我们一般认为 只要51%的挖矿算力是诚实的 比特币系统就是安全的

So, Bitcoin is often thought to be secure as long as 51% of the mining power is honest.

但这需要假定所有参与方都可以看到所有有效的区块和交易

But this assumes that all parties see all valid blocks and transactions.

比特币依赖于点对点网络传输信息

Bitcoin relies on its peer-to-peer network to deliver this information.

因此 如果你能控制点对点网络 你就可以控制比特币系统的信息流 从而控制区块链

So, if you control the peer-to-peer network, you can control the flow of information within Bitcoin, and thereby control the blockchain.

本次讲座中 我们将通过日蚀点对点网络中的通信信息来破坏比特币系统的安全性

In this talk, we're going to attack the peer-to-peer network and use information eclipsing to subvert Bitcoin security.

首先 我会为大家讲解什么是日蚀攻击 可以利用日蚀攻击达到何种攻击效果

So, first I'm going to tell you what an Eclipse attack is, and the bad things you can do with an Eclipse attack.

随后 我会为大家讲解如何实施日蚀攻击

Then I'm going to show you how you can perform an Eclipse attack.

如何将自己放置在实施日蚀攻击的位置上 以及实施日蚀攻击所需的必要资源

How you can get into position to perform an Eclipse attack on a peer-to-peer network, and the resources necessary to do this.

最后 我会讨论抵御日蚀攻击的方法 以及防御方法在比特币系统中的部署情况

And then finally, I'll discuss our countermeasures and their deployment in the Bitcoin ecosystem.

在讲解什么是日蚀攻击前 我需要简单向大家解释一下比特币系统的点对点网络

So, before I tell you what an Eclipse attack is, I have to explain a little bit about the Bitcoin peer-to-peer network.

点对点网络中包含很多节点 这里我们先主要关注紫色的节点

So, these are nodes in the peer-to-peer network. And we're going to be looking at the purple node.

默认情况下 节点可以向外创建8个TCP连接

And by default, nodes make 8 outgoing TCP connections.

箭头所指的方向表示TCP连接的方向

The direction of the arrow is the direction of the TCP connection.

同时 节点可以接受最多117个对内的TCP连接

And there are nodes allow up to 117 incoming TCP connections.

节点通过这些连接获取交易和区块 其中区块是交易的特定聚合形式

These connections, these edges are used to gossip transactions and blocks, where blocks are a special type of aggregation of transactions.

我用这种蓝色方块图标表示区块

And for blocks, I'm using this blue square icon.

我们可以看到 某个节点发现了一个区块 随后它会将区块发送到整个网络中

And so, you can see someone discovers a block, and they spread it through the network.

区块会通过这些边所代表的连接进行传递 传递是双向的

And the blocks flow across these edges. And they can flow in both directions.

我们的攻击只关注接受对内TCP连接的节点

So, our attack is only going to look at nodes that accept incoming connections.

并非所有节点都接受对内TCP连接

Not all nodes accept incoming connections.

一些钱包所关联的节点不会接受对内的连接

Some wallets, some nodes behind that do not accept incoming connections.

什么是日蚀攻击？

So, what is an Eclipse attack?

日蚀指的是信息日蚀 如果你可以控制点对点网络中节点所获取的信息 就成为日蚀

Well, the information eclipsing, if you manage to gain control over a node in a peer-to-peer network, access to information, you have eclipsed it.

例如 如果与紫色节点建立连接的节点都是发动合谋攻击的节点

So, for example, if the purple nodes, all of their connections were to attackers that sort of conspired against it,

那么这些攻击节点就可以阻止紫色节点获取全网的区块信息了

they could prevent it from learning about that block being sent around.

攻击节点也可以合谋告知紫色节点的一些信息 同时不把这些信息告知到全网

And they could also tell the things which it couldn't tell any other part of the network.

一般来说 攻击者一般不会处在实施攻击的位置上

And so, normally attackers are not in this position.

我们后面会为大家讲解 攻击者如何站在实施攻击的位置上

So, what I'm going to talk about later is how attackers can get into this position.

在此之前 我想解释一下 如果节点被日蚀 此会发生什么严重的后果

But before I do that, I want to explain why this is so bad once a node has been eclipsed.

我们可以把攻击示意图重新绘制一下

And we can redraw that picture.

攻击者位于紫色节点和剩余网络的中间位置

So, the attacker is sitting in the middle between the purple node and the rest of the network.

但这不意味着攻击者必须位于通信路径连接处

But this doesn't mean the attacker is an on-path attacker.

这只意味着与紫色节点建立连接的节点都是攻击节点

It just means that all of the purple nodes connections are to the attacker.

现在 攻击者可以决定紫色节点所能接收到的信息了

So, the attacker can decide what the purple node sees.

如果我们可以日蚀一个节点 会发生什么问题？

So, what bad things can you do if you eclipse a node?

会发生的问题是 攻击者可以用少于50%的算力实施50%攻击

Well, you can perform a 51% attack with less than 50% of the mining powder.

在幻灯片的例子中 我们有一个拥有40%算力的攻击者

So, in this case, we have an attacker that is 40% of the mining power.

