BEZIT – 6. cvičení

Radek Janoštík

Univerzita Palackého v Olomouci

21. 3. 2024

Analýza malware

- Většinou nemáme k dispozici zdrojové kódy malware
  - $\Rightarrow$ Není snadné zjistit, co přesně malware dělá

Analýza malware

- Většinou nemáme k dispozici zdrojové kódy malware
  - $\Rightarrow$ Není snadné zjistit, co přesně malware dělá
  - Bude umět ransomware po zaplacení výpalného soubory dešifrovat?
  - Nebo je to „nečestný“ ransomware a i po zaplacení nic nepůjde dělat?

Analýza malware

- Většinou nemáme k dispozici zdrojové kódy malware
  - $\Rightarrow$ Není snadné zjistit, co přesně malware dělá
  - Bude umět ransomware po zaplacení výpalného soubory dešifrovat?
  - Nebo je to „nečestný“ ransomware a i po zaplacení nic nepůjde dělat?
- Máme však k dispozici binární soubory. Co vše mohou obsahovat?
  - Vykonávané instrukce $\Rightarrow$ asembler
  - Statické řetězce $\Rightarrow$ možné klíče
  - Struktura programu, pojmenování funkcí $\Rightarrow$ odhad funkcionality

Analýza malware

- Binární soubor můžeme otevřít v textovém editoru
  - Část dat můžeme „nějak“ interpretovat
  - Většina však „rozsypaný čaj“ (ukázka)

Analýza malware

- Binární soubor můžeme otevřít v textovém editoru
  - Část dat můžeme „nějak“ interpretovat
  - Většina však „rozsypaný čaj“ (ukázka)
- Trochu lépe na tom budeme pomocí hexaeditoru (ukázka)
  - