Acta Informatica Pragensia
3(1), 2014, 8–22, DOI: 10.18267/j.aip.32
Online: aip.vse.cz

Sekce / Section:
Recenzované stati / Peer-reviewed papers

O čitelnosti příloh datových zpráv v PDF na platformě OS X

On the Legibility of Data Message PDF Attachments on the OS X Platform

Tomáš Zahradnický[1]

[1] Katedra počítačových systémů, Fakulta informačních technologií,
České vysoké učení technické v Praze
Thákurova 9, 166 21 Praha 3

zahradt@fit.cvut.cz

Abstrakt: Informační systém Datových schránek je významným informačním systémem státní infrastruktury. Návrhu takové informačního systému je nutné věnovat patřičnou péči i vzhledem k tomu, že pokud by se v návrhu později objevila trhlina, mohla by mít nedozírné následky na všechny uživatele. Trhlina, o které budeme pojednávat, se týká způsobu zasílání některých datových zpráv obsahujících přílohy ve formátu PDF. Některé z těchto příloh nemusí být příjemci používajícími výchozí internetový prohlížeč Safari na operačním systému OS X schopni správně otevřít. Článek tuto situaci analyzuje a dává odpověď na otázku, kdy bude příloha ve formátu PDF otevřena správně a kdy nikoliv.

Klíčová slova: Datová zpráva, datová schránka, ISDS, PDF, OS X, Safari.

Abstract: The Data Message Information System is a remarkable information system of the state infrastructure. Design of an information system of such importance should be done with much care also due to fact that if a design flaw appeared later, it could have severe impacts at the users. A flaw that will be discussed in this paper applies to a data message sending process of messages containing PDF attachments. Users with the default Safari web browser on OS X do not need to be always able to open such attachments. The paper analyses the situation and gives an answer to a question when will a PDF attachment be opened correctly and when not.

Keywords: Data Message, Data Mailbox, DMIS, PDF, OS X, Safari.

1 Úvod

Informační systém (IS) Datových schránek (ISDS) byl zřízen zákonem č. 300/2008 Sb. a byl uveden do provozu dne 1. 11. 2009. Správu ISDS zajišťuje Ministerstvo vnitra České republiky (MVČR) a provoz Česká pošta, s.p. ISDS je denně využíván orgány státní správy, samosprávy i desítkami tisíc dalších subjektů. Primární funkcí systému je odesílání a příjem datových zpráv, které mohou obsahovat přílohy. Podporované přílohy datových zpráv jsou uvedeny v příloze 3. vyhlášky MVČR č. 194/2009 Sb. ve znění pozdějších úprav. Informační systém takovýchto rozměrů musí být navržen perfektně ve všech směrech, i vzhledem k tomu, že pokud by se objevila v jeho návrhu později trhlina, mohla by mít zásadní následky. Autor tohoto článku se setkal s případem, kdy ISDS v rámci datové zprávy zaslané přes tento systém nesprávně identifikoval PDF soubory obsahující žaloby, v důsledku čehož se některé soubory žaloby jevily na počítači s operačním systémem OS X, do kterého byly staženy, jako v jednom případě čitelné a ve druhém nikoliv. Analýza této skutečnosti a její příčiny jsou předmětem tohoto článku.
Následující kapitola provede počáteční analýzu komponentů komunikace mezi webovým prohlížečem Safari a ISDS. Z analýzy vyplynou požadavky na návrh metod pro monitorování komunikace a operačního systému s cílem určení příčiny výše uvedeného problému. Následně budou navržené metody aplikovány, bude zachycena vzájemná komunikace serveru ISDS s webovým prohlížečem Safari a bude zjištěno, jakou roli hraje v procesu operační systém a jeho součásti. Zachycená komunikace bude vyhodnocena a bude určena příčina problému.

2 Předběžná analýza problému

Autor textu má k dispozici ve své datové schránce dva rozsudky jednoho ze soudů v Praze vydaných v letech 2013 a 2014, které byly zaslány jako přílohy datových zpráv ve formátu PDF. Po přihlášení do ISDS, otevření této datové zprávy a následném stažení PDF dokumentu s rozsudkem bylo pozorováno nestandardní chování prohlížeče Safari, který je výchozím prohlížečem na platformě OS X. Aby bylo možné zjistit, proč k tomuto chování dochází, bude nutné provést analýzu komunikace mezi prohlížečem Safari a ISDS, a dále mezi Safari a částmi operačního systému, a to prostřednictvím metod reverzního inženýrství v souladu s § 66 odst. 1. písm. d) zákona č. 121/2000 Sb. ve znění pozdějších úprav.

2.1 Analýza komunikace s ISDS

Uživatel komunikuje s ISDS prostřednictvím svého webového prohlížeče. Na platformě OS X je výchozím prohlížečem webový prohlížeč Safari, který je instalován spolu s operačním systémem. Vzhledem k tomu, že Safari je značně sofistikovaná aplikace, jejíž činnost je rozprostřena mezi několik vzájemně komunikujících procesů, bude nejdříve nutné identifikovat proces, který zajišťuje komunikaci prohlížeče se serverem ISDS. Tento proces bude v dalším označován jako komunikační proces (KP). Po identifikaci KP bude nutné se zaměřit na jeho komunikaci s ISDS, kterou bude nutné zachytit v čitelné podobě, a následně ji analyzovat. Nakonec bude nutné se zaměřit na roli operačního systému OS X a některých jeho komponentů, aby bylo možné potvrdit anebo vyvrátit jejich spoluúčast na problému.

2.1.1 Identifikace komunikačního procesu

Komunikační proces (KP) bude možné identifikovat pomocí nástroje `fs_usage` představující standardní systémový nástroj příkazové řádky operačního systému OS X. Tento nástroj vypisuje až do doby jeho ukončení informace o systémových voláních prováděných ze

všech běžících procesů, která se týkají aktivit souborového systému a síťového stacku. Z výpisů tohoto nástroje bude možné určit jméno KP.

2.1.2 Monitorování komunikace mezi ISDS a KP

ISDS vyžaduje použití spojení zabezpečeného na transportní vrstvě ISO/OSI síťového modelu protokolem Transport Layer Security verze 1.0 (Dierks, Allen, 1999) (TLS) anebo Secure Socket Layer verze 3.0 (Freier, Karlton, Kocher, 2011) (SSL). Pro odposlech komunikace těmito protokoly připadají v úvahu následující 2 možnosti:

1. Použít útoku typu Man-in-the-Middle (MITM) (Prowell, Kraus, Borkin, 2010), při kterém by komunikace mezi ISDS a webovým prohlížečem procházela přes prostředníka. Prostředník by musel: i) vstoupit do procesu navazování spojení mezi ISDS a KP tak, že by si s oběma komunikujícími stranami zřídil vlastní šifrovací klíče, ii) dešifrovat komunikaci přicházející z ISDS, zaznamenávat ji a následně ji šifrovat klíčem zřízeným mezi ním a KP a iii) dešifrovat komunikaci přicházející z KP, zaznamenávat ji a následně ji šifrovat klíčem zřízeným mezi ním a ISDS. Tuto metodu by bylo možné realizovat například s nástrojem Ettercap (Ornaghi et al., 2013) anebo Mitmproxy (Cortesi, Hils, 2013).
2. Využít skriptovatelnosti nástroje pro ladění aplikací LLDB (LLVM Developer Group, 2014) a v něm zajistit, aby při přijímání a odesílání dat byla tato data zapisována navíc například do konzole ladícího nástroje, anebo do externího souboru.

