Security Now! #781 - 08-25-20

SpiKey

This week on Security Now!

This week we look at a new Chrome remote code execution flaw, some interesting news of three new Ransomware victims, an emergency patch from Microsoft, the emergence of amateur RDP exploiters, the 15th birthday of the Zero Day Initiative, finally a good Windows 10 garbageware remover, recommendations of several of my most recommended remote networking utilities, then a bit of miscellany and SpinRite news. Then, finally, we examine a really terrific new high-tech hack against low-tech locks and their keys.

In just under a year, on August 17th, 2021, the use of IE11 will no longer be supported for Microsoft's online services like Office 365, OneDrive, Outlook, and more. Microsoft is also ending support for IE11 with the Microsoft Teams web app later this year, with support ending on November 30th.

Browser News

An RCE in Chrome's WebGL

Google Chrome users should, today, be moving to Chrome 85. And that's good, since it will fix a potentially serious remote code execution vulnerability in Chrome's WebGL rendering engine. It's a "use after free" read flaw discovered by Cisco's Talos unit and responsibly reported to Google more than 3 months ago, back on May 19th. Google quickly put the fix into their early release Dev and Beta channels in early June, and the stable channel is due to receive the fix today with Chrome 85. There's no indication that it was ever publicly known or exploited, so no emergency on this one.

Ransomware

What do The University of Utah, Jack Daniel's Whiskey, and Carnival Cruise Lines all have in common?

Friday, the University of Utah revealed that it had paid a ransomware gang $457,059—not to obtain the decryption key for their files (very few were encrypted) but rather to purchase the promise from the extortionists that the student information that had been obtained would not be publicly released.

The University explained that it had dodged a major ransomware incident and that the attackers managed to encrypt only 0.02% of the data stored on their servers. And the University staff was easily able to restore that from backups. However, the ransomware group then threatened to release student-related data they had obtained and exfiltrated online.

So the University said: *"After careful consideration, the university decided to work with its cyber insurance provider to pay a fee to the ransomware attacker. This was done as a proactive and preventive step to ensure information was not released on the Internet. The university's cyber insurance policy paid part of the ransom, and the university covered the remainder. No tuition, grant, donation, state or taxpayer funds were used to pay the ransom."*

The University disclosed that the attack took place a little over a month ago on July 19, 2020 and that the network belonging to the College of Social and Behavioral Science was the victim.

And...

Two other large and notable recent ransomware victims were Brown-Forman, famous for their distillation of **Jack Daniel's Tennessee Whiskey**, and **Carnival cruises**.

The Jack Daniel's folks said: "Our quick actions upon discovering the attack prevented our systems from being encrypted. Unfortunately, we believe some information, including employee data, was impacted. We are working closely with law enforcement, as well as world class third-party data security experts, to mitigate and resolve this situation as soon as possible. There are no active negotiations."

That statement from Brown-Forman came after Bloomberg News reported that it had received

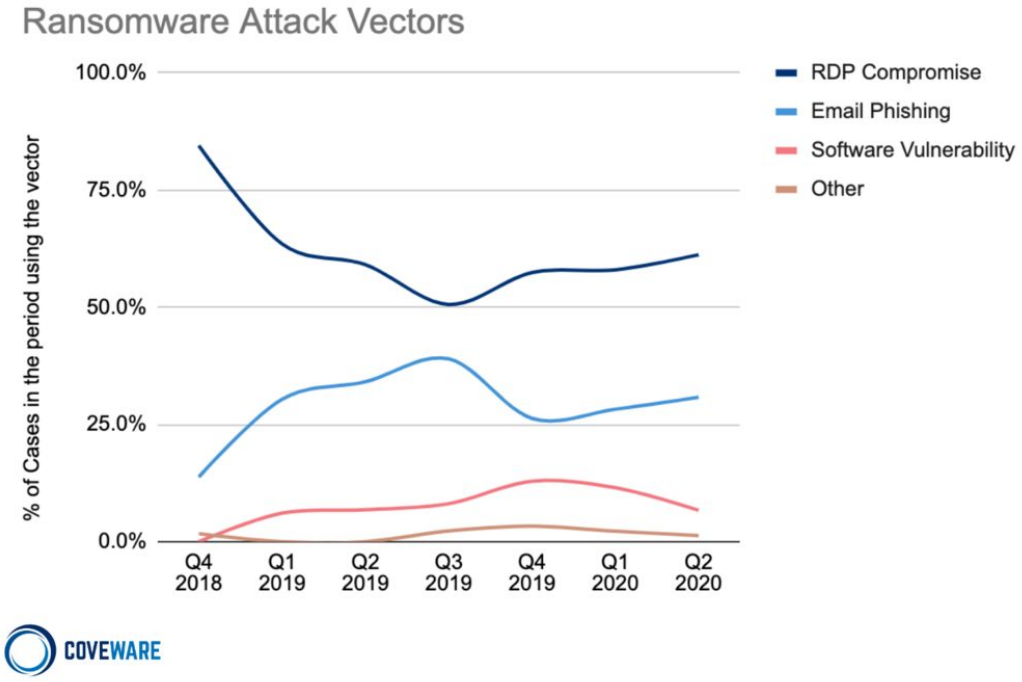
an anonymous tip of the ransomware attack. A site on the Dark Web claiming to be run by members of the REvil strain of ransomware, says it obtained 1 terabyte of data from Louisville, Kentucky-based Brown-Forman. The site said that stolen data included contracts, financial statements, credit histories, and internal correspondence of employees. Also included were screenshots of file structures and documents purportedly taken during the heist.

