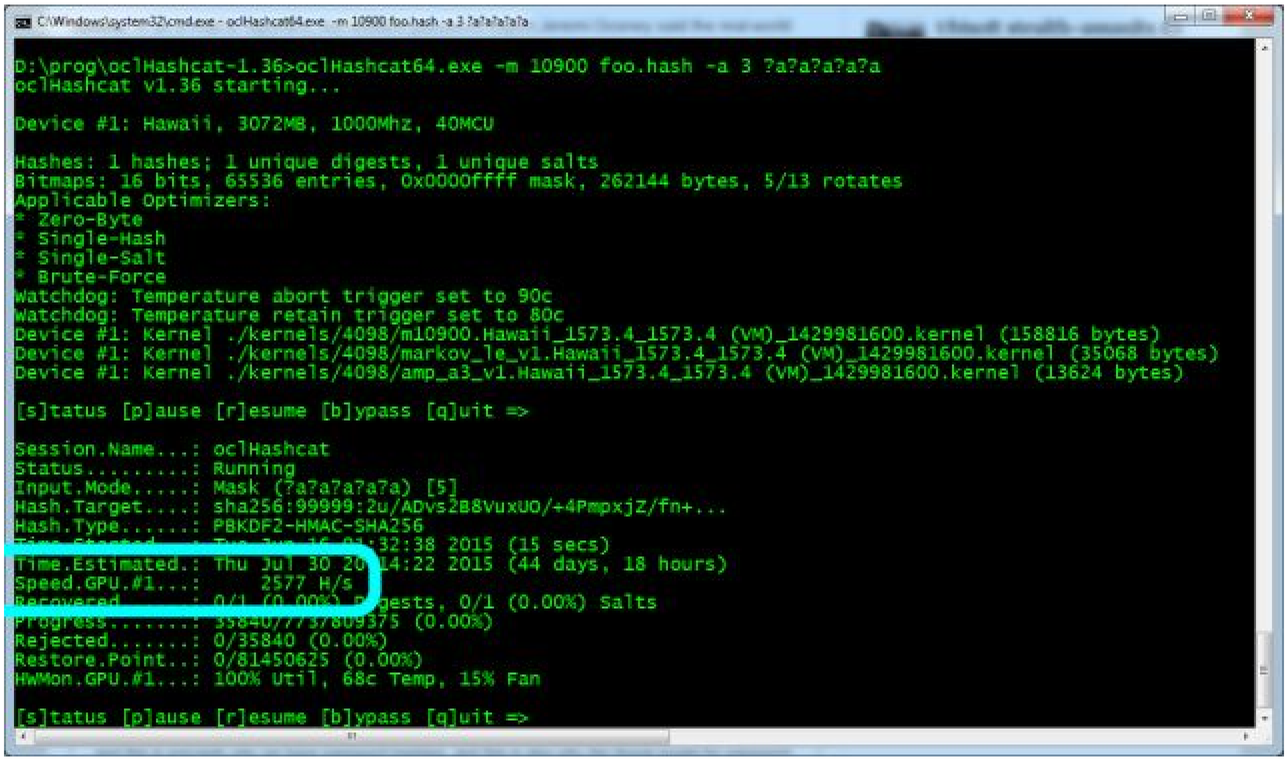
Security Now! #513 - 06-23-15
Q&A 215

This week on Security Now!

- A significant cross-application security flaw in Mac OS X and iOS.
- A worrisome problem in about 600 million Samsung phones.
- Some real-world Lastpass exploit math.
- The future of Web applications.
- We have a "Let's Encrypt" launch date!
- Brief (I promise) Sci-Fi media update,
- Feedback from our terrific listeners!