První z uvedených možností vyžaduje provedení útoku MITM a do komunikace mezi ISDS a KP je nutné aktivně zasahovat. Aby byly vyloučeny jakékoliv možné změny dat aktivním zásahem do komunikace, bude upřednostněno získání dat bez aktivní účasti prostředníka v podobě takové, v jaké přicházejí ze serveru ISDS do KP a naopak. Proto bude rozpracována druhá z výše uvedených možností, bude navržena, implementována a aplikována metoda zachycující komunikaci mezi ISDS a KP přímo z prostředí nástroje pro ladění aplikací LLDB.

2.2 Analýza chování prohlížeče Safari ve vztahu k operačnímu systému

Prohlížeč Safari využívá služeb operačního systému v podobě volání knihovních funkcí z nejrůznějších systémových frameworků. Bude tedy nutné analyzovat, jakým způsobem do procesu stahování souboru vstupuje operační systém a jeho součásti a hrají-li v tomto procesu svoji roli. Toto bude možné zjistit opět nástrojem pro ladění aplikací.

2.3 Shrnutí předběžné analýzy problému

Aby bylo možné provést analýzu chybného stažení souboru z ISDS, bude nutné nejdříve určit, který proces z množiny procesů webového prohlížeče Safari na platformě OS X provádí komunikaci se serverem ISDS. Tento proces bude následně analyzován pomocí nástroje pro ladění aplikací LLDB a pomocí skriptů v LLDB bude chování KP přizpůsobeno tak, aby navíc prováděl monitorování komunikace mezi webovým prohlížečem a ISDS. Tato komunikace bude zaznamenávána a bude následně posouzena na možné příčiny chybného stažení. Nakonec bude nutné zjistit, nehraje-li v procesu stahování svoji roli také operační systém či některá jeho součást, což bude zjištěno rovněž pomocí nástroje LLDB. V následující kapitole bude proveden návrh metod vedoucích ke zjištění potřebných informací.

3 Návrh metod

V této kapitole bude uveden návrh metod vedoucích k určení KP, jeho následnému přizpůsobení k monitorování dat, pomocí nástrojů pro ladění počítačového softwaru. Dále bude rozpracována metoda analýzy komponent operačního systému, které se mohou také podílet na zjištěné chybě.

3.1 Zjištění komunikačního procesu

Činnost webového prohlížeče Safari je rozdělena mezi několik nezávislých vzájemně komunikujících procesů, a to převážně z bezpečnostních důvodů. Jednotlivé panely webových stránek v oknech prohlížeče jsou umístěny každá do svého vlastního procesu `com.apple.WebKit.WebContent` (`WebContent`). Takové rozdělení má dobré důvody, mezi něž patří i použití zásuvných modulů, jejichž pád by způsobil pád prohlížeče. Pokud dojde k pádu zásuvného modulu, dojde k pádu jediné aplikace `WebContent` a nedojde tak k pádu prohlížeče. Mimo procesů `WebContent` však existují i další procesy, které jsou v důsledku spuštění prohlížeče Safari spuštěny, a proto není na první pohled zřejmé, který proces zajišťuje vlastní komunikaci mezi prohlížečem a ISDS. Vzhledem k tomu, že pro síťovou komunikaci se musí na platformě OS X používat knihovní funkce BSD soketů a každý soket je reprezentován popisovačem souborů (angl. file descriptor), je možné pro identifikaci KP použít standardní konzolový nástroj `fs_usage`, který vypisuje přístup ke všem popisovačům souborů, tedy i těm, které reprezentují jednotlivé síťové sokety. Nástroj bude vhodné použít s parametry `-w` pro získání širokého výstupu, a dále `-f network` pro výpis síťově specifických informací. Spuštění proběhne z prostředí aplikace `Terminal.app` příkazem: `sudo fs_usage -w -f network` a po dobu běhu nástroje `fs_usage` bude z webového prohlížeče Safari navázáno připojení k webovému serveru ISDS. Nástroj `fs_usage` vypíše do konzole aplikace `Terminal.app` mj. i informaci o tom, které procesy operačního systému komunikovaly pomocí síťových soketů a mezi těmito procesy musí být i hledaný KP.

3.2 Návrh metody monitorování komunikačního procesu

Po identifikaci KP bude použit nástroj pro ladění aplikací LLDB (LLVM Developer Group, 2014). Tento nástroj bude spuštěn z prostředí aplikace `Terminal.app` po dobu běhu KP. Po spuštění LLDB bude použit jeho příkaz `attach`, kterým dojde k připojení nástroje LLDB ke KP a k přerušení běhu KP. Pro komunikaci protokolem TLS se na OS X používají funkce `SSLRead` pro příjem dat a funkce `SSLWrite` pro odesílání dat. Ty provádějí šifrování a dešifrování dat, která přijímají a odesílají prostřednictvím soketových funkcí. Prototypy funkcí `SSLRead` a `SSLWrite` jsou následující (Apple, 2014a):

```
OSStatus SSLRead (
     SSLContextRef context,
     void *data,
     size_t dataLength,
     size_t *processed
  );
```

```
OSStatus SSLWrite (
     SSLContextRef context,
     const void *data,
     size_t dataLength,
     size_t *processed
  );
```

Obě funkce vrací datový typ `OSStatus`, kterým informují o výsledku příjmu (`SSLRead`) anebo odesílání dat (`SSLWrite`). Parametr `context` označuje, o které TLS spojení jde, parametr `data` obsahuje u příjmu dat ukazatel na blok, do kterého se uloží přijímaná data, u odesílání dat ukazatel na blok, který se bude odesílat. Parametr `dataLength` určuje velikost bloku dat a parametr `processed` obsahuje ukazatel, do kterého funkce uloží množství zpracovaných dat.

Dokumentace (Apple, 2011) definuje, kde tyto parametry nalezneme v okamžicích vstupu a výstupu z funkcí `SSLRead` a `SSLWrite`. V 64bitové architektuře x86-64 jsou parametry při volání funkcí postupně ukládány do registrů procesoru `rdi`, `rsi`, `rdx` a `rcx`. Parametr `context` bude tedy umístěn v registru `rdi`, parametr `data` v registru `rsi`, parametr `dataLength` v registru `rdx` a parametr `processed` v registru `rcx`. Na funkce `SSLRead` a `SSLWrite` bude nutné nastavit body přerušení programu (angl. breakpoint), a to pomocí příkazů:

```
(lldb) breakpoint set -b SSLWrite
(lldb) breakpoint set -b SSLRead
```

Nastavení bodů přerušení na tyto funkce způsobí, že dojde k přerušení běhu KP v případě, že se bude pokoušet odeslat anebo přijmout data ze šifrovaného spojení s ISDS. Pouhé přerušení je pro naše účely nedostatečné. V okamžiku přerušení funkce `SSLWrite` jsou odesílaná data v čitelné podobě k dispozici a ukazuje na ně registr `rsi`. Toto však neplatí u funkce `SSLRead`, u které jsou přijímaná data k dispozici až po jejím úspěšném dokončení. Nástroj LLDB je skriptovatelný a je možné ke každému bodu přerušení přiřadit skript, který se automaticky vykoná při přerušení programu. Předmětem následujících dvou sekcí bude návrh skriptů, které zřídí odposlech odesílání dat funkcí `SSLWrite` a přijímání funkcí `SSLRead`.