And as for Carnival Corporation, the operator of the world's biggest cruise lines, they disclosed that they were hit by a ransomware attack that provided unauthorized access to personal data of passengers and employees. And the company has not yet identified which of its many subsidiary cruise lines was breached. But they did need to disclose the attack in a regulatory filing with the SEC, part of which read:

Based on its preliminary assessment and on the information currently known (in particular, that the incident occurred in a portion of a brand's information technology systems), the Company does not believe the incident will have a material impact on its business, operations or financial results. Nonetheless, we expect that the security event included unauthorized access to personal data of guests and employees, which may result in potential claims from guests, employees, shareholders, or regulatory agencies. Although we believe that no other information technology systems of the other Company's brands have been impacted by this incident based upon our investigation to date, there can be no assurance that other information technology systems of the other Company's brands will not be adversely affected.

Security News

We all likely agree that a ransomware attack is the last thing any company wants. So the question is, how are these occurring. The traditional answer has been a phishing eMail which hooks some well meaning but unsuspecting insider.



While eMail phishing is, indeed, a popular entrypoint vector, three recent reports from Coveware, Emsisoft, and Recorded Future clearly show that phishing takes a back seat to our old friend RDP -- Microsoft's Remote Desktop Protocol.

In their report, Emsisoft explained what's happened this year:

> In recent months, organizations across every sector have come to rely heavily on Remote Desktop Protocol (RDP) to maintain business continuity while respecting social distancing.
>
> However, the rapid shift to remote working has also provided a unique opportunity for ransomware groups. Threat actors predicted that many organizations would not have the time or resources to securely implement RDP during the mass transition to working from home and, as a result, may be vulnerable to compromise.
>
> They were right. According to a McAfee report, the number of Internet-exposed RDP ports grew from approximately 3 million in January 2020 to more than 4.5 million in March.

Later in their report they note that while the threat is not new -- as we know all too well on this podcast -- the global shift to remote working has revealed that many organizations do not adequately secure RDP -- and that the bad guys are taking advantage. According to a report by Kaspersky, at the start of March 2020, there were about 200,000 daily brute-force RDP attacks in the U.S,. But, by mid-April, just six weeks later, that number had grown to nearly 1.3 million brute-force attacks per day. Now, today, RDP is regarded as the single biggest attack vector for ransomware.

This does sort of underscore the point I've been making, which is that RDP simply cannot be safely exposed to the public Internet. There are two reasons for this: it is no longer sane to trust that Microsoft hasn't, or won't, make a mistake in their provision of RDP. They have done so over and over in the past. And as we'll be learning in a few minutes, they just released an emergency patch for two more privilege elevation flaws in Windows' remote access service. The second reason we cannot trust RDP is that its native authentication mechanism are pathetic. I went looking for something that I might not know about RDP authentication and found a few months old "best practices" advisory from Microsoft. It amounted to "be sure to use a strong password." There is zero multi-factor support for RDP, which is unconscionable. There are multiple 3rd-parties who have responded to this need by creating much more secure RDP gateways laden with authentication features. So if you don't feel like rolling your own, and you have money to burn, you could simply throw some money at the problem and buy yourself this much-needed security. Or, you could get a bit clever and roll your own:

If the clients you have connecting, have fixed IPs, restricting access to the RDP port from only those IPs is an immediate, proven, fast and secure solution. If the IPs are largely fixed, as with the typical residential broadband service, the use of a DynDNS solution can allow for tracking their infrequent, but possible, changes. And typical changes may be so infrequent that updating the access firewall from DynDNS needn't even be automated.