olcHashCat performing 100,000-iteration PBKDF2-SHA256 @ 2577 guesses/second



C:\Windows\system32\cmd.exe - oclHashcat64.exe -m 10900 foo.hash -a 3 ?a?a?a?a?a

```
D:\prog\oclHashcat-1.36>oclHashcat64.exe -m 10900 foo.hash -a 3 ?a?a?a?a?a
oclHashcat v1.36 starting...

Device #1: Hawaii, 3072MB, 1000Mhz, 40MCU

Hashes: 1 hashes; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Applicable Optimizers:
* Zero-Byte
* Single-Hash
* Single-Salt
* Brute-Force
Watchdog: Temperature abort trigger set to 90c
Watchdog: Temperature retain trigger set to 80c
Device #1: Kernel ./kernels/4098/m10900.Hawaii_1573.4_1573.4 (VM)_1429981600.kernel (158816 bytes)
Device #1: Kernel ./kernels/4098/markov_le_v1.Hawaii_1573.4_1573.4 (VM)_1429981600.kernel (35068 bytes)
Device #1: Kernel ./kernels/4098/amp_a3_v1.Hawaii_1573.4_1573.4 (VM)_1429981600.kernel (13624 bytes)

[s]tatus [p]ause [r]esume [b]ypass [q]uit =>

Session.Name...: oclHashcat
Status.........: Running
Input.Mode.....: Mask (?a?a?a?a?a) [5]
Hash.Target....: sha256:99999:2u/ADvs2B8VuxUO/+4PmpxjZ/fn+...
Hash.Type......: PBKDF2-HMAC-SHA256
Time.Started...: Tue Jun 16 01:32:38 2015 (15 secs)
Time.Estimated.: Thu Jul 30 20:14:22 2015 (44 days, 18 hours)
Speed.GPU.#1...:     2577 H/s
Recovered......: 0/1 (0.00%) Digests, 0/1 (0.00%) Salts
Progress.......: 35840/7737809375 (0.00%)
Rejected.......: 0/35840 (0.00%)
Restore.Point..: 0/81450625 (0.00%)
HWMon.GPU.#1...: 100% Util, 68c Temp, 15% Fan

[s]tatus [p]ause [r]esume [b]ypass [q]uit =>
```

Security News

Unauthorized Cross-App Resource Access on MAC OS X and iOS
http://arxiv.org/pdf/1505.06836v1
On modern operating systems, applications under the same user are separated from each other, for the purpose of protecting them against malware and compromised programs. Given the complexity of today's OSes, less clear is whether such isolation is effective against different kind of cross-app resource access attacks (called XARA in our research). To better understand the problem, on the less-studied Apple platforms, we conducted a systematic security analysis on MAC OS~X and iOS. Our research leads to the discovery of a series of high-impact security weaknesses, which enable a sandboxed malicious app, approved by the Apple Stores, to gain unauthorized access to other apps' sensitive data. More specifically, we found that the inter-app interaction services, including the keychain, WebSocket and NSConnection on OS~X and URL Scheme on the MAC OS and iOS, can all be exploited by the malware to steal such confidential information as the passwords for iCloud, email and bank, and the secret token of Evernote. Further, the design of the app sandbox on OS~X was found to be vulnerable, exposing an app's private directory to the sandboxed malware that hijacks its Apple Bundle ID. As a result, sensitive user data, like the notes and user contacts under Evernote and photos under WeChat, have all been disclosed. Fundamentally, these problems are caused by the lack of app-to-app and app-to-OS authentications. To better understand their impacts, we developed a scanner that automatically analyzes the binaries of MAC OS and iOS apps to determine whether proper protection is missing in their code. Running it on hundreds of binaries, we confirmed the pervasiveness of the weaknesses among high-impact Apple apps. Since the issues may not be easily fixed, we built a simple program that detects exploit attempts on OS~X, helping protect vulnerable apps before the problems can be fully addressed.

What does that mean?

- A malicious app can obtain unauthorized access to other apps' sensitive data including passwords and tokens for iCloud, Mail app and all web passwords stored by Google Chrome.

- 88.6% of 1,612 OS X and 200 iOS apps tested were found "completely exposed" to unauthorized cross-app resource access (XARA) attacks allowing malicious apps to steal otherwise secure data.
- Attacking apps can be created which bypass all of Apple's App Store security checks.
- The team's attack apps were approved by the App Store in January 2015.
-
- Apple was initially notified last October 2014 and asked for six months silence. Four months later, in February 2015, Apple security people asked the researchers for an advanced copy of their paper... and to hold off on their disclosure for ANOTHER six months.
-
- Google's Chromium security team was much more responsive, immediately yanking keychain integration from Chrome. Google's group noted that the problem could not be solved at the application level. There was no safe workaround.