3.2.1 Metoda odposlechu u funkce SSLWrite

Funkce `SSLWrite` slouží pro odesílání dat a prostřednictvím ní KP zasílá data ISDS. Data v čitelné podobě jsou v okamžiku volání funkce k dispozici a ukazuje na ně ukazatel ve druhém parametru funkce `SSLWrite`, který je obsažený v registru `rsi`. Pro naše potřeby je postačující vytisknout pouze prvních 1024 bajtů dat do konzole ladícího nástroje a nechat KP následně pokračovat bez další interakce. Požadovaného chování lze dosáhnout pomocí nastavení následujícího skriptu k bodu přerušení u funkce `SSLWrite`:

```
(lldb) breakpoint command add 1
  Enter your debugger command(s).  Type 'DONE' to end.
  > memory read -c 1024 $rsi
  > process continue
  > DONE
```

Příkaz **`breakpoint command add 1`** přiřadí k bodu přerušení č. 1, který byl nastaven na funkci `SSLWrite`, skript obsahující tři příkazy:

1. **`memory read`** vypisuje prvních 1024 bajtů dat odesílaného bloku do konzole aplikace LLDB – data se vytisknou v aplikaci `Terminal.app`;
2. **`process continue`** instruuje nástroj LLDB k tomu, aby obnovil běh procesu KP, který po jeho provedení bude automaticky pokračovat bez další interakce;
3. **`DONE`** ukončuje zadávání skriptu.

Chování KP je tímto nastavením přizpůsobeno tak, že funkce `SSLWrite` vypisuje díky přiřazení skriptu prvních 1024 bajtů odesílaných dat v čitelné podobě do konzole aplikace LLDB.

3.2.2 Metoda odposlechu u funkce SSLRead

Funkce `SSLRead` slouží pro příjem dat odesílaných z ISDS směrem do KP. V době volání funkce `SSLRead` z KP však data nejsou ještě známa. KP předává funkci `SSLRead` ukazatel na

vyhrazené místo v paměti, do kterého má funkce uložit dešifrovaná data z TLS spojení s ISDS. Odposlech dat z funkce `SSLRead` je nutné rozdělit do 2 kroků, a to: i) zapamatování si ukazatele na vyhrazené místo, do kterého mají být přijmutá data uložena, který je určen argumentem `data` a nachází se v době přerušení programu v registru `rsi` a ii) tisk přijmutých dat v okamžiku opouštění funkce `SSLRead`.
Prvním krokem je zapamatování si adresy ukazatele na místo v paměti, do kterého budou uložena přijímaná dešifrovaná data z ISDS. Toho lze dosáhnout opět použitím přiřazení skriptu k bodu přerušení č. 2:

```
(lldb) breakpoint command add 2
Enter your debugger command(s).  Type 'DONE' to end.
> expression long $readaddr=$rsi
> process continue
> DONE
```

Příkaz **`breakpoint command add 2`** přiřadí k bodu přerušení č. 2, který byl nastaven na funkci `SSLRead`, skript obsahující tři příkazy:

1. **`expression long $readaddr=$rsi`** uloží obsah registru `rsi` do globální proměnné označené jako `$readaddr`;
2. **`process continue`** instruuje nástroj LLDB k tomu, aby obnovil běh procesu KP, který po jeho provedení bude automaticky pokračovat bez další interakce;
3. **`DONE`** ukončuje zadávání skriptu.

Výše uvedený skript zajistí uložení hodnoty registru `rsi` v době vstupu do funkce v době, kdy registr `rsi` obsahuje ukazatel na datový blok, do kterého se budou přijímat dešifrovaná data z TLS spojení. Skript hodnotu registru `rsi` uloží do globální proměnné `$readaddr`. Vzhledem k tomu, že data dosud nejsou známa, bude nutné s jejich tiskem vyčkat až na okamžik dokončení funkce.
Druhým krokem je tisk přijatých dat do konzole aplikace `Terminal.app`, který bude proveden v době, kdy jsou data již k dispozici – na poslední instrukci funkce `SSLRead`. Pro tyto účely bude nutné zjistit, kde funkce `SSLRead` končí a na poslední její instrukci nastavit třetí bod přerušení. Konec funkce `SSLRead` lze nalézt například pomocí příkazu:

```
(lldb) disassemble -n SSLRead
Security`SSLRead:
0x7fff8b57eab8:  pushq  %rbp
0x7fff8b57eab9:  movq   %rsp, %rbp
0x7fff8b57eabc:  pushq  %r15
...
0x7fff8b57ed31:  popq   %r14
0x7fff8b57ed33:  popq   %r15
0x7fff8b57ed35:  popq   %rbp
0x7fff8b57ed36:  ret
```

Z výpisu vyplývá, že poslední instrukcí funkce `SSLRead` je instrukce `ret`, nacházející se na adrese `0x7fff8b57ed36`. Na tuto adresu bude nutné nastavit třetí bod přerušení příkazem:

```
(lldb) breakpoint set -a 0x7fff8b57ed36
```

Nyní je nutné k bodu přerušení č. 3 přiřadit poslední skript:

```
(lldb) breakpoint command add 3
```

```
Enter your debugger command(s).  Type 'DONE' to end.
> memory read -c 1024 $readaddr
> process continue
> DONE
```

Příkaz **`breakpoint command add 3`** přiřadí k bodu přerušení č. 3, který byl nastaven na funkci `SSLRead`, skript obsahující tři příkazy:

1. **`memory read`** vypisuje prvních 1024 bajtů dat přijímaného bloku, na který ukazuje globální proměnná `$readaddr`, do konzole aplikace LLDB – data se vytisknou v aplikaci `Terminal.app`;
2. **`process continue`** instruuje nástroj LLDB k tomu, aby obnovil běh procesu KP, který, po jeho provedení, bude automaticky pokračovat bez další interakce;
3. **`DONE`** ukončuje zadávání skriptu.

3.2.3 Shrnutí k navrženým metodám

Výše navržená metoda zajistí pomocí bodů přerušení programu nastavených na komunikačních funkcích `SSLRead` a `SSLWrite` automatické vykonávání skriptů provádějících monitorování TLS komunikace do konzole aplikace `Terminal.app`. Skripty, které byly přiřazeny k funkcím `SSLRead` a `SSLWrite`, vypisují pro jednoduchost pouze prvních 1024 bajtů z každého datového bloku zabezpečené komunikace. Navržená metoda používá globální proměnnou `$readaddr`, do které ukládá adresu paměťového bloku, což nemusí být vhodné u vícevláknových aplikací. Tento problém by však bylo možné snadno odstranit použitím složitějších skriptů, které by ukládaly uchovávanou hodnotu registru `rsi` a velikost bloku například do tabulky, do níž by se mohlo indexovat například pomocí identifikátoru vlákna. Takové řešení by bylo možné vytvořit prostřednictvím jazyka Python, který rovněž může být použit pro tvorbu skriptů v nástroji LLDB. Pro účely této analýzy však plně postačuje navržená metoda a v následující kapitole budou prezentovány pomocí ní získané výsledky.