蓝色节点也是一个旷工 拥有30%的算力

The purple node is also a miner with 30%.

网络中的其余节点总计有30%的算力 我们用一个云图表示网络中的其余节点

And the rest of the network has 30%. The rest of network is drawn by this cloud.

我们在左上角给出每一个参与方所看到的区块链

And we're going to represent each party's view of the blockchain up here.

攻击还没有开始 所以每一个参与方看到的都是蓝色的区块

The attack hasn't started yet. So, they all see the blue block.

攻击者要做的第一件事情就是把网络划分为两部分

The first thing the attacker does is it partitions the network

使得两个参与方相互之间无法收到对方生成的新区块

so that the two other parties can't build on each other's blocks.

这样一来 攻击者就可以独立与两部分参与方竞争区块生成速度了

The attacker can then outcompete each miner individually.

由于攻击者有40%的挖矿算力 因此攻击者可以生成比紫色节点更长的区块

So, because the attacker is 40% of the mining power, it'll generate a longer block chain than the purple node.

比特币全网总接受最长的区块

Bitcoin always uses the longest blockchain.

攻击者也可以用相同的方法对另一部分网络实施攻击

And it can perform the same attack against the rest of the network.

这样一来 攻击者生成的区块链会得到全网的共识

So, the attacker’s blockchain will become the consensus chain.

此攻击与51%攻击等价 因为攻击保证只有攻击者生成的区块才能被添加到区块链中

Oh, this is equivalent to a 51% attack, because the attacker can ensure only the attackers blocks are added to the blockchain.

因此 攻击者有能力选择添加至区块链中的交易 使得整个比特币停止运行

So, the attacker could choose not to include any transactions in the blockchain. And Bitcoin would just stop working.

攻击者还可以对历史区块链进行篡改 还可以实现其它一系列攻击效果

Or the attacker could use this to go undo history in the blockchain, and lots of other bad things.

此攻击实例中 我们假设攻击者不仅拥有日蚀能力 还拥有挖矿算力

Now, in this attack, we assumed the attacker not only had eclipsing power, but also mining power.

当攻击者拥有挖矿算力时 攻击者可以进一步实现其它攻击目的

There are some other attacks that can be improved with mining power.

例如 我们可以改进自私挖矿攻击 这里我就不详细讲解改进方式了

For example, we can do an improved selfish mining attack. But I'm not going to talk about that here.

现在 我们先看看 当攻击者没有任何挖矿算力时 可以完成何种攻击目标

So now, let's look at a situation in which the attacker does not have any mining power.

我们在攻击演示图中增加一个商人

And we've added a merchant to this picture.

紫色节点和商人可以看到相同的区块链

And the purple miners are going to share the same view of the blockchain.

现在 攻击者想实现双重花费

Now, the attacker wants to do a double spend.

这里我们假定紫色节点拥有30%的算力 网络中的剩余节点拥有70%的算力

So, yes, the purple node has 30% of the mining power. The rest of the network has 70%.

攻击者要对一笔交易实现双重花费

So, the attacker double spends a transaction.

攻击者在左侧的交易中将COIN\_0支付给了商人

This transaction gives COIN\_0 to the merchant.

而另一个双重花费交易将COIN\_0返还给了攻击者

And then, there's a separate double spend transaction that gives COIN\_0 back to the attacker.

攻击者将网络划分成了两个部分

The attacker has partitioned the network.

由于旷工和商人看到的是相同的区块链

And because both the miner and the merchant see the same thing,

旷工将此交易包含在区块中 并在此区块上挖矿

the miner includes the this transaction in a block, and then confirms it and mines blocks on top of that.

网络剩余节点也会完成相同的操作

And the same thing happens with the rest of the network.

网络剩余节点看到双重花费交易 将交易包含在区块中 尝试创建此区块

The rest of the network sees the transaction, includes it, and starts building blocks.

但是这两个参与方并不知道网络中还存在另一个区块链

But neither side knows that the other blockchain exists.

因此 由于攻击者将网络划分成两部分 商人完全没办法知道交易被双重花费了

So, the merchant has no idea that the transaction has been double spent due to the partition.

同时 商人也完全不知道网络中还存在一个更长的区块链

And the merchant has no idea that there is an existing longer blockchain.

因此 商人会把货品发送给攻击者

So, the merchant releases its goods to the attacker.

接下来 攻击者移除其设置的中继节点

And then, the attacker starts relaying blocks again.

一瞬间 商人看到的区块链就不合法了 此区块链会被全网移除

And all of a sudden, this blockchain is no longer valid. It's thrown away.

支付给商人的交易会被全网移除 攻击者重新得到了已经支付的数字货币

The transaction to the merchants is thrown away, and the attacker gets the money back.

此攻击过程中 商人的确看到交易已经被确认 图中显示交易已经被三个区块确认

So, the merchant in fact did see a transaction confirmed, in this example by three blocks.

因此商人会认为此交易已经是安全的了

So, the merchants thought it was safe.

但由于日蚀攻击 商人并不能意识到此交易仍然不安全

But because of the eclipsing, the merchant was unaware that it was unsafe.

