BEHIND THE CURTAIN

INFORMATIKUSOK VS. PENTESTEREK

silent signal

Veres-Szentkirályi András 2021-02-27

$ whoami

Veres-Szentkirályi András

- OSCP, GWAPT, SISE
- Silent Signal alapító
- pentester, toolmaker

Pentester

silent signal

- bizonyos módszertan mellett minden hibát feltár
- nem csak első vérig (vö. bug bounty)

Eldobható kód

silent signal

- Python, Ruby, Perl
- REPL: IPython
- CTF külön kategória
- gyors fejlesztés vs. gyors futás, olvashatóság, karbantarthatóság
- instrumentálhatóság vs. specifikáció követése

Egyedi protokollok kezelése

silent signal

```
decode_packets(<<?OLD_PACKET_FORMAT:2, Tag:4,
               LenBits:2, Body/binary>>, Context) ->
  {PacketData, S2Rest} = case LenBits of
      0 -> <<Length, Obj:Length/binary, SRest/binary>> = Body, {Obj, SRest};
      1 -> <<Length:16, Obj:Length/binary, SRest/binary>> = Body, {Obj, SRest};
      2 -> <<Length:32, Obj:Length/binary, SRest/binary>> = Body, {Obj, SRest}
  end,
  NewContext = decode_packet(Tag, PacketData, Context),
  decode_packets(S2Rest, NewContext).
```

Alternatíva: `http://kaitai.io/` vs.
`https://github.com/kaitai-io/kaitai_struct/issues/27`

Kiterjeszthető toolok

silent signal

	Burp Extender	mitmproxy	Piper
Programming language(s)	JVM: Java, Kotlin, Jython, JRuby …	Python 3	∀
Development cycle	slow	fast	fast
Composability	low	low	high

This matters, since **pentesting is all about improvization and one-off solutions**.

Source code and binaries under GPL:
`https://github.com/silentsignal/burp-piper`
(also in BApp Store)

Language of choice (Piper)

- Jython: worst of both worlds
- JRuby: same as Jython
- Java: I still don't like it
- Kotlin: better syntax at least
- `https://kotlinlang.org/docs/reference/comparison-to-java.html`