4 Komunikace webového prohlížeče s ISDS

Metoda, která byla navržena ve 3. kapitole, byla aplikována na spojení KP s ISDS a dosažené výsledky budou nyní prezentovány. Po uvedení všech nástrojů, které byly použity, budou uvedeny kroky, které byly provedeny před samotným záznamem komunikace. Poté bude následovat prezentace požadavku KP na ISDS o stažení přílohy v PDF následovaná odpovědí ISDS. Komunikace KP s ISDS bude následně vyhodnocena a bude dána odpověď na to, proč může být v jednom případě dokument zobrazen uživateli korektně a ve druhém nikoliv.

4.1 Použité nástroje

Pro záznam komunikace a následné analýzy byl použit:

1. počítač MacBookPro8,2;
2. operační systém OS X verze 10.9.2 (13C64);
3. webový prohlížeč Safari verze 7.0.2 (9537.74.9);
4. nástroj pro ladění aplikací LLDB verze lldb-310.2.36;
5. nástroj pro reverzní analýzu Hex-Rays IDA Pro Advanced verze 6.5.131217 (Hex-Rays, 2014b);
6. zásuvný modul pro nástroj Hex-Rays IDA Pro Advanced Hex-Rays Decompiler (x86) pro IDA Pro verze 1.9.0.131213 (Hex-Rays, 2014a).

4.2 Počáteční stav

Jako výchozí stav pro monitorování komunikace byl zvolen stav těsně před započetím stahování souboru přílohy. Do tohoto stavu se uživatel dostane následováním těchto kroků:

1. provede přihlášení k ISDS na stránkách `https://www.mojedatovaschranka.cz` zadáním svého uživatelského jména, hesla, kontrolního kódu a následným stiskem tlačítka „Přihlásit se“;
2. stiskne potvrzující tlačítko „OK“ v následném okně informujícím ho o tom, že nemá žádné nové datové zprávy;
3. zvolí vybranou datovou zprávu.

Z tohoto stavu budou prováděna další zkoumání. ISDS nabízí v případě autorem zvolené datové zprávy ke stažení přílohu `166117235_0_Korostensky2T13-13,R.PDF`. V dalším budeme zkoumat, jaká komunikace probíhá mezi klientem a serverem při jejím stahování.

4.3 Žádost webového prohlížeče na stáhnutí přílohy z ISDS

Komunikace začíná požadavkem webového prohlížeče na server ISDS. Tento požadavek je zachycen v níže uvedeném výpise a pochází z nástroje LLDB:

```
0x7ffeeac994b0: 47 45 54 20 2f 70 6f 72 74 61 6c 2f 44 53 2f 64  GET /portal/DS/d
0x7ffeeac994c0: 64 3f 6f 70 65 72 3d 45 6e 63 6c 6f 73 75 72 65  d?oper=Enclosure
0x7ffeeac994d0: 26 58 43 53 52 46 3d 38 61 39 37 66 33 39 30 30  &XCSRF=8a97f3900
0x7ffeeac994e0: 32 61 32 37 65 66 61 65 32 64 36 65 37 39 38 37  2a27efae2d6e7987
0x7ffeeac994f0: 61 36 38 65 35 66 33 26 64 6d 49 64 3d 31 36 36  a68e5f3&dmId=166
0x7ffeeac99500: 31 31 37 32 33 35 26 65 6e 63 4e 75 6d 3d 30 20  117235&encNum=0
0x7ffeeac99510: 48 54 54 50 2f 31 2e 31 0d 0a 48 6f 73 74 3a 20  HTTP/1.1..Host:
0x7ffeeac99520: 77 77 77 2e 6d 6f 6a 65 64 61 74 6f 76 61 73 63  www.mojedatovasc
0x7ffeeac99530: 68 72 61 6e 6b 61 2e 63 7a 0d 0a 41 63 63 65 70  hranka.cz..Accep
0x7ffeeac99540: 74 2d 45 6e 63 6f 64 69 6e 67 3a 20 67 7a 69 70  t-Encoding: gzip
0x7ffeeac99550: 2c 20 64 65 66 6c 61 74 65 0d 0a 43 6f 6f 6b 69  , deflate..Cooki
0x7ffeeac99560: 65 3a 20 5f 69 66 3d 30 36 3b 20 49 50 43 5a 2d  e: _if=06; IPCZ-
0x7ffeeac99570: 58 2d 43 4f 4f 4b 49 45 3d 30 33 2d 30 35 61 61  X-COOKIE=03-05aa
0x7ffeeac99580: 64 39 65 35 63 32 62 64 34 33 30 36 38 33 32 64  d9e5c2bd4306832d
0x7ffeeac99590: 39 35 38 38 65 36 63 36 30 38 37 38 3b 20 49 53  9588e6c60878; IS
0x7ffeeac995a0: 44 53 6c 61 73 74 3d 31 33 39 32 32 30 39 35 33  DSlast=139220953
0x7ffeeac995b0: 38 35 37 37 0d 0a 43 6f 6e 6e 65 63 74 69 6f 6e  8577..Connection
0x7ffeeac995c0: 3a 20 6b 65 65 70 2d 61 6c 69 76 65 0d 0a 41 63  : keep-alive..Ac
0x7ffeeac995d0: 63 65 70 74 3a 20 74 65 78 74 2f 68 74 6d 6c 2c  cept: text/html,
0x7ffeeac995e0: 61 70 70 6c 69 63 61 74 69 6f 6e 2f 78 68 74 6d  application/xhtm
0x7ffeeac995f0: 6c 2b 78 6d 6c 2c 61 70 70 6c 69 63 61 74 69 6f  l+xml,applicatio
0x7ffeeac99600: 6e 2f 78 6d 6c 3b 71 3d 30 2e 39 2c 2a 2f 2a 3b  n/xml;q=0.9,*/*;
0x7ffeeac99610: 71 3d 30 2e 38 0d 0a 55 73 65 72 2d 41 67 65 6e  q=0.8..User-Agen
0x7ffeeac99620: 74 3a 20 4d 6f 7a 69 6c 6c 61 2f 35 2e 30 20 28  t: Mozilla/5.0 (
0x7ffeeac99630: 4d 61 63 69 6e 74 6f 73 68 3b 20 49 6e 74 65 6c  Macintosh; Intel
0x7ffeeac99640: 20 4d 61 63 20 4f 53 20 58 20 31 30 5f 39 5f 31   Mac OS X 10_9_1
0x7ffeeac99650: 29 20 41 70 70 6c 65 57 65 62 4b 69 74 2f 35 33  ) AppleWebKit/53
0x7ffeeac99660: 37 2e 37 33 2e 31 31 20 28 4b 48 54 4d 4c 2c 20  7.73.11 (KHTML,
0x7ffeeac99670: 6c 69 6b 65 20 47 65 63 6b 6f 29 20 56 65 72 73  like Gecko) Vers
0x7ffeeac99680: 69 6f 6e 2f 37 2e 30 2e 31 20 53 61 66 61 72 69  ion/7.0.1 Safari
0x7ffeeac99690: 2f 35 33 37 2e 37 33 2e 31 31 0d 0a 41 63 63 65  /537.73.11..Acce
0x7ffeeac996a0: 70 74 2d 4c 61 6e 67 75 61 67 65 3a 20 66 72 2d  pt-Language: fr-
0x7ffeeac996b0: 66 72 0d 0a 52 65 66 65 72 65 72 3a 20 68 74 74  fr..Referer: htt
0x7ffeeac996c0: 70 73 3a 2f 2f 77 77 77 2e 6d 6f 6a 65 64 61 74  ps://www.mojedat
0x7ffeeac996d0: 6f 76 61 73 63 68 72 61 6e 6b 61 2e 63 7a 2f 70  ovaschranka.cz/p
0x7ffeeac996e0: 6f 72 74 61 6c 2f 49 53 44 53 2f 69 6e 64 65 78  ortal/ISDS/index
0x7ffeeac996f0: 3f 65 76 3d 4c 47 31 6b 64 43 77 73 4f 7a 45 32  ?ev=LG1kdCwsOzE2
0x7ffeeac99700: 4e 6a 45 78 4e 7a 49 7a 4e 54 73 77 4c 44 73 77  NjExNzIzNTswLDsw
0x7ffeeac99710: 4f 7a 45 37 4f 7a 73 37 4d 41 25 33 44 25 33 44  OzE7Ozs7MA%3D%3D
0x7ffeeac99720: 34 62 39 39 63 38 38 37 26 58 43 53 52 46 3d 63  4b99c887&XCSRF=c
0x7ffeeac99730: 32 65 66 63 32 38 62 32 63 62 36 37 32 30 32 30  2efc28b2cb672020
0x7ffeeac99740: 32 32 35 66 39 61 39 35 37 62 35 62 33 35 39 0d  225f9a957b5b359.
0x7ffeeac99750: 0a 44 4e 54 3a 20 31 0d 0a 0d 0a 00 00 00 00 00  .DNT: 1.........
```