这样一来 攻击者有能力实施双重花费攻击

And the attacker was able to perform double spending.

此攻击者没有任何挖矿算力 它利用了第三方提供的挖矿算力

So, this attacker didn't have any mining power. It cooperate a third party's mining power.

在我们的论文中 我们给出了其它一些挖矿算力日蚀攻击

And in our paper, we have some other new mining power eclipsing attacks.

现在 我将会为大家解释如何实施此攻击

So now, I'm going to explain how you get in this position to perform this attack.

一般来说 当攻击者加入网络 想向紫色节点发起攻击时

So, normally an attacker that joins the network to attack the purple node

攻击者不一定能和紫色节点建立连接

is not necessarily going to be connected to by the purple node.

因此 攻击者需要对紫色节点进行操控 使得紫色节点可以与攻击者建立对外连接

So, the attacker needs to manipulate the purple node such that the purple node will make outgoing connections to it.

攻击者需要将紫色节点存储的节点表替换为攻击者的IP地址

So, the attacker fills the purple nodes peer tables with attacker IPs.

我会在下一页幻灯片中解释节点表的工作原理

And I'll explain how peer tables work in the next slide.

当节点重启时 节点会丢失其建立的所有对外连接

The node then restarts and loses its outgoing connections.

此时 节点会从节点表中读取IP地址 并建立新的连接

And then, when the node goes to draw on its peer tables to make new connections.

由于节点表中的所有IP地址都是攻击者的IP地址 因此节点会与攻击者建立连接

Because they've been filled with attacker IPs, it will connect to the attacker.

节点表是如何工作的呢？

So, how do these peer tables work?

每个节点会从两个表中选择建立连接的节点IP

Well, each node picks its peers from two tables.

新表存储了此节点知晓、但是尚未和此节点建立连接的节点IP地址

The new table stores IPs that the peer is has heard about but not connected to you.

过表存储了曾经与此节点建立过连接、但现在可能没有建立连接的节点IP地址

And the tried table has IPs that the node has peer with at some point in the past, but may not be a current peer.

两个表都包含分桶 分桶中存储了具体的IP地址

These tables contain buckets. And inside the buckets, there are IP addresses.

我们用绿色圆圈表示诚实IP地址

We represent honest IPs by the green circle.

两个表会为每个IP地址存储一个对应的时间戳

The tables also store timestamp for each IP.

我这里不会为大家解释新表中时间戳的作用 因为这对发起攻击起不到关键作用

I won't go into how timestamps work for the new table, because it's not important to our attack.

过表中存储的时间戳 指的是节点与此节点最晚建立连接的时间

But in the tried table, the timestamp is the last time that node connected to our node.

也就是节点与此节点建立连接的最晚时间

So, the last time this IP address connected to the node.

如果一个节点想创建一个对外连接 它需要从这两个表中选择IP地址

So, if a node wants to make an outgoing connection, it's going to use these IPs.

节点首先会确定其从新表还是从过表中选择IP地址

The first thing it decides is whether it's going to choose from the new or tried table.

一旦它选择好了表 它就会从这张表中选择一个IP地址

Once it's selected a table, it selects an IP address from in the table.

节点会尽可能选择对应时间戳更接近当前时间的IP地址

This selection is biased towards fresher more recent timestamps.

随后 节点尝试与此IP地址建立对外连接

And then, it attempts an outgoing connection to that IP address.

攻击者要做的是用攻击者IP地址、即图中红色源泉的地址填满这两个表

So, what the attacker wants to do is just fill these tables up with attacker IP, it is represented by the red circles,

使得当节点选择IP地址时 其只能选到攻击者的IP地址

so that when it goes to choose an IP address, there is no choice but to choose attacker IPs.

进一步 攻击者需要保证攻击者IP地址的时间戳永远是最新的 从而持续发起攻击

Furthermore, the attacker can till the game in their favor by ensuring that their IPs are always the freshest IPs.

攻击者如何让自己的IP地址进入到这两个表中？

So, how does the attacker actually get IPs into these tables?

当节点与另一个节点建立连接时 此节点会从另一个节点的过表中得到IP地址

Well, whenever a node makes a connection to another node, it gets its IP in the tried table of that node.

进一步 一旦建立连接 节点可以宣布其它节点的IP 这些IP会进入到此节点的新表中

Furthermore, once a connection has been established, a node can announce additional IPs and those go in the new table.

我这里画了4个红色的点 大家可以想象一共有…

Now, I drew it with four dots. But you could imagine this with…

在实际中 通信的消息可能包含上千个IP地址 一个节点也可以发送多个消息

Well, in actual practice, these messages can contain up to a thousand IPs and you can send multiple messages.

很容易在新表中填满攻击者的IP地址 我就不为大家详细讲解方法了

So, filling the new tables is trivial. And we won’t talk about it in this attack.

在过表中填满攻击者的IP地址是很困难的

It's much harder to fill the tried table.

这也是我们在实施攻击时关注的核心步骤

And so that's what we're going to focus on.

