaminsky: “构建新的安全通道很*棘手*。看看 DTLS 漫长而痛苦的开发周期。”

Twitter（续）

Moxie Marlinspike: “我不知道, 从语义安全的角度来看, 很难比 SSH 更差了。它在许多方面比 TLS 更差。”

Kaminsky: “你怀疑它有像 BEAST 那样的漏洞等待被发现吗?”

Moxie: “已经发现了。CBC 密码套件现在完全不能用了, 因为选定密文攻击。更糟糕的是...就像 TLS 一样。坏协议在某些情况下勉强过关。慢慢把自己逼入死胡同。”

Mosh 生命周期内的安全漏洞

TLS:

- goto fail (Secure Transport)
- GnuTLS verify (GnuTLS)
- Heartbleed (OpenSSL)
- Lucky Thirteen
- BEAST
- CRIME
- POODLE
- FREAK
- Logjam
- 2013 RC4 attacks

SSH:

- memory corruption attack
- X11 trust race condition
- weak tty permissions
- password limit circumvention
- root password auth bug
- unfinished roaming feature allows private key extraction
- command injection to xauth

Mosh:

- (目前为止没有已知的安全漏洞)

教训

委员会是最糟糕的。

小项目有巨大的优势。

Bug 是由功能引起的。功能越少, Bug 越少。

Feynman 式的格言: 你不如大项目的最佳贡献者, 但你可能比平均贡献者更好。

对于安全性, 最差的贡献者可能才是关键。

我真的（大部分）相信的建议

经过 20 年的委员会设计, SSL/TLS 及其实现如此复杂和充满 bug, 以至于对于特定的专注任务, “自己动手”有时可能是更合理的路径, 即使 TLS 能完成工作。

“坏态度”的建议

1. 一定要自己构建加密协议。
2.
3.
4.
5.
6.
7.
8.

“坏态度”的建议

1. 一定要自己构建加密协议。
2.
3.
4.
5.
6.
7.
8.

“坏态度”的建议

1. 一定要自己构建加密协议。
2. “安全”语言并不那么安全。
3.
4.
5.
6.
7.
8.

Hacker News

“任何在 2016 年用 C 语言做安全工作的人, 在我看来都是在犯职业过失, 将用户置于风险之中, 因为他们的自我无法接受不亲手操作比特。”

内存安全能防止的安全漏洞

TLS:

- goto fail (Secure Transport) — 未防止
- GnuTLS verify (GnuTLS) — 未防止
- Heartbleed (OpenSSL) — 已防止
- Lucky Thirteen — 未防止
- BEAST — 未防止
- CRIME — 未防止
- POODLE — 未防止
- FREAK — 未防止
- Logjam — 未防止
- 2013 RC4 attacks — 未防止

SSH:

- memory corruption attack — 已防止
- X11 trust race condition — 未防止
- weak tty permissions — 未防止
- password limit circumvention — 未防止
- root password auth bug — 未防止
- roaming stub allows private key extraction — 已防止
- command injection to xauth — 未防止

内存安全能防止的安全漏洞

(与上一页相同, 带标注)

流行的内存安全语言太强大了

你说的: “内存安全”

你的意思是: “Haskell”

别人听到的是: “JavaScript, Python, 或 Ruby on Rails”

任何有 eval 的语言显然非常想使用它。

Java 的安全记录也不太好。

“坏态度”的建议

1. 一定要自己构建加密协议。
2. “安全”语言并不那么安全。
3.
4.
5.
6.
7.
8.

“坏态度”的建议

1. 一定要自己构建加密协议。
2. “安全”语言并不那么安全。
3.
4.
5.
6.
7.
8.

“坏态度”的建议

1. 一定要自己构建加密协议。
2. “安全”语言并不那么安全。
3. HTTPS 是坏的。
4.
5.
6.
7.
8.

HTTPS 的神圣承诺

存在一家来自 173 家公司名单的公司, 该公司曾经证明我 URL 栏中域名的 WHOIS 联系电子邮件属于现在控制我正在交谈的服务器的同一个人 (除非其证书被盗)。

HTTPS 的神圣承诺

(与上一页相同)

HTTPS 的承诺相当薄弱

真正应该验证的是文件作者, 而不仅仅是服务器维护者的身份。

下载后, 无法验证文件, 没有谁证明服务器身份的记录, 也没有证据。

甚至破坏了自愿缓存和病毒扫描。

保证由 173 家半可靠的公司中的任何一家提供。

Pinning 来救场

浏览器厂商: 使用 HSTS 头来固定特定的服务器证书!

网站: 听起来像 TOFU。如果在用户第一次访问我们之前, CA 发出了一个恶意证书怎么办?

厂商: 请求 Google、Apple、Microsoft 和 Mozilla 将你的服务器证书直接硬编码到浏览器中!