Výpis 1. Žádost KP na ISDS na stažení souboru `166117235_0_Korostensky2T13-13,R.PDF`.

4.4 Odpověď serveru ISDS na žádost o stažení PDF souboru s přílohou

Na požadavek KP uvedený v sekci 4.3 odpovídá ISDS následovně:

```
0x7ffeea2afc00: 48 54 54 50 2f 31 2e 31 20 32 30 30 20 4f 4b 0d  HTTP/1.1 200 OK.
0x7ffeea2afc10: 0a 44 61 74 65 3a 20 57 65 64 2c 20 31 32 20 46  .Date: Wed, 12 F
0x7ffeea2afc20: 65 62 20 32 30 31 34 20 31 32 3a 35 32 3a 33 32  eb 2014 12:52:32
0x7ffeea2afc30: 20 47 4d 54 0d 0a 53 65 72 76 65 72 3a 20 49 53   GMT..Server: IS
0x7ffeea2afc40: 44 53 0d 0a 4c 61 73 74 2d 4d 6f 64 69 66 69 65  DS..Last-Modifie
0x7ffeea2afc50: 64 3a 20 57 65 64 2c 20 31 32 20 46 65 62 20 32  d: Wed, 12 Feb 2
0x7ffeea2afc60: 30 31 34 20 31 33 3a 35 32 3a 33 32 20 47 4d 54  014 13:52:32 GMT
0x7ffeea2afc70: 0d 0a 43 6f 6e 74 65 6e 74 2d 44 69 73 70 6f 73  ..Content-Dispos
0x7ffeea2afc80: 69 74 69 6f 6e 3a 20 61 74 74 61 63 68 6d 65 6e  ition: attachmen
0x7ffeea2afc90: 74 3b 20 66 69 6c 65 6e 61 6d 65 3d 22 31 36 36  t; filename="166
0x7ffeea2afca0: 31 31 37 32 33 35 5f 30 5f 4b 6f 72 6f 73 74 65  117235_0_Koroste
0x7ffeea2afcb0: 6e 73 6b 79 32 54 31 33 2d 31 33 2c 52 2e 50 44  nsky2T13-13,R.PD
0x7ffeea2afcc0: 46 22 0d 0a 43 61 63 68 65 2d 43 6f 6e 74 72 6f  F"..Cache-Contro
0x7ffeea2afcd0: 6c 3a 20 6e 6f 2d 73 74 6f 72 65 2c 6e 6f 2d 74  l: no-store,no-t
0x7ffeea2afce0: 72 61 6e 73 66 6f 72 6d 2c 70 72 69 76 61 74 65  ransform,private
```

```
0x7ffeea2afcf0: 2c 6d 61 78 2d 61 67 65 3d 30 0d 0a 45 78 70 69  ,max-age=0..Expi
0x7ffeea2afd00: 72 65 73 3a 20 30 0d 0a 4b 65 65 70 2d 41 6c 69  res: 0..Keep-Ali
0x7ffeea2afd10: 76 65 3a 20 74 69 6d 65 6f 75 74 3d 34 2c 20 6d  ve: timeout=4, m
0x7ffeea2afd20: 61 78 3d 35 31 32 0d 0a 43 6f 6e 6e 65 63 74 69  ax=512..Connecti
0x7ffeea2afd30: 6f 6e 3a 20 4b 65 65 70 2d 41 6c 69 76 65 0d 0a  on: Keep-Alive..
0x7ffeea2afd40: 54 72 61 6e 73 66 65 72 2d 45 6e 63 6f 64 69 6e  Transfer-Encodin
0x7ffeea2afd50: 67 3a 20 63 68 75 6e 6b 65 64 0d 0a 43 6f 6e 74  g: chunked..Cont
0x7ffeea2afd60: 65 6e 74 2d 54 79 70 65 3a 20 74 65 78 74 2f 70  ent-Type: text/p
0x7ffeea2afd70: 6c 61 69 6e 0d 0a 0d 0a 00 00 00 00 00 00 00 00  lain............
```

Výpis 2. Odpověď serveru ISDS na požadavek KP. Součástí odpovědi jsou hlavička s výchozím jménem souboru (`Content-Disposition`) s hodnotou `attachment; filename="166117235_0_Korostensky2T13-13,R.PDF"` a hlavička obsahující chybný MIME typ dokumentu (`Content-Type`) s hodnotou `text/plain`.

4.5 Vyhodnocení komunikace

V sekci 4.3 byl uveden výpis žádosti webového prohlížeče Safari odeslaný z KP serveru ISDS. Tento požadavek je standardní kromě požadavku na jazyk, kterým je francouzština, která je aktuálně nastaveným jazykem operačního systému počítače, na kterém pracuje autor článku. V sekci 4.4 byla uvedena odpověď serveru na žádost KP. Nyní bude odpověď serveru ISDS blíže analyzována, nejdříve hlavičky `Content-Disposition` a `Content-Type`, pak bude následovat diskuze o zjištěné chybě.