Samsung Keyboard Flaw

- > 600 million devices initially vulnerable including the recently released Galaxy S6.
- If the flaw in the keyboard is exploited, an attacker could remotely:
  - Access sensors and resources like GPS, camera and microphone
  - Secretly install malicious app(s) without the user knowing
  - Tamper with how other apps work or how the phone works
  - Eavesdrop on incoming/outgoing messages or voice calls
  - Attempt to access sensitive personal data like pictures and text messages

- Security Researcher Ryan Welton with "NowSecure":
  - Samsung was notified last December 2014.
  - Disclosed at Black Hat London conference last Tuesday.
  - "Abusing Android Apps and Gaining Remote Code Execution"
  - Released proof-on-concept code:
    - https://github.com/nowsecure/samsung-ime-rce-poc/

- Ryan's Blog posting at "NowSecure"
  - Remote Code Execution as System User on Samsung Phones
  - https://www.nowsecure.com/blog/2015/06/16/remote-code-execution-as-system-user-on-samsung-phones/
  - A remote attacker capable of controlling a user's network traffic can manipulate the keyboard update mechanism on Samsung phones and execute code as a privileged (system) user on the target's phone. This can be exploited in a manner that requires no user interaction — a user does not have to explicitly choose to download a language pack update to be exploited. The Swift keyboard comes pre-installed on Samsung devices and cannot be disabled or uninstalled. Even when it is not used as the default keyboard, it can still be exploited.
  - Swift has an update mechanism to allow new languages to be added or existing languages to be upgraded.
  - GET http://skslm.swiftkey.net/samsung/downloads/v1.3-USA/az_AZ.zip
    - (Note: Plaintext transfer!... not https!)
  - The zip is downloaded and its files extracted... as the very powerful SYSTEM user.
  - The ZIP is protected by a manifest containing the SHA1 hash of the zip... but the manifest is also transferred via http: plaintext... allowing it to be modified.
  - Thus... an executable CAN be provided by the MITM which will be executed at device restart.
  - The files are device-specific, but the Swift update check's "User-Agent" query header provides all the information needed: 'User-Agent': 'Dalvik/1.6.0 (Linux; U; Android 4.4.2; SM-G900T Build/KOT49H)'
  - YouTube video of the exploit:
  - https://www.youtube.com/watch?v=uvvejToiWrY

It is exploitable if the attacker can arrange access to the device's network, such as a wireless hotspot or local network.
Attacker needs to be IN the connection -- MITM -- when the device is either restarted (rebooted) or device to perform updates.
Patch Status Maintained Here:

- https://www.nowsecure.com/keyboard-vulnerability/
- http://bit.ly/sn-sam (Halfway down the page)

Robert Graham did some math

- "Should I panic because Lastpass was hacked?"
- (BlackICE firewall)
- http://blog.erratasec.com/2015/06/should-i-panic-because-lasthash-was.html
- <quote> Maybe, maybe not. Lastpass uses 100,000 iterations in its PBKDF2 algorithm. If you chose a long, non-dictionary password, nobody can crack it. Conversely, if you haven't, then yes, you need to change it.
- 2577 hashes (guesses) per second.
- Consider normal hashes, not the stronger ones used by Lastpass. My desktop can crack 1 billion of those per second. Consider that a password can be built from UPPER and lower case letters, numbers, and punctuation marks -- or about 64 variations per character.

  In this case, a 5 letter password has 1 billion combinations, so a fast computer can guess it in a second. Adding one letter, with it's 64 different possibilities, makes this 64 times harder, meaning it'll take a minute. Another letter (7), and it becomes an hour. Another letter (to 8), and it becomes several days. Another letter (9), and it becomes a year. Another letter (10), and it becomes 64 years. Another letter (11), and it's thousands of years, and another letter (12) and its millions of years.

  Lastpass re-hashes the password 100,000 times, which slows this down dramatically. What I could've hashed in an hour now takes a decade. On the other hand, consider an adversary like the NSA or a hacker with a botnet that controls 100,000 computers, that would speed things back up to the normal rate. But even with 100,000 computers, the NSA won't be able to brute-force a 12 letter password.

  Unfortunately, brute-force isn't the only option. Hackers may instead use a dictionary attack, where they use word lists and common password choices (like GoBroncos!), and then mutate them with common patterns, like adding numbers on to the end. This speeds things up dramatically, making it easy to crack even 12 letter passwords in minutes.

  In between the two are Markov chains, which is sort of like brute-forcing, but which follows the logic humans use to construct passwords. If a password letter is lower-case, it's overwhelmingly likely that the next letter will also be lower case, for example.

  The upshot is that your 12 character password is a lot weaker than you assume. Your passwords not only have to be long, but also fairly random and not based much on dictionary words, and random in ways that Markov chains can't easily guess.

"WebAssembly" (aka WASM) will be a 20x faster bytecode for browsers.