But, if highly dynamic roaming access is needed, then no form of IP-based access restriction will suffice. This lifts the requirement for authentication from the network layer to the application layer. Again, history teaches that what we must avoid is public access to an RDP endpoint. We can't trust Microsoft and we can't allow for brute-forcing authentication. That requires the use of

something in front of the RDP endpoint. I've talked about using a VPN offering some form of strong multi-factor authentication -- either TOTP or certificate-based. But I wanted to add another option to the pot by noting that SSH is widely available, almost always offers the very strong authentication options we're looking for, and can be used to tunnel RDP. If you Google "tunnel rdp over ssh" you'll be rewarded with all the suggestions you might need, and for all OS platforms. The only theoretical down-side is that as a purist, both RDP and SSH use TCP. Long ago, when we were first covering the operation of VPNs, I noted that there can be some tunnel confusion when TCP is tunnelled inside TCP, since you have two sets of TCP's error recovery and packet retransmission. The theoretical optimal solution is for the VPN tunnel to use dump old UDP packets and for the tunnelled protocol, RDP over TCP in this case, to handle any packet losses. I did find some SSH UDP tunneling systems, but there is so much apparent success with simply tunneling RDP over standard SSH TCP tunnels, that it appears my theoretical concerns might be just that: theoretical.

So, anyway, I wanted to propose another option to the need for somehow hiding RDP from the outside world. It's clearly necessary to add another layer of security in front of RDP.

Speaking of which...
Last Thursday, Microsoft issued an emergency out-of-cycle update for Windows 8.1, Win 8.1 RT, and Windows Server 2012 R2. The emergency update patches a pair of recently disclosed security vulnerabilities.

https://support.microsoft.com/en-us/help/4578013/security-update-for-windows-8-1-rt-8-1-and-server-2012-r2

They're both high-severity privilege escalation vulnerabilities residing in the Remote Access Service (RAS)... which is obviously a particularly vulnerable area of a server. Interestingly, both of these vulnerabilities were patched as part of the previous week's August Patch Tuesday—but that was for Windows 10, Windows 7 (for those receiving extended support), and Windows Server 2008, 2012, 2016, 2019, and Windows Server versions 1903, 1909, and 2004 systems. In other words, everything other than windows 8.1 and its corresponding server 2012 R2.

As near as I can determine, the patches for these two OSes simply weren't ready in time for the regular monthly patch cycle. But they were too critical for Microsoft to leave them hanging until September's Patch Tuesday. So Microsoft decided to make them available now. It may be that they need to be manually installed, that wasn't clear. Assuming that they'll be part of next month's patch batch, end-users probably need not worry. But if you're running a Windows Server 2012 R2 it might be worth doing. Microsoft says that no reboot will be needed. So the patch for both can be applied without needing to down the server.

Iranian script-kiddies using RDP to deploy Dharma ransomware
While we're on the subject of RDP-based attacks, some interesting and disturbing research by a group known as Group-IB detailed the collision of RDP and ransomware. They explain that apparently low-skilled hackers, likely from Iran, have joined the ransomware business targeting companies in Russia, India, China, and Japan. They're going after the new low-hanging fruit represented by casually or hastily-deployed RDP servers using publicly available tools.

The group is deploying the Dharma ransomware and based on forensic artifacts it appears to be a non-sophisticated, financially-motivated group new to cybercrime. Their extortion demand ranges from 1-5 Bitcoin (currently $11,700 - $59,000) and they locate targets by scanning IP address ranges for exposed RDP endpoints. Their tool of choice for this stage is Masscan, an open-source port scanner we've talked about before.

Once they have located a potential target they launch a brute-force authentication attack using NLBrute, which is a utility that repeatedly attempts to authenticate using a list of username and passwords, attempting to find a combination that works. If they get in, they sometimes attempt to elevate their privilege by exploiting an old vulnerability (CVE-2017-0213) in Windows 7 through 10.

Researchers at cybersecurity company Group-IB learned about this new group in June during an incident response engagement at a company in Russia. Based on forensic artifacts, they determined the attacker to be “Persian-speaking newbie hackers.”

This conclusion is supported by clues from the next stages of the attack, which appear to lack the confidence of an actor who knows what to do once they've gotten in. Group-IB write: “Interestingly, the threat actors likely didn’t have a clear plan for what to do with the compromised networks. Once they establish the RDP connection, they decide which tools to deploy to move laterally. For instance, to disable built-in antivirus software, the attackers used Defender Control and Your Uninstaller.”

Further evidence that the operation is the work of a script kiddie from Iran comes from search queries in Persian to find other tools necessary for the attack and from the Persian-language Telegram channels providing them.

The number of victims compromised so far by this attacker is unknown, as is the path that led the threat actor to the Dharma ransomware-as-a-service (RaaS) operation. However, given that the Dharma operators provide a toolkit that makes it easy for anyone to become a cybercriminal, it should not come as a surprise that inexperienced individuals are deploying this file-encrypting malware. The senior analyst at Group-IB, Oleg Skulkin, said that the Dharma ransomware source code, which was leaked in March, likely explains the increasing use of this malware strain. Oleg indicated that “It’s surprising that Dharma landed in the hands of Iranian script kiddies who are using it for financial gain, as Iran has traditionally been a land of state-sponsored attackers engaged in espionage and sabotage”

We've talked about how this new “Ransomware as a service” model is allowing many hackers who would never be in the Ransomware game to become players. I don't see how this problem is going to go away.