4.5.1 Hlavičky komunikace

Hlavička `Content-Disposition`, která je definována ve standardu RFC 6266 (Reschke, 2011), má v zaznamenané odpovědi hodnotu: `attachment; filename="166117235_0_Korostensky-2T13-13,R.PDF"`. Hodnota má dvě části: Část prvá – `attachment` – říká webovému prohlížeči, že by měl vyzvat uživatele k uložení souboru, zatímco část druhá – dispoziční parametr `filename` – poskytuje informaci o doporučeném jménu souboru. V zaznamenaném případě se soubor má jmenovat `166117235_0_Korostensky-2T13-13,R.PDF`.

Hlavička `Content-Type`, která je definována ve standardu RFC 2045 (Freed, Borenstein, 1996a), obsahuje tzv. typ internetového média (dříve MIME typ) s informací o tom, jak by měl prohlížeč interpretovat následující data, případně doplněný o informaci o kódování formou nepovinného parametru `charset`. Hlavička obsahuje hodnotu `text/plain`, a nepovinný parametr není uveden. Prohlížeč by tedy měl data chápat jako prostý text s tím, že kódování textu není uvedeno. Dokument RFC 2046 (Freed, Borenstein, 1996b) uvádí, že pokud parametr `charset` chybí, měla by být implikována znaková sada US-ASCII. Tuto implikaci nahrazuje dokument RFC 2616 (Fielding et al., 1999) na kódovou sadu ISO-8859-1 a zároveň uvádí, že některý počítačový software se v případě chybějícího parametru `charset` chová tak, že se pokouší kódování určit sám z obsahu dat. Pravidla pro uvádění parametru `charset` jsou dále změněna dokumentem RFC 6557 (Melnikov, Reschke, 2012), který rozlišuje dva případy: i) pokud lze vyčíst informaci o kódování dokumentu z obsahu dokumentu, pak by hodnota `charset` neměla být uvedena vůbec, aby nezpůsobila případnou nejednoznačnost při určení kódování a ii) mít explicitně uveden parametr `charset` tak, že výchozí hodnota již není zapotřebí.

4.5.2 Chyba při přenosu souboru

Typ internetového média (MIME typ), který byl ISDS odeslán a poskytnut KP, byl `text/plain` bez dalšího určení kódování souboru. Stažený soubor by měl být prohlížečem interpretován jako soubor s prostým textem. Vzhledem k tomu, že soubory PDF obecně

nejsou textovými soubory, jde o špatnou identifikaci typu souboru. Soubory typu PDF mají svůj vlastní MIME typ, který je definován v RFC 3778 (Taft et al., 2004), a to `application/pdf`. Alternativně by bylo možné použít obecnější typ `application/octet-stream`. Žádný z těchto dvou typů internetového média však serverem nebyl poskytnut a namísto něj byl ISDS odeslán typ `text/plain`. V důsledku toho došlo ke spuštění potenciálně nebezpečného (Barua, Shahriar, Zulker, 2011) procesu detekce typu internetového média (Microsoft Corp.; Hemsley, Barth, Hickson, 2010; Zalewski, 2012) s tím, že pokud se detekce nezdaří, MIME typ se nastaví záchranným mechanismem na obecný MIME typ reprezentující binární data `application/octet-stream`. Tímto záchranným mechanismem prohlížeč soubor ponechá „jak je“ a k jeho otevření následně použije mechanismy operačního systému – identifikace aplikace k otevření souboru obvykle podle jeho přípony, pokud nějakou má.
Stahovaný soubor, pro nějž je hlavička odpovědi ISDS uvedena ve Výpise 2, byl i přes špatný MIME typ otevřen v aplikaci pro prohlížení PDF souborů. Autor má k dispozici záznam komunikace, který není díky vázání mlčenlivostí oprávněn publikovat, ve kterém se za jméno souboru v prohlížeči doplnila automaticky přípona `.txt` a soubor PDF tak skončil s dvojí příponou `.pdf.txt`. Operační systém se rozhodl podle poslední přípony, která byla `.txt`, čímž došlo k otevření souboru ve výchozí aplikaci pro zpracování textu, kterou je na OS X aplikace `TextEdit.app`, a soubor se uživateli jevil jako nečitelný. Aby bylo možné blíže analyzovat, proč se jeden PDF soubor otevřel správně a druhý nikoliv, je nutné prozkoumat metodu detekce typu MIME, kterou prohlížeč (anebo operační systém) provádí a která vuústila jednou otevřením souboru v aplikaci pro prohlížení PDF souborů a podruhé v aplikaci `TextEdit.app`, ve které byl uživateli nečitelný.

4.5.3 Analýza detekce typu MIME

Nyní bude analyzován prohlížeč Safari a proces stahování souboru. Safari je postaven na frameworku `WebKit2.framework` (Apple, 2014b), který je vyvíjen pod licencí s otevřeným zdrojovým kódem. Studiem zdrojových kódů frameworku bylo zjištěno, že ke stahování souboru se využívají služby operačního systému, obsažené v systémovém frameworku `CFNetwork.framework` (Apple, 2012). Zdrojový kód aktuální verze frameworku `CFNetwork.framework` není k dispozici, a proto byly použity nástroje Hex-Rays IDA Pro Advanced (Hex-Rays, 2014b), zásuvný modul Hex-Rays Decompiler (x86) (Hex-Rays, 2014a) v kombinaci s nástrojem LLDB pro porozumění chování frameworku a prohlížeče při stahování souboru. Tato činnost byla provedena podle § 66 odst. 1 písm. d) zákona č. 121/2000 Sb. ve znění pozdějších úprav. Bylo zjištěno, že nad prvním datovým blokem stahovaného souboru přílohy je volána metoda frameworku `CFNetwork.framework` s názvem: `CFNetwork`URL-ConnectionClient::sniffAndSendDidReceiveResponse`. Tato metoda má 1 vstupní argument, kterým je pole, obsahující datový typ s prvním datovým blokem stahovaného souboru. Metoda zjišťuje, jsou-li data kódována v některém z kódování jako například Apple MacBinary (Olson et al., 1987), BinHex (Faltstrom, Crocker, Fair, 1994) a další, a dále obsahuje volání metody `CFNetwork`URLResponse::guessMIMETypeCurrent`, která z obsahu prvního bloku přenášeného souboru určuje MIME typ souboru. Funkce `CFNetwork`URLResponse::guess-MIMETypeCurrent` volá nad tímto blokem dat interní funkci `CFNetwork`isAllText`, která hraje klíčovou v určení MIME typu, proto bude podrobně popsána. Následující pseudokód v programovacím jazyce C byl pořízen nástrojem Hex-Rays IDA Pro Advanced a zásuvným modulem Hex-Rays Decompiler (x86):

```
unsigned char __fastcall CFNetwork`isAllText(const __CFData *cfDataRef)
{
  const __CFData *cfDataRef_1;                // Kopie ukazatele na datový blok
  CFIndex         delka_dat_v_bloku;          // Množství dostupných dat v bloku
  CFIndex         zkoumana_delka;             // Množství znaků, které budeme zkoumat
  char*           ukazatel_na_aktualni_znak;  // Ukazatel na aktuální znak bloku
  char*           ukazatel_na_posledni_znak;  // Ukazatel na poslední znak bloku
  char            znak;                       // Aktuální znak z bloku
  unsigned char   vysledek;                   // Výsledek funkce: 0=netextový, 1=text

  cfDataRef_1 = cfDataRef; // Zkopíruj ukazatel na vstupní blok dat
  if ( cfDataRef )         // Je blok dat platný? Pokud ne, skonči s výsledkem 1
  {
    // Zjisti délku datového bloku
    delka_dat_v_bloku = CFDataGetLength(cfDataRef);

    // Nastav zkoumanou délku na 512 znaků
    zkoumana_delka = 512;

    // Máme k dispozici méně než 513 znaků?
    if ( delka_dat_v_bloku < 513 )
      zkoumana_delka = delka_dat_v_bloku; // Ano, zkoumejme jen tolik, co máme

    // Máme alespoň 1 znak ke zkoumání?
    if ( zkoumana_delka > 0 )
    {
      // Zjisti ukazatel na první znak v bloku
      ukazatel_na_aktualni_znak = CFDataGetBytePtr(cfDataRef_1);

      // Zjisti, kde je konec zkoumaného bloku
      ukazatel_na_posledni_znak = zkoumana_delka + ukazatel_na_aktualni_znak - 1;

      // Dokud jsme v bloku, vezmi znak po znaku a zkoumej jejich hodnotu
      while ( ukazatel_na_aktualni_znak <= ukazatel_na_posledni_znak )
      {
        // Načti aktuální znak z bloku
        znak = *ukazatel_na_aktualni_znak;

        // Nastav hodnotu výsledku na 0
        vysledek = 0;

        // Zjisti, je-li znak v definovaných mezích
        if ( znak >= 9 && znak != 11 && znak >= 27 )
        {
          // Přejdi na následující znak v bloku
          ++ ukazatel_na_aktualni_znak;

          // Pokud je hodnota znaku > 31, pokračuj na následující znak
          if ( znak > 31 )
            continue;

          // Znak byl <= 31, prohlaš blok za netextový skočením s výsledkem 0.
        }
        return vysledek;
      }
    }
  }

  // Prohlaš blok za textový skočením s výsledkem 1.
  return 1;
}
```