一旦成功 攻击者就可以重复这一步骤

And then, the attacker can just repeat this.

对于在控制范围内的每一个节点 攻击者都可以把其控制的IP地址不断传入过表中

And for each node under its control, for each IP address that it controls, it gets an additional IP into the tried table.

随后 攻击者等待此节点重启

And then, the attacker waits for the node to restart.

一旦节点重启 当此节点选择并建立新的对外连接时

Once the node has restarted, when it goes to select new outgoing connections,

因为过表和新表中的IP地址都被攻击者所控制 此节点会与攻击者IP地址建立连接

because it's filled the tried and new table, these connections will be to attacker IPs.

这样一来 攻击者就日蚀了这个节点的所有对外连接

And so, the attacker has eclipsed all outgoing connections.

但请大家注意 这些节点还包含了对内连接

But remember that I said, these nodes also have incoming connections.

如何日蚀对内连接呢？

So, how do we do this?

非常简单 攻击者可以对相同的IP地址建立117个对内连接

Well, it's actually really simple. You can make 117 connections from the same IP.

这样就可以日蚀此节点的全部对内连接了

So you can just fill up all the incoming connection slots on the node and eclipse it.

节点很容易重启吗？实际中可行吗？为何节点会重启？攻击过程需要节点重启

So, how easy are these restarts? Is it practical? How can this happen? The attacker needs these restarts.

我们知道 用户每周基本会有一天要为计算机安装补丁

Well, we all know there's a day of the week in which you patch your computer.

因此 攻击者可以在补丁发布前一段时间实施攻击 或等待补丁发布

So, attackers could plan around that or could wait for that to happen.

每隔一段时间总会出现一些重要安全漏洞补丁的 节点或者打补丁 或者等着被攻击

Also, there's critical security vulnerabilities that are released that the victim either needs to patch or it's lose/lose for the victim.

在比特币中 每隔一段时间就会发布一些抵御拒绝服务攻击漏洞的补丁

There's been several denial of service and packets of death CVEs in Bitcoin,

操作系统发布补丁的频率会更高

and many more in the underlying OSes.

研究表明 每过10小时 一个比特币公开节点就会有25%的概率变为离线状态

And one study said that within about 10 hours, a public Bitcoin node has about 25% chance of going offline.

需要注意的是 比特币点对点网络的安全性并不要求所有节点时刻保持在线状态

And then, finally the security of Bitcoins peer-to-peer network should not rely on 100% node uptime.

当然了 填满过表并不像想象中的那样容易

So, the attack isn't as easy as just filling the tried table.

此过程包含一个复杂的细节步骤 这也是这些分桶的核心作用

There's one complicating detail and it's the reason for these buckets.

当一个IP地址被发送到过表中时 比特币系统会把此IP地址分成两个部分

So, an IP address is going to be sent to tried. And what Bitcoin does is it breaks the IP address into two pieces.

第一个部分称为组 第一个部分包含IP地址的16位前缀

The first part is called the group. It's the slash 16 prefix of the IP.

而16位后缀就是IP地址的另一部分了

And the second part is just the second part of the IP.

我们用哈希算法将组分成4个桶 因此每个组会对应4个桶

We hash the group to 4 bucket, so it chooses 4 buckets.

随后 我们会选择其中一个桶 并把IP地址的第二部分放入这个桶中

And then, the second part of the IP chooses one of those buckets for the address to be placed in.

这一存储过程会让攻击实施起来更加困难

Now, this actually makes the attack more difficult.

假设我们有一系列连续的IP地址

Imagine you have a whole bunch of contiguous IP addresses.

由于IP地址是连续的 因此这些IP地址会很容易被划分在相同的组中

If they're within the same group which they're likely to be unless, it's an enormous amount of continued IP addresses.

这些IP地址的16位前缀会完全一致

They'll be within the same slash 16.

因此 我们只能用攻击IP地址填满4个桶

You're only going to be able to fill those 4 buckets.

这意味着攻击节点包含大量的IP地址

So, this either requires massive numbers of IP addresses,

或者攻击节点需要一个僵尸网络 使IP地址足够多样化

or it requires that you have a botnet that has sufficient IP diversity.

在我们的论文中 我们同时考虑了这两种场景

And in our paper, we consider both scenarios.

在本次讲座中 我们只考虑僵尸网络这一场景

But in this talk, we're just gonna look at botnets.

你该怎么做呢？

So, what do you do?

你是一个攻击者 你只有有限个IP地址

You're an attacker. You have a limited number of IP addresses.

你拥有的IP地址越多 你就越容易发起攻击

The more IP addresses you have, the better off it is.

IP地址越被划分在不同的组中 过表中的诚实节点IP地址就会越少 情况越好

The more IP addresses in distinct groups, the fewer honest IP is in the tried table, the better as well,

因为与你竞争的IP地址就会越少

because you have less to compete with.

我们已经讲解过 IP地址的选择是有偏的

And we've already showed that you have this selection bias

你可以利用这一点 使节点尽可能与攻击节点建立连接

that you can exploit to get more bang for your buck for your IPs.