So, again... no exposed open RDP ports, okay? Arrange to put something, anything, in front of RDP so that you or your company are not open to exploitation. This is not as bad as Microsoft’s original wide open Windows file and print sharing, which is what drove me to create GRC’s ShieldsUP! service so many years ago... but it definitely needs attention.

The Zero-Day Initiative turns 15

It turns out that there's a bit of a synchronized 15th birthday, since last Thursday, Pwn2Own's founding parent, the Zero Day Initiative (ZDI) also turned 15, just as this podcast did. **Our** first podcast was August 19th, 2005, one day before the founding of the ZDI program.

On last week's occasion of their 15th birthday, ZDI announced that more than $25 million dollars in bounties had been paid to security researchers over the past decade and a half. Those monies went to more then 10,000 security researchers across more than 7,500 successful bug submissions. In explaining the genesis of ZDI, they said:

> Starting in 2005, 3Com announced a new program called the Zero Day Initiative. The plan was to financially reward researchers who discover previously unknown software vulnerabilities and disclose them responsibly. The information about the vulnerability would be used to provide early protection to customers through TippingPoint IPS (Intrusion Prevention System) filters while the ZDI worked with the affected product's maker to fix the vulnerability.
>
> [So that's in interesting angle here. The commercial "TippingPoint IPS" would benefit from providing immediate awareness of a vulnerability, and could offer their customers unique early protection thanks to the IPS being immediately updated before any fix was made available from the vendor. That, would then, of course, fix the problem downstream of the IPS. And even after the vendor's problem is fixed, there's certainly some value in knowing when attacks are occuring — even if there's no longer any backend vulnerability — and also from where those attacks are originating.]
>
> That [first] year, the ZDI published a total of one advisory, pertaining to Symantec VERITAS NetBackup. Fifteen years later, we've published more than 7,500 advisories as we evolved into the world's largest vendor-agnostic bug bounty program. To say it's been a journey is an understatement. It's certainly had some ups and downs, but the program is stronger than ever and on track for our largest year ever. As we begin our 16th year, let's take a look at some of the more notable happenings in the life of the ZDI program.

I read through the entire posting and it provided such a useful — and synchronized — perspective and walk through the past fifteen years of **this** podcast that I wanted to share their observations:

2005 – 2010

Looking back at our activities through these years induces nostalgia as it reminds us of the bugs we bought in products (and companies) that are no longer with us. We can also see the rise of research into different products and technologies. For example, we bought only two Apple bugs in 2006. That number rose to 52 by 2010. Java bugs, particularly sandbox escapes, were also popular during this time. It's a bit odd to look back at the progression from buying bugs in what was simply known as "Java", to buying bugs in "Sun Microsystems Java", to buying bugs in "Oracle Java".

This time period also saw the first Pwn2Own contest, which was in 2007. The contest launched at a time when "I'm a Mac. And I'm a PC" commercials dominated the airwaves and Apple

devices had an aura of invincibility around them. Astute security researchers knew better, and Dino Dai Zovi proved it, winning himself a MacBook and $10,000. The contest has grown exponentially since that time. There are now three different competitions: Pwn2Own Vancouver, which focuses on enterprise software; Pwn2Own Tokyo, which focuses on consumer devices; and Pwn2Own Miami, introduced this year with a focus on ICS-SCADA products. Pwn2Own also served as a "coming out" for many high-profile researchers who, after winning the contest, went on to work on various prestigious teams and projects.

2010 – 2015

This was a transitional period for the program as 3Com, together with ZDI, was purchased by Hewlett-Packard, then later split off as part of Hewlett Packard Enterprise. However, the core principles upon which the program was founded on remain the core principles we operate by today:

- Encourage the responsible disclosure of zero-day vulnerabilities to the affected vendors.
- Fairly credit and compensate the participating researchers, including yearly bonuses for researchers who are especially productive within the program.
- Hold product vendors accountable by setting a reasonable deadline for remediating reported vulnerabilities.
- Protect our customers and the larger ecosystem.

By this time, the ZDI was large enough to have an impact on the overall ecosystem. It was during this period that we grew to become the world's largest vendor-agnostic bug bounty program, a title we still hold. In 2011, we had our first public zero-day disclosure when a vendor failed to meet the patch deadline. Over the years, holding vendors accountable has helped lower their response time from more than 180 days to less than 120. Even though we reduced our disclosure window, the rate of 0-day disclosure stayed relatively consistent.

Another big change during this period was the increase in research work done by the vulnerability researchers employed by the ZDI program. There have always been great people working on the program doing root cause analysis on submissions, but an increase in the size of the team allowed for members of ZDI to begin reporting their own bugs as well. ZDI researchers increasingly published their findings and expanded their speaking at high-profile conferences including Black Hat and DEFCON.