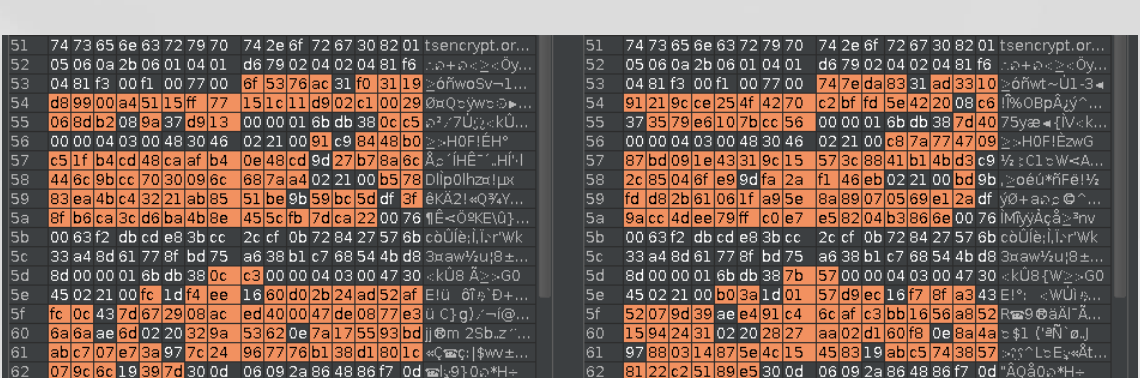
Comparison: Burp text diff

Comparison: Python + git diff

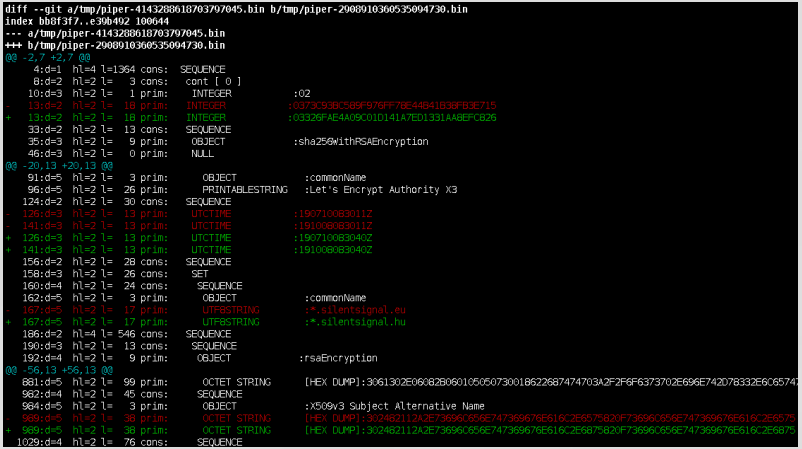
silent signal

```
diff --git a/tmp/piper-8350675374160785262.bin b/tmp/piper-613289896204064887.bin
index a2def11..667609c 100644
--- a/tmp/piper-8350675374160785262.bin
+++ b/tmp/piper-613289896204064887.bin
@@ -2047,7 +2047,7 @@
         "disabled": false,
         "downloads_url": "https://api.github.com/repos/dnet/chrl/downloads",
         "events_url": "https://api.github.com/repos/dnet/chrl/events",
-        "fork": false,
+        "fork": true,
         "forks": 0,
         "forks_count": 0,
         "forks_url": "https://api.github.com/repos/dnet/chrl/forks",
@@ -2268,7 +2268,7 @@
         "node_id": "MDEwOlJlcG9zaXRvcnk0OTE2MDA0",
         "notifications_url": "https://api.github.com/repos/dnet/cmp/notifications{?since,all,participating}",
         "open_issues": 0,
-        "open_issues_count": 0,
+        "open_issues_count": 1,
         "owner": {
             "avatar_url": "https://avatars1.githubusercontent.com/u/163115?v=4",
             "events_url": "https://api.github.com/users/dnet/events{/privacy}",
```

Comparison: Burp binary diff



Comparison: OpenSSL + git diff



Dokumentáció, reprodukálhatóság

silent signal

- bug bounty: nesze itt egy hiba
- teljes körű vizsgálat: bizonyít(ani próbál)om, hogy a rendszer biztonságos
- screenshot: egyszerű készíteni, az alapján újracsinálni nehéz
- „futtatható dokumentáció”: nagyobb befektetés, cserébe reprodukálható

Shell

silent signal

- modern fejlesztő/üzemeltető: fúj, nem típusos, nehezen karbantartható
- REPL before it was cool, interaktív + script egyben
- pipe, subshell: kis blokkokból összelegózunk komplex megvalósítást
- fejből pár gyakori hasznosság:
  - `tr` – karakterhelyettesítés (vö. URL-safe base64), `-d` kapcsolóval törlés (vö. újsorok törlése)
  - `cut` – *n*. oszlop kiválasztása karakterpozíció vagy határolójel alapján
  - `grep` – reguláris kifejezésre szűrés (hasznos: `-o`)
  - `sed` – reguláris kifejezés kiértékelése
  - `awk` – all of the above ;)

radare2

silent signal

- `https://rada.re/`
- Unix filozófia, külön is hasznos toolok
- `rax2` – konverzió számrendszerek között + bájtokra
  - `rax2 -s 414141` ⇒ `AAA`
  - `... | rax2 -S` ⇒ hexpairs
  - utóbbi kettő `base64 -w0` paranccsal kombinálva kiváló
  - `rax2 0x41` ⇒ `65`
- `radiff2` – bináris diff, akár disassemblerrel is
- `rahash2` – hash akár egy fájl egy részéről is

OpenSSL

silent signal

- lib *és* CLI
- konverzió: `x509`, `pkcs12`, `rsa`, `ec`, `pkey`
- titkosítás: `rsautl`, `dgst` (`-sign` és `-verify`), `enc`
- kulcsgenerálás: `genpkey` (új), `genrsa` (régi)
- `s_client` – SSL/TLS kliens
- `req` – CSR ill. self-signed tanúsítvány műveletek
- `asn1parse` – ASN.1 DER dekódolás, javasolt kapcsolók: `-i`, `-strparse`