- Google, Apple, Microsoft & Mozilla are partnering!
- asm.js / http://asmjs.org/   (a compiler-friendly subset of JavaScript)
- NaCl (Google's Native code -- executable code sandboxing)
- PNaCl (Google portable native (byte)code)
- Microsoft has .Net (a very mature bytecode system)
- JAVA -- bytecode (JVM)
- https://github.com/WebAssembly
- https://github.com/WebAssembly/design/blob/master/HighLevelGoals.md

- High Level Goals:
    - Define a portable, size- and load-time-efficient binary format to serve as a compilation target which can be compiled to execute at native speed by taking advantage of common hardware capabilities available on a wide range of platforms, including mobile and IoT.
    - Specify and implement incrementally:
        - a Minimum Viable Product (MVP) for the standard with roughly the same functionality as asm.js, primarily aimed at C/C++;
        - an effective and efficient polyfill library for the MVP that translates WebAssembly code into JavaScript in the client so that WebAssembly MVP can run on existing browsers;
        - a follow-up to the MVP which adds several more essential features; and
        - additional features, specified iteratively and prioritized by feedback and experience, including support for languages other than C/C++.
    - Design to execute within and integrate well with the existing Web platform:
        - maintain the versionless, feature-tested and backwards-compatible evolution story of the Web;
        - execute in the same semantic universe as JavaScript;
        - allow synchronous calls to and from JavaScript;
        - enforce the same-origin and permissions security policies;
        - access browser functionality through the same Web APIs that are accessible to JavaScript; and
        - define a human-editable text format that is convertible to and from the binary format, supporting View Source functionality.

Let's Encrypt Launch Plan: Week of September 14, 2015

- https://www.eff.org/deeplinks/2015/06/lets-encrypt-launch-plan-week-september-14-2015
- Let's Encrypt Launch Plan: Week of September 14, 2015

Miscellany:

Matthew Green (@matthew_d_green) / 2:37pm · 22 Jun 2015 · Twitter for iPhone:
"My wife bought me an iPod nano for Father's Day, which is awesome but OH MY GOD ITUNES.

PDP-8 recreation kit on track:

- http://bit.ly/pdp8kit (Then click "Introduction")
- Way more lights than my PDP-8's (Five banks and an instruction decoder)

"Infinite Loop"

- The website:
    - http://loopgame.co/
- For iOS:
    - https://itunes.apple.com/us/app/infinite-loop/id977028266?mt=8

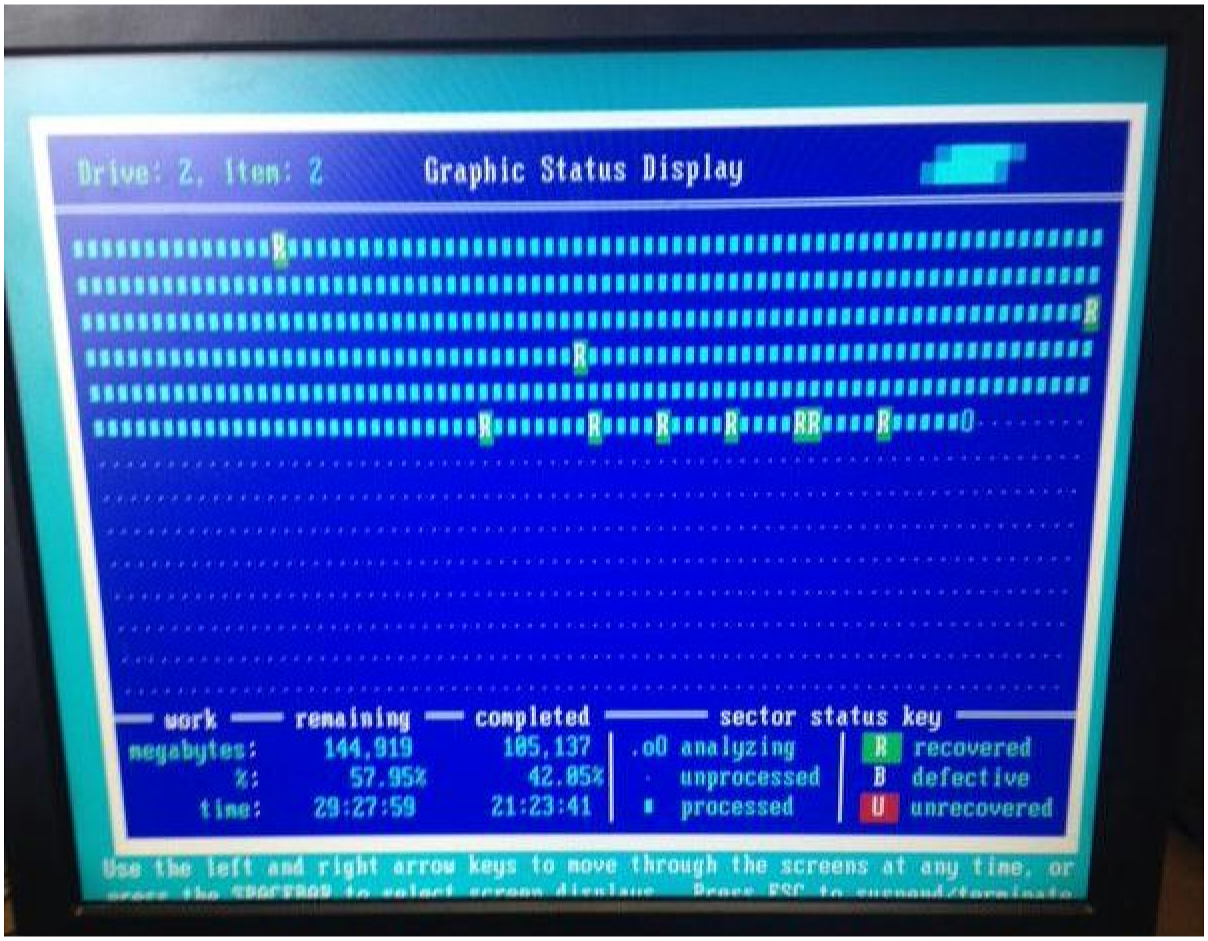
- For Android:
  - https://play.google.com/store/apps/details?id=com.balysv.loop
- About the game:
  - http://lekevicius.com/projects/infinite-loop/