The increased size also helped spot some trends in exploitation. It was also during this time that we saw a surge in submissions of Java bugs. However, once browsers implemented "Click-to-Play," practical exploitation became more difficult. Bugs exploiting Use-After-Free (UAF) conditions in Internet Explorer were also quite common until the Isolated Heap and MemGC mitigation were silently introduced by Microsoft. ZDI researchers found a way to exploit the mitigations and were awarded $125,000 from Microsoft for the submission. Interestingly, Microsoft chose not to fix all the submitted bugs, so a portion of the report ended up as a publicly-released 0-day. In case you're wondering, all of the money was donated to various STEM charities.

During this timeframe, the bug bounty landscape became normalized and broadened. Vendors such as Microsoft and Google started their own bounty programs. Bug bounty platforms were

created that allowed companies like Starbucks and Uber to offer bounties.

[And by "Bug bounty were created" what they mean, without naming them, of course, is "HackerOne", which we've spoken of often.]

The idea of crowdsourcing research entered the mainstream. Not every program was successful, as some vendors suddenly realized that if you offer money for bug reports, you get bug reports. This left some companies scrambling to react after starting their program with mixed results. It was definitely a time of growth and learning throughout the industry.

Pwn2Own continued to grow as well. 2010 saw Pwn2Own's first successful mobile device exploit, demonstrated by Ralf-Philipp Weinmann and Vincenzo Iozzo against the Apple iPhone 3GS. We also started seeing vendors release large patches just before the contest. Since the rules require the "latest version" for all exploits, contestants often found themselves "patched out" just before the contest. It also meant the ZDI had to scramble to get the targets up to date with all of the latest patches – often staying up all night installing updates. In 2012, a second contest – Mobile Pwn2Own – was added to focus on phones and tablets.

2015 – Present

In 2015, Trend Micro acquired the HP TippingPoint IPS and the ZDI program along with it. This opened a new world of opportunity for ZDI, as the vulnerability intelligence produced by the ZDI program could now be used to improve not only the TippingPoint IPS but other products within Trend Micro's line of security solutions as well. ZDI's association with Trend Micro also resulted in a massive increase in interest in vulnerabilities in Trend Micro products themselves. To their credit, Trend Micro product teams have not shied away from the work of fixing the bugs submitted by independent ZDI researchers, and we have established a Targeted Initiative Program just for select Trend products.

The threat landscape shifted as well. Before 2015, we rarely saw an Adobe Reader submission outside of Pwn2Own. Once we reached 2015, there were more than 100 submissions. Many of those reports were submitted by ZDI researchers. Overall, internal finds represent ~20% of all of the cases we process every year. Bugs affecting Acrobat, Foxit, and other PDF readers continue to be prevalent. We've also seen the rise of deserialization bugs and a sharp increase in ICS/SCADA vulnerabilities. Home routers have also become a popular target since they can be compromised en masse to be used in botnets and DDoS attacks. As a result, the ZDI adapted and began accepting hardware-related submissions, especially those related to IoT devices.

The introduction of the Wassenaar Arrangement posed some challenges – especially when purchasing bug reports from member countries. However, we were able to navigate the paperwork needed to transfer "cyber arms" and stay on the right side of the law.

The Virtualization category was introduced to Pwn2Own in 2016, and since that time, we've had several guest-to-host escapes demonstrated. The contest celebrated its 10th anniversary in 2017 by acquiring 51 0-day vulnerabilities over the three-day contest. In 2019, we partnered with Tesla to award a Model 3 to a pair of researchers who exploited the car's infotainment system. ZDI researchers also demonstrated their own exploit of the infotainment system. The contestants have changed over the years, as well. In the beginning, individual researchers made

up the majority of entries with only a few teams participating. At one point, this shifted to most participants being teams sponsored by their employers. There have even been instances of teams filing bug reports with vendors before the contest in the hopes of killing their competitors' exploits. In the past couple of years, that has shifted back towards individuals and small, independent teams.

And we've never stopped growing. We hit our peak of 1,450 published advisories in 2018, and we're set to eclipse that this year. In fact, we've been recognized as the world's leading vulnerability research organization for the past 13 years. According to Omdia, the ZDI was responsible for over half of all measured vulnerability disclosures in 2019, more than any other vendor.

Moving Forward

Over the past 15 years, we've seen trends in the exploit economy and vulnerability marketplace come and go, but through it all, we've been laser-focused on one thing: making the digital world more secure, one CVE at a time. Through the tireless work of ZDI researchers and the wider community, we're determined to continue disrupting the vast cybercrime economy and raising the bar for enterprise software security for the next 15 years and beyond.

And, of course, through our own 15 years we have covered all of this, week by week, watching the terrain change as the industry as a whole has moved forward.