注意到 如果你能让诚实节点的时间戳越陈旧 攻击成功的概率也会越高

But also, notice that the selection bias if you make the honest IPs more stale. It helps you as well.

因此 你能做的是长时间地运行攻击过程

And so, what you can do is you can just run the attack for longer.

如果诚实节点的时间戳都是1小时之前 而你的攻击过程执行了5个小时

If the honest IPs all have timestamps from one hour ago and you run the attack for five hours,

则诚实节点时间戳就变成6小时前了

now there's six hours stale.

你基本上可以利用两种资源：你拥有的IP地址 以及发动攻击的时间长度

So, you basically have two resources, the number of IP addresses you have, and the time invested in the attack.

当然 情况比想象的糟糕 攻击者可以利用的点比想象的多

So, it actually gets worse. There's some other things the attacker can exploit.

每个分桶只能存储64个IP地址

So, these buckets only hold 64 IP addresses.

如果一个分桶已经满了 即这个分桶已经包含了64个IP地址

So, if we have a bucket that's full that is 64 IP addresses in it,

当新的IP地址被哈希到这个桶中时 节点会从桶中驱逐出一些IP地址

and the new IP is going to be hashed to that bucket, it needs to evict something.

比特币系统中的地址驱逐过程是

And so, the Bitcoin eviction routine is to

从旧IP地址中随机选择4个IP地址并驱逐 再将4个新的IP地址添加进桶中

randomly select 4 IPs to delete the oldest IP, and then to insert the new IP in its place.

这进一步为攻击者提供了2个漏洞

This gives the attacker two additional vulnerabilities.

因为被驱逐的IP地址是随机选择的 如果攻击者的一个IP地址被驱逐

Because this selection process is random, if the attacker evicts one of their own IP addresses,

攻击者可以对这个IP地址重新执行攻击过程 从而潜在驱逐一个诚实节点IP地址

they can just rerun that IP address and potentially evict an honest one.

因此 通过不断尝试 诚实节点IP地址被驱逐的概率就会越来越高

So, they can just improve their chances by trying over and over again.

我们之前也已经讨论过 攻击者可以保证其IP地址总是新鲜的

Additionally as we've already discussed, the attacker can ensure they're the most fresh.

因此 如果4个节点中有一个诚实IP地址 攻击者基本可以保证把这个IP地址驱逐

So, if one of the 4 is an honest IP, the attacker can basically ensure that the honest IP will be evicted.

我们已经解释了攻击方法 但谁可以实施攻击呢？是否容易实施攻击呢？

So, we've explained how to do this. But who can do this? How easy is this to do?

我们的验证方法是查看比特币系统源代码 利用概率分析方式建模

So, our approach was to look at the Bitcoin source code, and model it with probability analysis,

进一步应用蒙特卡洛仿真方法验证我们的概率分析结果

and then also use Monte Carlo simulations to validate our probability analysis.

我们应用此模型确定有效的攻击参数 攻击执行时间 所需的IP地址数量

We use these models to determine effective attack parameters, time invested, number of distinct IPs.

随后 我们在实际场景中对这些参数进行验证 攻击一个在线的比特币节点

And then we experimentally verified these parameters against live Bitcoin nodes.

我们对自己的一个比特币节点实施了攻击 但此节点确实属于比特币网络中的一部分

We did these to our own Bitcoin nodes, but they were part of the Bitcoin network.

我们执行了多种实验 但我这里只为大家讲解其中的两个实验

So, we ran several experiments, but I'm only going to tell you about two.

第一个实验是最糟糕情况下的实验

The first experiment was a worst-case experiment.

我们想知道攻击者在何种情况下一定可以成功实施攻击 无论过表中存了什么IP地址

So, we wanted to see how an attacker could win all the time, no matter what was in the tried table.

我们人工将过表中的IP地址用诚实IP地址填充 并在攻击前将时间戳设置为最新

So, we artificially filled the nodes tried table with honest IPs and these had the freshest timestamps of just before the attack.

我们的模型预测攻击过程需要4600个IP地址、每组两个IP地址、攻击时间为5小时

Our model predicted we needed about 4600 IPs with two IPs per group, and about 5 hours invested in the attack.

攻击结束后 节点过表中的IP地址几乎都是攻击者的IP地址

After the attack, the nodes tried table was almost completely filled with attacker IPs.

我们执行了50次实验 在每次实验中 攻击者有100%的概率日蚀全部8个对外连接

And we performed this experiment 50 times, and in each time, the attacker was able to eclipse all 8 outgoing connections for a 100% success rate.

当查看已有僵尸网络的规模后 我们发现此攻击所需的僵尸网络规模很小

And when we look at botnet sizes, this is clearly a pretty small botnet.

实际僵尸网络的IP地址更加多样化 IP地址数量也更多

Botnets have both the diversity and numbers far in excess of this.

因此 我们想在第二个实验中知道 此攻击在实际非最糟糕情况下的节点攻击效果

So, the second experiment, what we ran was we wanted to see how would this work against a realistic non-worst-case node.

我们有一些比特币节点 我们让这些节点与比特币点对点网络连接了超过43天

So, we had some Bitcoin nodes. We'd connected them to the network for about 43+ days.