Výpis 3. Pseudokód funkce `CFNetwork`isAllText`, která je součástí systémového frameworku `CFNetwork.framework`. Tato funkce prozkoumá prvních maximálně 512 bajtů textu na to, neobsahuje-li znaky s hexadecimálními kódy 00-08, 0B, 0E-1A a 1C-1F (Hemsley, Barth, Hickson, 2010). Pokud ani jeden z těchto znaků zkoumaný blok dat neobsahuje, funkce indikuje, že soubor je čistě textový vrácením hodnoty logické 1, v opačném případě funkce končí s hodnotou logické 0 indikujíc netextový obsah.

V případě souboru `166117235_0_Korostensky2T13-13,R.PDF` funkce `CFNetwork`isAll-Text` vrátí hodnotu 0, což znamená, že prvních 512 bajtů souboru nelze považovat za textový soubor. Funkce `CFNetwork`URLResponse::guessMIMETypeCurrent` se následně pokusí ještě

o několik dalších pokusů o identifikaci a nakonec popsaným záchranným mechanismem nastaví detekovaný MIME typ na `application/octet-stream`. Na tuto změnu MIME typu reaguje prohlížeč tak, že ponechá jméno souboru na jménu doporučeném hlavičkou `Content-Disposition`. Soubor je uložen na disk s doporučeným jménem a při poklepání na něj dojde k otevření prohlížeče PDF souborů.
V případě druhého PDF souboru, který autor má k dispozici, ale díky vázání mlčenlivostí ho nemůže publikovat, funkce `CFNetwork`isAllText` vrátí hodnotu 1, což znamená, že prvních 512 bajtů PDF souboru lze považovat za čitelný text a následně je MIME typ ponechán na hodnotě `text/plain`, je přidána přípona `.txt` a soubor je uložen na disk se jménem obsahujícím doporučené jméno souboru a přidanou příponou `.txt`. Soubor nakonec má dvě přípony `.pdf.txt` a při poklepání na něj dojde díky poslední příponě `.txt` k otevření aplikace pro úpravu textu `TextEdit.app` a soubor je jeví uživateli jako nečitelný.

4.5.4 Shrnutí komunikace webového prohlížeče s ISDS

Díky špatnému MIME typu získanému z ISDS došlo ke spuštění detekce MIME typu operačním systémem v jeho frameworku `CFNetwork.framework`. Detekční algoritmus, který byl v konečném důsledku použit (uvedený ve Výpise 3), ve funkci `CFNetwork`isAllText` zkoumal maximálně prvních 512 bajtů souboru. Pokud žádný z těchto bajtů neobsahoval některý z nepovolených znaků (00-08, 0B, 0E-1A a 1C-1F hexadecimálně (Hemsley, Barth, Hickson, 2010)), byl MIME typ nastaven na `text/plain`, soubor byl prohlášen za textový a byla mu doplněna přípona `.txt` navíc ke jménu doporučeném HTTP hlavičkou `Content-Disposition`. Operační systém pro otevření tohoto souboru použil výchozí aplikaci pro zpracování textu `TextEdit.app`, která zobrazila soubor jako nečitelný.
V dokumentovaném případě byl ISDS rovněž ohlášen KP MIME typ souboru `text/plain` a rovněž došlo k detekci MIME typu souboru. V tomto případě však PDF soubor obsahoval komprimovaná data a v prvních 512 znacích se objevily znaky, které funkce `CFNetwork`isAllText` vyhodnotila jako nepřípustné pro text, vrátila hodnotu 0, detekce selhala a MIME typ byl nastaven záchranným mechanismem na `application/octet-stream`. V důsledku toho operační systém ponechal jméno souboru takové, jaké bylo doporučeno hlavičkou `Content-Disposition` a soubor byl korektně otevřen v aplikaci pro prohlížení PDF.
Oba popsané případy mají společný jmenovatel, kterým je špatný MIME typ přijmutý z ISDS. MIME typ `text/plain` je nevhodný pro soubory PDF a v obou uvedených případech způsobil nejednoznačnost v identifikaci MIME typu přijímaného PDF souboru. Operační systém v obou případech spustil vlastní detekci MIME typu, jejíž výsledek byl závislý na obsahu prvních 512 bajtů PDF souboru. Bylo-li prvních 512 bajtů PDF souboru reprezentovatelných jako text, byla souboru přidána automaticky přípona `.txt` a pro uživatele se dále jevil jako nečitelný. V opačném případě detekce textu selhala, byl použit záchranný MIME typ `application/octet-stream`, soubor byl ponechán s doporučeným jménem a k jeho otevření byla použita výchozí aplikace pro prohlížení PDF souborů.