Miscellany

O&O AppBuster
https://www.oo-software.com/en/ooappbuster

Remote Utilities
https://www.remoteutilities.com/

It's a paid commercial app but NOT a subscription, which being an old fart I would never consider, just on principal. But they have an entirely useful limited Free License:

"Our free license allows you to add up to 10 remote computers in your Viewer address book. You can use the free license in a business and personal setting. Only one free license key is allowed per individual, company or organization. For more information, please see our EULA."

The system consists of a Host (server), a Viewer (used by the tech) and an optional Self-Hosted Server which can be used to fully privatize the entire system.

There's also an "Agent" which can be used for spontaneous access to a remote system without installation. You'd send the Agent to someone, have them run it, then you could connect and do everything you want. And there's, similarly, a zero-installation "Viewer" that can be launched from a USB thumb drive, so you could access your set of remote systems from any other system.

PCWorld, in their review of Remote Utilities, wrote: "For power users, there's plenty to like about Remote Utilities. Several connection modes are offered beyond the full remote desktop experience. There's also a file transfer mode, remote device manager, registry viewer, remote webcam access, and a terminal mode — which is an excellent way to perform simple command line tasks from a distance.

There's an "MSI Configurator" to create a custom Host installer for unattended access or to customize the remote Agent module with your own logo and welcome text for attended support.

There's "Power control mode" which allows you to remotely restart a PC in normal or safe mode, shut it down, lock it or put it to sleep.

There's Active Directory support. Fetch Active Directory tree, add new domain controllers and access Active Directory workstations and servers with one click using Windows Security authorization.

And, of course, you can optionally enable standard TOTP 2-factor authentication for access to specific or every remote Hosts, generating the session access code using any standard TOTP authenticator app.

All traffic that Remote Utilities sends over the network is encrypted using TLS 1.2 for secure communication between the endpoints. Encryption is always on and cannot be disabled by the user. Transferred data is encrypted regardless of connection type and user license.

The address book can be encrypted in case the Viewer workstation is compromised.

Host identity is automatically certificate-based to ensure that you're connecting to the same Host you intended to connect to. If the Host certificate is ever changed the Viewer immediately raises a red flag and shows a warning for you to decide on further action.

Remote Utilities can be deploy in a totally isolated environment — no communication with their servers is necessary. There are 3 ways to do that:

1. Establish direct connection with remote computers on the same LAN or VPN.
2. Enable connecting through Host to use one of your Hosts to broker connection with other Hosts on the same LAN.
3. Run a self-hosted server to route your Internet-ID connections, synchronize your address book between multiple techs and authorize on remote Hosts in one click.

Regardless of the authorization method chosen, blank passwords are not allowed. There are no default or technical passwords either. Only you and the people you authorize will know how to access the Host that you have installed.

The system is protected from brute-force password cracking. When an excessive number of incorrect password attempts is seen, the system automatically begins increasing the amount of time required before each new attempt can be made. And if the system determines that the Host is under brute-force attack the originating IP address will be temporarily banned. And note that this only applies to exposed Hosts on an internal LAN. In the normal case, Hosting machines are

behind their local NAT router and are not presenting any open ports to the public.

And the features go on and on.

SyncThing

https://syncthing.net/ -- QNap (or Drobo) or macOS, Windows, Linux, FreeBSD, Solaris and OpenBSD. Run it on desktop computers and synchronize them with a server for backup. It uses rendezvous servers so it doesn't need IP addresses or advanced configuration: it just works, locally over LAN and globally over the Internet. Every machine is identified by a cryptographically strong ID. Give your ID to your friends, share a folder and watch it go. It can use external rendezvous relay nodes if it's unable to punch through the NATs, or can use UPnP or allow for manual port forwarding. And Leo will like that it's open protocol, open source (on GitHub), open development and open discourse.

I'm still in love with Sync.com. But Sync.com is limited to synchronizing a single folder tree among two or more Windows devices. It's perfect for what it does. And, somehow is also manages to retain every version of everything it has seen.

By comparison, SyncThing allows amazingly diverse and complex synchronizing networks to be set up. SyncThing is a bit funky and it has some learning curve. But if your need for creating a complex synchronization network might be long lived, the investment will likely pay you back richly.

All New Certs - Thanks DigiCert!

Over the weekend, using DigiCert's entirely automated self-service system, I rekeyed GRC's certificates ahead of the next-Tuesday, September 1st deadline, after which certs can only have a 397 day life. And among those that I rekeyed was GRC's "revoked.grc.com" cert, which I also used DigiCert's automated system to immediately revoke. So "revoked.grc.com" is back on the air and functioning properly. So I bought myself an extra year before I need to do that again, and I also realized a benefit of having the expiration date of all the certs synchronized: If we're going to be needing to do this annually, doing that certificate renewal work in a single batch will be much more convenient than being interrupted multiple times per year.