我们要攻击的节点 其过表中包含了大概300个诚实IP 显然过表并没有被填满

The node that we're gonna look at had a tried table of around 300 honest IP. So, it was not full at all.

基于此 我们认为使用一个包含400个IP的僵尸网络、执行1小时攻击就足够了

And so, based on this, we determined the botnet of around 400 IPs should be sufficient investing one hour.

攻击结束后 过表中的大部分存储内容还是空的 但大多数IP地址都是攻击者的IP地址

And while the tried table was still empty after the attack, a majority of the IPs were attacker IPs.

我们得到攻击成功的概率为84%

And we got an 84% percent success rate.

我们执行了50次实验 84%的实验结果中 节点的8个对外连接都被日蚀

So, ran the experiment 50 times, and in each time all 8 outgoing connections were eclipsed.

随后我们查看了Carna僵尸网络 因为此僵尸网络的IP地址是公开的

And we looked at the Carna botnet, because it's IPs have been published.

如果我们从Carna中随机采样1250个IP地址

And if you were to randomly sample 1250 IPs from the Carna,

则平均上看 这些地址的分组已经足够多样化

you would get the group diversity on average.

因此 只用Carna的子集 我们就已经有了所需的、充分多样化的IP地址了

So, just shrink a subset of Carna, and you would have enough groups to perform this attack.

我们给出了几个日蚀攻击的抵御方法

So, we developed some countermeasures.

在思考所有的漏洞后 我们提出的第一种方法针对的是IP地址新鲜时间戳选择过程

The first kind of measure we came up with thinks about this vulnerability, we have a vulnerability towards selecting fresher timestamps.

如果IP地址的选择完全是随机的 则这个漏洞就不存在了

So, if we just randomize the selection process, that vulnerability goes away.

另一个我们要解决的漏洞是有偏的IP地址驱逐过程 旧的IP地址会被驱逐

An additional vulnerability we face is eviction bias, where the oldest IP addresses are evicted.

我们还要解决频繁尝试漏洞 攻击者可以利用这个漏洞提高概率…

And then we also have this additional vulnerability of try-try again, in which you could improve your chances by…

如果攻击者的IP地址被驱逐 攻击者可以重新执行攻击 提高不被驱逐的概率

If you evict yourself, you can run it again and then hope not to evict yourself, and continually improve your chances.

频繁尝试效果怎么样？我们画了这样一张图

And what this looks like? We graph this.

y轴表示的是攻击者在过表中填充的IP地址数量 攻击者所能填充的IP地址数量

And so the y-axis is number of IPs that attacker gets from the tried table, how full the attacker gets the tried table,

横着的这条黑线表示过表中已经存满了IP地址

this line here is the tried table being completely full.

大家可以看到 曲线是线性的

And so, as you can see, it's actually fairly linear.

当插入4000个IP地址时 过表中大约也存储了4000个IP地址

For about 4000 IP addresses, you get nearly 4000 IP addresses in the tried table.

因此 我们提出了确定性随机驱逐过程 以抵御日蚀攻击

So, we propose a countermeasure of deterministic random eviction,

此过程不仅将IP地址映射到分桶中

in which not only do IP addresses map to buckets,

每个IP地址也会映射到分桶中确定的位置上

but they deterministically map to positions within those buckets,

驱逐过程也直接指向各个位置上的IP地址

evicting whatever happens to be in that position.

这样做就可以移除这两个漏洞了

This removes both of those vulnerabilities.

我们也能得到一个更好的曲线

And we get a much nicer line.

此时 为了让比特币系统驱逐全部4000个IP地址 你需要万级IP地址来实施攻击

So, to get close to what under Bitcoin eviction took about 4000 IP addresses, you'd need more than thousand IP addresses.

我们还需要解决的一个问题是 在实际场景中 过表的填充速度过慢

An additional problem we face is that in our realistic scenario, the tried table filled up very slowly.

我们提出了试探连接方法

So, we have feeler connections.

试探连接法会从已经被实际验证在线的新比特币节点中得到IP地址

A feeler connection takes IPs from new verifies that they are in fact online Bitcoin nodes.

随后 节点会将这些IP地址添加到过表中

And then, it adds those IPs to the tried table to fill up the tried table.

另一个问题是 过表会驱逐诚实节点

Another problem is that good honest IPs are evicted from the tried table.

因此在驱逐前我们增加了验证过程

So we have a test before evicted.

我们验证过表中的IP地址 如果此IP地址在线 则不进行驱逐

We test IPs in the tried table. If they're online, we don't evict them.

我们提出的这个防御方法借鉴了Storm僵尸网络中抗节点污染系统的思想

We actually borrowed this from Storm Botnet’s anti-peer-poisoning system.

我们将我们的攻击告诉了比特币开发者

So, we told the Bitcoin developers about our attack.

他们非常惊叹 并快速进行了相应

And they were awesome, and took a really quick action.

他们在比特币v0.10.1版本中实现了防御措施1、2、6

They implemented countermeasures 1,2 and 6 in Bitcoin v0.10.1,

上次检查时 我发现60%的公开节点已经应用了此版本的比特币系统

which has 60% public node deployment right now on last time I checked.