Pinning II

网站: 我们做到了。但是, 我们的一些用户在防火墙后面, 似乎仍然在对他们的会话进行中间人攻击。

厂商: 该 pin 仅对“公共”根 CA 有效。如果用户安装了“私有”根 CA, 它将覆盖 pin, 即使你的证书已被硬编码到浏览器中。

网站: 为什么浏览器会明知故犯地允许中间人攻击?

厂商: 许多公司使用病毒扫描中间件, 而在 HTTPS 下做到这一点的唯一方法就是完全地进行中间人攻击。遵守 pin 可能是正确的做法, 但我们的浏览器会被认为坏了, 我们会失去市场份额。我们只会在竞争对手已经这样做之后才这样做。

Pinning III

网站: 在忽视 pin 时, 至少显示一个坏锁和警告: “您的会话正被私有权威窃听。点击此处禁用。”

厂商: 如果我们这样做, 就必须永远显示警告, 因为这些资源会进入缓存并永久损坏它。所以这不是一个很有用的警告。

网站: 如果缓存“永久损坏”了, 你应该永远显示警告!

厂商: 那样我们的浏览器会被认为坏了, 我们会失去市场份额。我们只会在其他厂商之后这样做。

网站: 你能至少为偏执的人提供这个选项吗?

厂商: lol no

“坏态度”的建议

1. 一定要自己构建加密协议。
2. “安全”语言并不那么安全。
3. HTTPS 是坏的。
4.
5.
6.
7.
8.

“坏态度”的建议

1. 一定要自己构建加密协议。
2. “安全”语言并不那么安全。
3. HTTPS 是坏的。
4.
5.
6.
7.
8.

“坏态度”的建议

1. 一定要自己构建加密协议。
2. “安全”语言并不那么安全。
3. HTTPS 是坏的。
4. 密码哈希是坏的。
5.
6.
7.
8.

密码哈希是坏的

密码哈希是坏的, 因为它让你认为用户向你发送密码是可以的。

用户不应该向你发送他们的密码。

你的网站安全不应该依赖于你对密码复杂性要求的执行。

你不希望服务器被入侵时暴露任何允许坏人拦截或破解用户密码的东西。

更好的做法：委托。使用公钥认证，或“使用 Google 登录” / “使用 Facebook 登录” / OpenID Connect。

“坏态度”的建议

1. 一定要自己构建加密协议。
2. “安全”语言并不那么安全。
3. HTTPS 是坏的。
4. 密码哈希是坏的。
5.
6.
7.
8.

“坏态度”的建议

1. 一定要自己构建加密协议。
2. “安全”语言并不那么安全。
3. HTTPS 是坏的。
4. 密码哈希是坏的。
5.
6.
7.
8.

“坏态度”的建议

1. 一定要自己构建加密协议。
2. “安全”语言并不那么安全。
3. HTTPS 是坏的。
4. 密码哈希是坏的。
5. 前向保密通常是秘密的, 这是坏的。
6.
7.
8.

前向保密的定义

前向保密（Forward secrecy）

在多方之间的通信中, 在时间 t 存在一个“棘轮”, 在稍后的时间 u 存在一个“删除事件”。

前向保密: 在时间 u 之后某方暴露秘密材料, 不会帮助窃听者解码在时间 t 之前编码的密文。

(通俗地说, t 可能是会话结束时, u 是所有方都已经擦除了保护该会话的密钥或任何可以派生该密钥的东西时。)

问题：无法通信 u 何时发生

网站应该每天擦除其密钥缓存。

客户端如何了解这是否已经发生? 没有办法询问。

TLS 1.3 草案包含异步密钥轮换, 但没有经过认证的方式来确认消息。

我的观点: 如果你关心 PFS, 你应该想要经过认证的 PFS。任何值得做的操作都值得确认, 包括密钥轮换。

“坏态度”的建议

1. 一定要自己构建加密协议。
2. “安全”语言并不那么安全。
3. HTTPS 是坏的。
4. 密码哈希是坏的。
5. 前向保密通常是秘密的, 这是坏的。
6.
7.
8.

“坏态度”的建议

1. 一定要自己构建加密协议。
2. “安全”语言并不那么安全。
3. HTTPS 是坏的。
4. 密码哈希是坏的。
5. 前向保密通常是秘密的, 这是坏的。
6.
7.
8.

“坏态度”的建议

1. 一定要自己构建加密协议。
2. “安全”语言并不那么安全。
3. HTTPS 是坏的。
4. 密码哈希是坏的。
5. 前向保密通常是秘密的, 这是坏的。
6. 端到端安全是坏的, 密钥托管是好的。
7.
8.