And speaking of certs, not only did the state of California recently allow a cert needed for uploading COVID labs test results to expire, but last Wednesday the 19th Spotify also suffered an outage when they allowed one of their critical TLS certs to expire. A Cloudflare network engineer indicated that a wildcard certificate for the Spotify hostname *.wg.spotify.com had not been renewed and expired.

As I indicated, it's unclear what the upshot of the new annual certificate renewal will be. Compared with the previous three year certs, this will triple the opportunities for this sort of mischief. And we still have the broken certificate revocation problem. This reduces the time that a stolen certificate can be used before it self-expires, but that's not much consolation since all certs will still be valid for 13 months. If we ever did get a revocation system working, these new shorter cert lifetimes would reduce the length of time that revoked certificates would need to remain in a revocation list.

SpinRite

One thing I meant to add last week was that as a result of the benchmarking we've done, we have learned a great deal about SSDs operating in the real world. Samsung, Kingston, OCZ-Vertex, Crucial and others. Their non-uniformity of their operation was not what anyone would expect from something with the seeming purity of solid state memory. Their operation was kinda all over the map and varied widely at the five points the benchmark tests. The thing I meant to note was that one brand stood out from all of the others. Samsung was, by far, the most rock-solid and every one of those that we saw followed the governing specifications to the letter. Across the industry, not all SSDs are created equal.

SpiKey

So, we already know that smartphone cameras now have sufficient resolution, and our software has become clever enough, that a photo of a traditional house key at a distance can be used to reconstruct a working physical key. And we also know that the vibrations of objects in a distant room -- a balloon, a bag chips, a lightbulb, or the leaves of a plant -- can be observed optically by laser or similar technology at a distance to reconstruct the acoustic waves those objects are being subjected to, to eavesdrop on conversations occurring in that room.

And now, with the publication of some intriguing new research, another piece of our traditional perception and assumption of security has just fallen to the wayside. The research paper, which documents the detailed and painstaking work by three quite enterprising students in the Department of Computer Science at the National University of Singapore bears the title:

Listen to Your Key: Towards Acoustics-based Physical Key Inference
https://www.comp.nus.edu.sg/~junhan/papers/SpiKey_HotMobile20_CamReady.pdf

ABSTRACT
Physical locks are one of the most prevalent mechanisms for securing objects such as doors. While many of these locks are vulnerable to lock-picking, they are still widely used as lock-picking requires specific training with tailored instruments, and easily raises suspicion. In this paper, we propose SpiKey, a novel attack that significantly lowers the bar for an attacker as opposed to the lock-picking attack, by requiring only the use of a smartphone microphone to infer the shape of victim's key, namely bittings (or cut depths) which form the secret of a key. When a victim inserts his/her key into the lock, the emitted sound is captured by the attacker's microphone. SpiKey leverages the time difference between audible clicks to ultimately infer the bitting information, i.e., the shape of the physical key. As a proof-of-concept, we provide a simulation, based on real-world recordings, and demonstrate a significant reduction in search space from a pool of more than 330 thousand keys to three candidate keys for the most frequent case.

In other words, these researchers have shown that just capturing the sound of a traditional physical key being slid into its lock is all that's needed to recreate that key with a high level of confidence. A nearby smartphone -- or even the house's nearby smart doorbell -- provides audio which is sufficiently accurate to provide the clues.

We all know how a traditional physical lock and key work. Inside the lock are a series of six spring-loaded pins which are each split at a different location along their length. When the proper key is inserted into the lock, the ridges on the key, pushing against those internal springs, positions each of the pins such that the splits in the pins line up with the edge of the lock's cylinder, thus no pin prevents the cylinder from freely rotating in the lock.

Because I'm a bit odd, throughout my lifetime I have often stopped to appreciate the sheer beauty of that simple invention. It requires no power. It is durable and largely weather proof except in the face of extreme freezing. And it is extremely reliable. So much so that it's failure is vanishingly infrequent. And when it does eventually fail, typically after decades of use and wear, it does so only after providing ample clues that its need for servicing is becoming acute. "Jiggling the key in the lock" is a longstanding meme.

But mostly, it achieves all this in an example of a brilliant tradeoff: We get all this in return for accepting that it's not perfect protection. Is it cryptographically secure? Of course not. Can it be "picked" and defeated by anyone skilled in the art with a few simple lockpicking tools? Yep. Are there sufficient combinations that no one else's key will open it? Nope. A famous hack is to try locks with keys they don't belong to. Sometimes you just get lucky, specifically because the universe of all possible combinations is comparatively small. But the likelihood of any random key working in any random lock is low enough that no one bothers to try.

But, it's exactly that comparatively small universe of possibilities that allows this research to succeed.