但我们感觉“试探连接”和“驱逐前验证”也是非常重要的防御方法

But we felt feeler and test-before-evict were also very impo**r**tant countermeasures.

我们在一个补丁中实现了这两种方法 当前此补丁正在等待被检验

So, we implemented these in a patch. And it's currently awaiting review.

这些防御方法的有效性怎么样？

How effective are these countermeasures?

我们在最糟糕场景 即过表中存储的均为诚实节点的场景下部署了这些防御方法

Well, the countermeasures that are currently deployed in the worst case scenario with a full tried table

实施攻击所需的IP地址数量从4600个提高至41000个 成功概率降低到50%

raise it from about 4600 IPs to 41,000 IPs with an attacker success rate of a coin flip.

在真实节点场景下 所需的IP地址数量大约翻了10倍 且攻击成功概率仍然变低

In the live node setting, they increase it by almost a factor of 10 with a lower attacker success rate.

在最糟糕情况下 “驱逐前验证”方法效果最好

And in worst case, with our patch test-before-evict is really helpful,

因为如果过表中存储的全都是诚实IP地址

because if your tried table is filled with honest IPs,

则攻击者无法将这些IP地址驱逐 因为诚实节点大多常处于在线状态

then the attacker can't push any of them out, because they'll all be online.

因此 如果你现在处于安全状态 则你基本会一直处于安全状态

So, if you're in a safe state, you tend to stay in the safe state.

在论文中 我们在假定诚实节点会更换的条件下进一步分析了此防御方法的效果

We analyze this in more detail assuming turning of honest IPs in our paper.

试探连接可以保护实际场景中的节点不遭受日蚀攻击

And feeler connections actually prevent us from talking about the live realistic node setting,

因为过表的填充速度会变得更快

because the tribe tables going to fill up much quicker.

这一防御方法效果更好 会使得实际攻击效果趋近于最糟糕情况下的结果

So, it's better than this. It probably is moving this more to a worse case.

但我们后续需要执行更多的实验 进一步讨论此防御方法的效果

But we have to perform more experiments to discuss it in detail.

事实证明 此攻击确实太糟糕了 点对点网络实际上是有漏洞的

So, one objection is that this is really bad. The peer-to-peer network is vulnerable.

旷工或许不应该连接点对点网络 他们应该只连接诚实参与方

Maybe miners shouldn't connect to the peer-to-peer network at all, and only connect to trusted parties.

但是谁来确定谁才是诚实参与方呢？

But how do you decide who to trust?

大矿池由于忙于挖矿 其行为反而比诚实节点不可信一些

Major mining pools have engaged in things which are less than honest behavior.

如果你在网络中创建了一个诚实节点集合

And if you did create some trusted club of networks,

你如何让其它节点加入到此集合中 你如何确定谁才是可信的

how do you let new people in, how do you determine who to trust,

你如何保证诚实节点集合不会成为一个中心化的节点集合呢？

and how do you prevent this from becoming a centralized club?

我们的目标是让比特币系统在点对点系统中足够健壮

So, our goal was to make Bitcoin robust to peer to peer attacks

同时使得比特币系统仍然是去中心化的 这是比特币系统的核心

while preserving the decentralization that's the heart of Bitcoin.

最后是总结部分 日蚀攻击破坏了比特币的核心安全性保证

So, in summary, eclipse attacks violate Bitcoins core security guarantees.

日蚀攻击是实际可行的

These attacks are practical.

即使利用一个非常小的僵尸网络也可以对真实比特币节点发起攻击

Even a very small botnet can pull them off against realistic Bitcoin nodes.

我们当前已经有了应对日蚀攻击的防御方法

And we currently have countermeasures to resist these.

而其中一部分防御方法已经在实际中部署 从而保护比特币系统

And some of these countermeasures are already deployed and protecting Bitcoin.

你好 这是个很棒的工作

Hey, really nice work.

谢谢

Thanks.

我很好奇 为何比特币开发者最初要设计一个复杂的分桶机制和随机节点选择机制？

I'm curious why Bitcoin originally decided to do this complicated buckets and tried a new versus a random selection that seems to be superior to that?

这么设计的思考逻辑是什么？

What was their thinking there?

我没法告诉你思考逻辑是什么 因为我也不知道他们的思考逻辑是什么

So, I can't speak to their thinking, because I don't know they're thinking.

通过阅读代码 我猜想他们的思考逻辑是这样的

Looking at the code, what I imagine their thinking was that

最新知道的节点处于在线状态的概率更高 因此更应该连接这些节点

they wanted to ensure that nodes that you've just recently heard about that were more likely to be online, would be more likely to be chosen.

不过比特币开发者很喜欢我们提出的防御方法 我们的方法可以简化代码

But they actually really liked this countermeasure. It simplified their code base.

因此 他们很快进行了测试并完成了部署 他们认为新的机制确实优于之前的机制

And so, they tested it and adopted it, and felt that it was superior to the previous behavior.