端到端安全是坏的

我曾经相信端到端安全。

但是, 我曾经认为自己是端点。

现在我拥有的端点通过 Wi-Fi 或它们自己的 LTE 与其制造商安全地通信。

我应该有权窃听我自己的设备关于我所说的内容。

端到端安全是坏的（续）

今天, 端到端安全意味着端点是唯一能保卫自己的东西。

如果你只能看到密文, 很难提供纵深防御或甚至检测攻击。

每个廉价设备都是单点故障。

制造商不会为一个 10 美元的设备持续提供安全补丁。

提议的研究议程

我们如何为加密通信构建防火墙和审计器, 以避免单点故障?

(示例方法: Blindbox、延迟密钥释放、只读密钥)

“坏态度”的建议

1. 一定要自己构建加密协议。
2. “安全”语言并不那么安全。
3. HTTPS 是坏的。
4. 密码哈希是坏的。
5. 前向保密通常是秘密的, 这是坏的。
6. 端到端安全是坏的, 密钥托管是好的。
7.
8.

“坏态度”的建议

1. 一定要自己构建加密协议。
2. “安全”语言并不那么安全。
3. HTTPS 是坏的。
4. 密码哈希是坏的。
5. 前向保密通常是秘密的, 这是坏的。
6. 端到端安全是坏的, 密钥托管是好的。
7.
8.

“坏态度”的建议

1. 一定要自己构建加密协议。
2. “安全”语言并不那么安全。
3. HTTPS 是坏的。
4. 密码哈希是坏的。
5. 前向保密通常是秘密的, 这是坏的。
6. 端到端安全是坏的, 密钥托管是好的。
7. Snowden 文件不应该改变我们的行为。
8.

GNU emacs, 1987年5月

```
;;; spook.el --- spook phrase utility
;;; for overloading the NSA line eater
;; Just before sending mail, do M-x spook.
;; A number of phrases will be inserted
;; into your buffer, to help give your
;; message that extra bit of attractiveness
;; for automated keyword scanners.
;; Help defeat the NSA trunk trawler!
```

欧洲联盟，2001年7月

(图片幻灯片)

布什政府赢得首次 FISA 上诉，2002年

(图片幻灯片)

纽约时报，2005年12月

(图片幻灯片)

AT&T；举报人，2006年4月

(图片幻灯片)

USA Today, 2006年5月

(图片幻灯片)

华尔街日报，2006年6月

(图片幻灯片)

FISA 修正案，2008年7月

(图片幻灯片)

Obama 支持 FAA, Hillary Clinton 反对

(图片幻灯片)

Snowden 文件, 2013年10月

(图片幻灯片)

“Google 已开始加密”， 2013年11月

(图片幻灯片)

Microsoft 正在寻求加密，2013年11月

(图片幻灯片)

“坏态度”的建议

1. 一定要自己构建加密协议。
2. “安全”语言并不那么安全。
3. HTTPS 是坏的。
4. 密码哈希是坏的。
5. 前向保密通常是秘密的, 这是坏的。
6. 端到端安全是坏的, 密钥托管是好的。
7. Snowden 文件不应该改变我们的行为。
8.

“坏态度”的建议

1. 一定要自己构建加密协议。
2. “安全”语言并不那么安全。
3. HTTPS 是坏的。
4. 密码哈希是坏的。
5. 前向保密通常是秘密的, 这是坏的。
6. 端到端安全是坏的, 密钥托管是好的。
7. Snowden 文件不应该改变我们的行为。
8.

“坏态度”的建议

1. 一定要自己构建加密协议。
2. “安全”语言并不那么安全。
3. HTTPS 是坏的。
4. 密码哈希是坏的。
5. 前向保密通常是秘密的, 这是坏的。
6. 端到端安全是坏的, 密钥托管是好的。
7. Snowden 文件不应该改变我们的行为。
8. 未经同意的自动更新是坏的, 因为我们可能变坏。

nytimes.com, 2016年2月

(图片幻灯片)

这不是值得祝贺的事情

没有人会因为制作了一个“连他们自己都无法入侵”的系统而祝贺 SSH、GPG、OpenSSL、Apache 或 Mosh 的制作者。他们也不应该。

好的设计 = 连设计者都没有特殊访问权限。

当你保留在未经同意或公开审查的情况下自动更新用户软件的能力时, 你就成为了攻击面的一部分。

尊重用户的知情同意。

计算机安全的“坏态度”指南

1. 一定要自己构建加密协议。
2. “安全”语言并不那么安全。
3. HTTPS 是坏的。
4. 密码哈希是坏的。
5. 前向保密通常是秘密的, 这是坏的。
6. 端到端安全是坏的, 密钥托管是好的。
7. Snowden 文件不应该改变我们的行为。
8. 未经同意的自动更新是坏的, 因为我们可能变坏。

Keith Winstein

https://cs.stanford.edu/~keithw