Once the audio of a key-insertion has been obtained, SpiKey's inference software gets to work filtering the signal to extract the strong, metallic clicks as the key's ridges hit the lock's pins. The "click" occurs when one of the spring-loaded pins crosses over the top of any of the key's ridges:

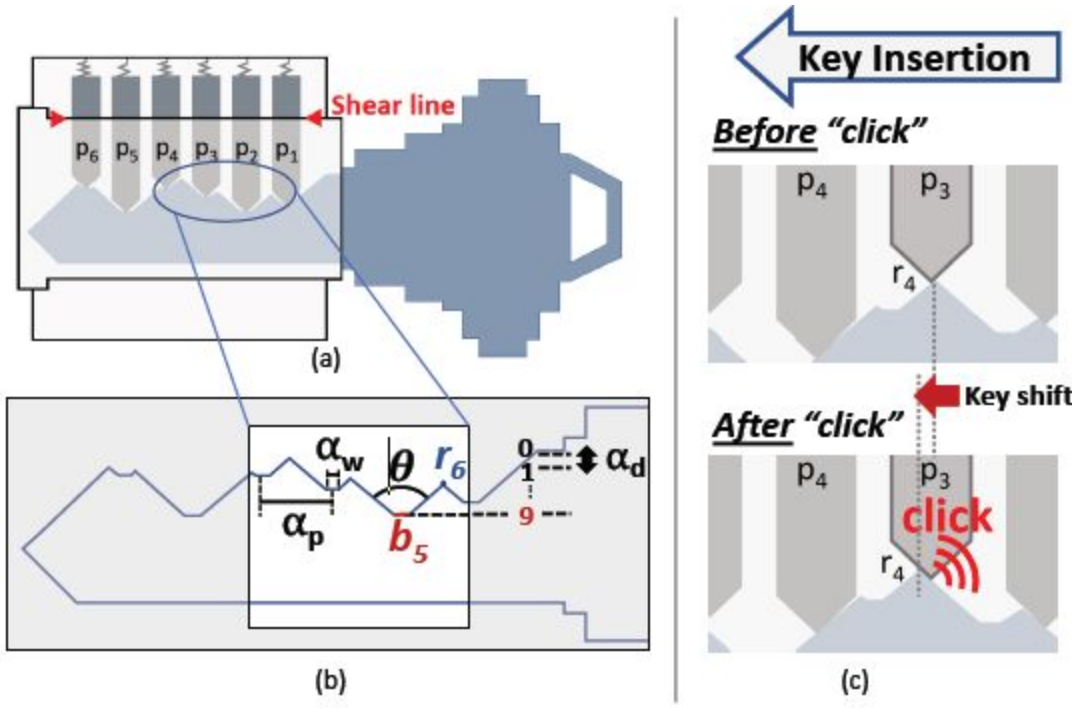


Figure 2: (a) With a correct key inserted, pins align on a shear line and unlocks the lock; (b) depicts key construction parameters; and (c) depicts key insertion producing *click* sound as a pin slips off of a key ridge.

These clicks drive the inference analysis: the time between them allows the SpiKey software to compute the key's inter-ridge distances and, what locksmiths refer to as the "bitting depth" of those ridges, which is how deeply they cut down into the key shaft and where they plateau out.

If a key is inserted at a nonconstant speed, the analysis can be defeated, though the software can compensate for small insertion speed variations. Given all the available acoustic information, complete disambiguity cannot be obtained, and multiple possible keyings will remain. This is why, as the paper's abstract noted, the SpiKey software will output the three most likely key designs to fit the lock used in the audio file, reducing the potential search space from 330,000 keys to just three.

> When a victim inserts a key into the door lock, an attacker walking by records the sound with a smartphone microphone. SpiKey detects the timing of these clicks from the sound. We then utilize the click timestamps to compute the adjacent inter-ridge distances given a constant insertion speed. We use the computed distances to infer the relative differences of adjacent bitting depths, which SpiKey exploits to ultimately obtain a small subset of candidate keys that includes the victim's keycode.

We detect all click events from the audio recording. To provide a better understanding, we posted a video of a corresponding spectrogram of key insertion recording at http://bit.ly/2JciYB6 (https://grc.sc/781). Prior to detecting clicks, we reduce the impact of low-frequency ambient noise, by subjecting it to a high-pass filter, to retain only frequencies above 15kHz that contains

information about the clicks. Subsequently, we identify the starting point of each click, or its onset, in the pre-processed signal by applying change-point detection algorithm on short time-windows around the computed peaks to account for their millisecond granularity. It finds the least sum of standard deviations across two regions that transition from low to high amplitude. We construct a click time series from the obtained click onsets.

https://www.comp.nus.edu.sg/~junhan/papers/SpiKey_HotMobile20_CamReady.pdf

The paper goes on to explain exactly how they convert the click onset timings into a few possible candidate keyings. For any of our listeners who are interested, I have the link to the full research paper in the show notes. But suffice to say, they pulled this off and it's now possible to record the sounds of a key being slid into a lock to recreate the physical ridge and plateau pattern of the key with relatively high confidence.