你好 我是来自Rutgers大学的Grady Clark

Hi, Grady Clark, Rutgers University.

不好意思 我想问两个问题

I'm wondering I have two questions.

为什么比特币开发者只决定实现对策1、2、6 但没实现“驱逐前尝试”机制呢？

So why did the Bitcoin developers only decide to do 1, 2 and 6, and not to try-before-evict.

此种攻击方法的主要目标是谁？

And who are the main targets of this type of attack?

据我所知 当然可能是错的 矿池一般只对外暴露一个节点 因此应该很难实施攻击

So, our mining pools open because from what I understand, this might be wrong. They represent a single node that tries many things.

这就是我想问的问题 谢谢

So, these are the questions, thank you.

好的 我先回答你的第一个问题

Alright, let me answer your first question.

很容易实施对策1、2、6 而且实施结果也非常有效

The countermeasures 1, 2 and 6 were just really easy wins.

这几个对策可以减少代码的长度 使系统更简洁 且攻击者更难实施攻击

They reduced the number of lines of code. They were simpler. There was less things that the attacker could use against the system.

因此 这几个对策的实施不会对代码造成太大的改变 简单修改代码即可实施完毕

So, and there was very little risk of code changes. They were just making code simpler.

“试探连接”机制和“驱逐前尝试”机制会对系统架构造成较大的影响

The feeler connections and test-before-evict involved larger architecture.

我们自己的补丁涉及的改动量比较大 并且这两个机制也需要更多的测试工作

The patch that we have is much bigger. And it requires much more testing.

我们已经完成了很多的测试 希望我们能够完成这两个机制的部署

We've performed a lot of tests on it. And hopefully we can get it deployed.

这就是为什么比特币开发者选择实施对策1、2、6的原因

But that's the reason that they chose 1, 2 and 6.

他们可以很快地完成这三个对策的部署

And they were able to bang it out amazingly quickly.

至于攻击目标是谁这个问题 的确如你所说 旷工一般会使用网关

The issue in terms of who would be targeted, yes miners use gateways.

如果旷工完全依赖于点对点网络的话 那么旷工易受到此类攻击

And if miners relied purely on the peer-to-peer network, they would be vulnerable to these attacks.

我们没有调研旷工的现状 因此不知道那些旷工易受到攻击 哪些旷工不易受到攻击

We didn't do a survey of miners to determine who was vulnerable and who is not.

商人也可以成为攻击目标 如果有人运行本地比特币系统 也可以用来攻击比特币钱包

But you could also attack merchants with this. You could attack wallets if you were doing some like local Bitcoin.

如果你与另一个人在进行比特币交易 你自己也可能成为攻击目标

Well, if you were doing some sort of Bitcoin transaction with a person, you could do this.

我们知道 某个人在一次狂欢节上实施过日蚀攻击

And we know in fact that someone did an on-path eclipse attack at a carnival.

人们在狂欢节上用比特币购买物品 而狂欢节网络上传速度较慢 攻击实施起来更快

And they did this to make things run faster. They had a very slow uplink. And people were buying things at the carnival with Bitcoin.

攻击者入侵了每个人的网络连接 把连接都指向了一个节点

So they just like hijacked everyone's connection, sent it through one node,

这使得所有节点依赖于单一节点的缓慢上传速度与比特币网络建立连接

and then had that node use the slow uplink to the rest of the Bitcoin network.

你好 我是来自Rice大学的Matthew

Hi, I'm Matthew from Rice University.

是否可以考虑节点年龄 如果节点没有为区块链做出过贡献 就不会被添加进IP库中

Was there any consideration of using node age in the sense of no contribution to the blockchain as a consideration for including IP inside,

相对来说 新用户更可能是攻击者 因此我们是否可以通过这种方式保护过表？

so that relatively new users that were more likely to be attackers could be prevented from being in the tried tables?

我理解一下 你的问题是：能否判断节点在比特币中存在的时间或对比特币做出的贡献

So, if I understand your question, you're asking we could determine nodes which have been around a long time and have contributed Bitcoin.

正确

Correct.

并以此作为节点可信度的度量方式

And use that as a trustworthy.

是的

Yes.

我认为这是一个很不错的想法

So, I think that's a great idea.

目前在比特币系统中 生成的区块没有与节点的IP地址关联到一起

Currently in Bitcoin, nodes are not associated with blocks being produced.

是的

Right.

如果真能实现这样一个机制 确实可以做到查看某个节点在比特币系统中的存在时间

But if you did have a mechanism for this, it would be very good to look at the age of someone,

并认为出现时间更长的节点可信度更高 更可能是系统中的可信任节点

and say, well this person has long term been known as honest, a reputation system.

但是当前比特币节点不包含任何密码学身份证明机制

But Bitcoin nodes don't actually have any cryptographic proof of identity.

同时 部分节点的IP地址变化频率会比较高

And some of them switch IP addresses quite frequently.

是的

Right.

因此 比特币本身很难实现这样一个机制 但有可能在比特币之上建立这样的机制

So, it would be a little bit difficult to do with Bitcoin, but you could layer something on top.

谢谢

Thank you.