5 Závěr

Článek pojednával o dvou různých případech stahování přílohy datové zprávy ve formátu PDF z Informačního systému Datových schránek (ISDS) prohlížečem Safari na operačním systému OS X. V jednom případě došlo k otevření přílohy v nesprávné aplikaci a jevila se tak pro uživatele nečitelná, protože byla ke jménu souboru přidána dodatečná přípona `.txt`, ve druhém případě byl soubor korektně stažen a zobrazen. Příčina přidání přípony `.txt` byla analyzována, nejdříve formou počáteční analýzy, která provedla identifikaci komponent, které

se na komunikaci webový prohlížeč – ISDS podílely. Následně byla navržena metoda záznamu komunikace mezi komunikačním procesem (KP) webového prohlížeče a ISDS, která byla aplikována. Ze záznamu komunikace mezi ISDS a KP vyplynulo, že z ISDS do KP přicházela špatná identifikace souboru v podobě typu internetového média (typu MIME) v hlavičce `Content-Type`, která měla hodnotu `text/plain`, namísto očekávaného typu PDF souborů, který je `application/pdf`. Dále bylo zjištěno, že v důsledku špatného MIME typu a chybějícího parametru `charset` u tohoto typu došlo k nejednoznačnosti u MIME typu a došlo ke spuštění detekce MIME typu frameworkem operačního systému `CFNetwork.framework`. Tento framework se pokoušel určit, šlo-li skutečně o textový soubor a použil k tomu funkci `CFNetwork`isAllText`, která zkoumala maximálně prvních 512 bajtů souboru. Pokud byly tyto bajty reprezentovatelné jako text, detekovaný MIME typ byl nastaven na `text/plain`, k souboru byla přidána přípona `.txt` a soubor byl otevřen v aplikaci pro úpravu textů `TextEdit.app`, která soubor zobrazila jako uživateli nečitelný. Pokud detekce textu prováděná funkcí `CFNetwork`isAllText` skončila neúspěchem, tj. byl v prvních 512 bajtech PDF souboru objeven netextový znak, byl detekovaný MIME typ nastaven záchranným mechanismem na typ `application/octet-stream`, soubor byl ponechán se jménem doporučeným hlavičkou `Content-Disposition` a k jeho otevření se použilo standardních mechanismů operačního systému, které ho korektně otevřely v prohlížeči PDF souborů.
Žádný počítačový program by neměl „věřit“ vstupním datům, dokud se jejich důkladným prověřením „nepřesvědčil“ o jejich nezávadnosti a platnosti. Ostatní vstupní data by měl prohlásit za neplatná a odmítnout je, viz standardy tvorby bezpečného kódu (Howard, LeBlanc, 2003) a (Seacord, 2013). Skutečnost, že ISDS odeslal KP špatný MIME typ pro PDF soubor poukazuje na fakt, že sám umožnil, aby do něj byla vložena datová zpráva se špatným MIME typem. ISDS tedy důvěřuje některému z vkládajících, že jím uváděné MIME typy příloh jsou správné. Tímto vkládajícím byl například nejméně jeden ze soudů v Praze a nejméně v jednom soudním sporu toto chování způsobilo nežádoucí právní důsledky – průtah a škody.

Poděkování
Zde, na tomto místě, bych chtěl vyjádřit svůj dík soudnímu znalci ing. Viktoru Kotrubenkovi, za jeho cenné připomínky k obsahu tohoto článku.

Seznam použitých zdrojů

Apple, Inc. (2012). *CFNetwork Programming Guide*. Retrieved from https://developer.apple.com/library/mac/documentation/Networking/Conceptual/CFNetwork/CFNetwork.pdf.

Apple, Inc. (2011). *OS X ABI Function Call Guide*. Retrieved from https://developer.apple.com/library/mac/documentation/DeveloperTools/Conceptual/LowLevelABI/Mac_OS_X_ABI_Function_Calls.pdf.

Apple, Inc. (2014a). *Secure Transport Reference*. Retrieved from https://developer.apple.com/library/mac/documentation/Security/Reference/secureTransportRef/secureTransportRef.pdf.

Apple, Inc. (2014b). *The WebKit Open Source Project*. Retrieved from http://www.webkit.org.

Barua, A., Shahriar, H., & Zulker, M. (2011). Server Side Detection of Content Sniffing Attacks. In *22nd International Symposium on Software Reliability Engineering* (pp. 20-29). IEEE.

Cortesi, A., & Hils, M. (2013). *Mitmproxy: A man-in-the-middle proxy*. Retrieved from http://mitmproxy.org/.

Dierks, T., & Allen, C. (1999). *The TLS Protocol Version 1.0*. Retrieved from http://tools.ietf.org/html/rfc2246.

Faltstrom, P., Crocker, D., & Fair, E. (1994). *MIME Content Type for BinHex Encoded Files*. Retrieved from https://tools.ietf.org/html/rfc1741.

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. (1999). *Hypertext Transfer Protocol -- HTTP/1.1*. Retrieved from https://tools.ietf.org/html/rfc2616.

Freed, N., & Borenstein, N. (1996a). *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*. Retrieved from https://tools.ietf.org/html/rfc2045.

Freed, N., & Borenstein, N. (1996b). RFC 2046 - *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*. Retrieved from https://tools.ietf.org/html/rfc2046.

Freier, A., Karlton, P., & Kocher, P. (2011). *The Secure Sockets Layer (SSL) Protocol Version 3.0*. Retrieved from http://tools.ietf.org/html/rfc6101.

Hemsley, G. P., Barth, A., & Hickson, I. (2010). *MIME Sniffing Standard, Living Standard*. Retrieved from http://mimesniff.spec.whatwg.org.

Hex-Rays S.A. (2014a). *Hex-Rays Decompiler: Overview*. Retrieved from https://www.hex-rays.com/products/decompiler/index.shtml.

Hex-Rays S.A. (2014b). *The Interactive Disassembler*. Retrieved from https://www.hex-rays.com/index.shtml.

Howard, M., & LeBlanc, D. (2003). *Writing Secure Code*. Redmond, WA, USA: Microsoft Press.

LLVM Developer Group. (2014). *The LLDB Debugger*. Retrieved from http://lldb.llvm.org/index.html.

Melnikov, A., & Reschke, J. (2012). *Update to MIME regarding „charset“ Parameter Handling in Textual Media Types*. Retrieved from https://tools.ietf.org/html/rfc6657.

Microsoft Corp. (nedatováno). *MIME Type Detection in Windows Internet Exp*lorer. Retrieved from http://msdn.microsoft.com/en-us/library/ms775147.aspx.

Olson, P., Loeb, L., Shapiro/Maug, N., Hagerman, M., Pester, M., & Bond, W. (1987). *The MacBinary II Standard*. Retrieved from http://files.stairways.com/other/macbinaryii-standard-info.txt.

Ornaghi, A., Valleri, M., Escobar, E., & Milam, E. (2013). *A comprehensive suite for man in the middle attacks*. Retrieved from http://ettercap.github.io/ettercap/index.html.

Prowell, S., Kraus, R., & Borkin, M. (2010). *Seven Deadliest Network Attack*s. Burlington, MA, USA: Syngress.

Reschke, J. (2011). *Use of the Content-Disposition Header Field in the Hypertext Transfer Protocol (HTTP)*. Retrieved from https://tools.ietf.org/html/rfc6266.

Seacord, R. C. (2013). *Secure Coding in C and C++*. USA: Addison-Wesley Professional.

Taft, E., Pravetz, J., Zilles, S., & Masinter, L. (2004). *The application/pdf Media Type*. Retrieved from https://tools.ietf.org/html/rfc3778.

Zalewski, M. (2012). *The Tangled Web*. San Francisco, CA, USA: No Starch Press.