TNSS topic ideas: Thanks SO much!! Wow!!

SyFi (and Sci-Fi)
- Dark Matter was not as bad as I feared from the snippet I saw.
- KillJoys looks like fun too. (Legally sanctioned bounty hunters.)
- "Humans" had a GREAT UK debut... AMC brings it this weekend.
  - IMDB of 8.2/10 @ 1,441 users.
  - In a parallel present where the latest must-have gadget for any busy family is a 'Synth' - a highly-developed robotic servant that's so similar to a real human it's transforming the way we live.

SpinRite:

Peter Butler @satanssnaredrum - 1:35am · 18 Jun 2015 · Twitter for iPhone
@SGgrc hoping to have a testimonial in a few hours. #fingerscrossed being spinwrited soon to be spinwritten pic.twitter.com/tyZ0qpfjLl



```
Drive: 2, Item: 2          Graphic Status Display

— work —— remaining —— completed ———— sector status key ——
megabytes:   144,919     105,137 | .oO analyzing     | R recovered
       %:     57.95%      42.05% |  ·  unprocessed   | B defective
    time:   29:27:59    21:23:41 |  ▪  processed     | U unrecovered

Use the left and right arrow keys to move through the screens at any time, or
press the SPACEBAR to select screen displays.  Press ESC to suspend/terminate
```

Peter Butler @satanssnaredrum - 8:04am · 18 Jun 2015 · Twitter Web Client
@SGgrc Worked a treat... It's a brilliant product you have. Managed to save my ass again. I Messaged your GRC feedback page. :)

From: "Peter Butler"
Subject: Another Spinwrite Testimonial
Date: Thu, 18 Jun 2015 15:01:17 -0000
X-Location: Kildare, Ireland

Hi Steve, I'm a CCTV technician from Kildare, Ireland. I tweeted you a spinwrite pic earlier.

A customer rang me with an error on his CCTV DVR. The DVR had recorded some important footage, but when he was trying to play it back, the DVR was crashing and freezing. The recorder appeared to have written to a corrupt partition and was unable to read it back.

A serious "dumping" incident had taken place which the CCTV would have recorded, so the customer was very anxious to get the footage. So I thought of Spinwrite. I'm an avid listener of your Security now podcast for the past few years and thought I'd give it a go.

I started up Spinwrite on level 2 (for Data recovery) and left it overnight. This morning I returned to work, rebooted the machine and it started with no problem. It was SpinWritten. I was even able to boot into the DVR Software, Review and download the important footage, and give it back to the customer.

Thanks to all at GRC for an amazing product. It's saved my ass multiple times now... Thanks to Steve and Leo for a great podcast too... Makes my commute to work on a Wednesday morning very enjoyable.

Many thanks, Peter