Frida

- https://frida.re/
- dinamikus instrumentáció: reprodukálható debugger-helyettesítő
- gadget: másik OS, másik arch: https://github.com/frida/frida/releases/
- trace: gyors képet ad érdekes függvényekről, beavatkozást is lehetővé tesz
  - `-a 'Core!0x105ae0'` – ha csak a cím van meg
  - `-i '*rypt*'` – minden ilyen nevű szimbólumra ugrik
  - `-m '+[Curve25519 *:]'` – Objective-C iOS-en
- objection: iOS és Android app hacking – https://github.com/sensepost/objection
- TLS kulcsszerzésre lásd korábbi előadásom
  - https://vimeo.com/425497704
  - https://silentsignal.hu/docs/S2_VSzA_MITM_2020.pdf

Formázás és feldolgozás

- `bat` – syntax highlight: `https://github.com/sharkdp/bat`
- JSON
  - `python -m json.tool` – built-in JSON pretty-printing
  - `gron` – grepelhető JSON: `https://github.com/tomnomnom/gron`
  - `jq` – XPath-szerűség JSON-re
- XML
  - `xmllint --format` – XML pretty-printing
  - `xmllint --xpath` – XPath

Hálózatkezelés

silent signal

- `iptables` – célpont csomagok gyors megszerzése
- `tcpsocks` – furcsa csomagok SOCKS-ba irányítása: `https://github.com/vi/tcpsocks`
- Wireshark + dissectorai ⇒ exportok
  - JSON – szeret dupla kulcsokat csinálni
  - PDML – XML, de legalább well-formed
- Scapy – tetszőleges csomag összerakása és válasz feldolgozása Pythonból
  `https://scapy.net/`

Példák: Wi-Fi

Melyik csatornákon volt AP?

```
$ cut -f 6 -d ';' r.kismet.csv | sort -un
Channel
1
6
11
36
40
44
48

$ xmllint --xpath '//wireless-network/channel/text()' r.kismet.netxml | sort -un
```

Példák: apktool

Smali bájtkód módosítása után fordítás + aláírás + újratelepítés – opcionálisan bővíthető `logcat` futtatással, ami meg `grep`elhető…

```
apktool b --use-aapt2 && (cd dist ; jarsigner -sigalg SHA1withRSA -digestalg SHA1
   -keystore .../objection/utils/assets/objection.jks -storepass basil-joule-bug
    *.apk objection && adb install -r *.apk)
```

Példák: cert pinning konzisztencia

Network Security Config XML-ben felsorolt hash-ek szerepelnek a bájtkódban? Ha igen, hol? (Tipikus példa: OkHttp pinning)

```
$ xmllint --xpath '//pin/text()' res/xml/network_security_config.xml
    | tr '=' '\n' | sort -u | while read i; do echo $i; grep -Frl $i; done
```

Példák: Android KeyStore feloldás

`10152_USRPKEY_pinKey` fájlban lévő AES-256 kulccsal oldható fel a megadott Base64 kódolással Shared Preferencesbe mentett kriptogram (IV mellette ugyanúgy ugyanott)

```
$ base64 -d <<<746nQ5wQYejBNx8Q5Jt4iw== | openssl aes-256-cbc -d
  -K $(dd if=10152_USRPKEY_pinKey bs=1 skip=45 count=32 2>/dev/null | rax2 -S)
  -iv $(base64 -d <<<3tfqSUrQaSvbeH9UvNU5dg== | rax2 -S)
```

Példák: paraszt XML parsing

Nem volt kéznél `xmllint`, és a végén még protobuf dekódolás is kellett:

```
$ grep string tmp/main_preferences.xml  | cut -f 2 -d '>' | cut -f 1 -d '<'
    | head -n1 | rax2 -s
    | protoc --decode=google.crypto.tink.EncryptedKeyset tink.proto
```

Példák: iOS jailbreak detekció

Elegáns minimális változtatás:

```
$ radiff2 -a arm -b 64 -D CensoredID*
--- 0x00129bd0  e00318aa
- mov x0, x24
+++ 0x00129bd0  000080d2
+ movz x0, 0
```

Összefoglalás

- más igények más megoldások
- ha még nem ijesztettünk el, keresünk:
  - junior pentestert
  - senior pentestert
  - C# fejlesztőt

KÖSZÖNÖM!

VERES-SZENTKIRÁLYI ANDRÁS

vsza@silentsignal.hu

facebook.com/silentsignal.hu

@SilentSignalHU

@dn3t

silent signal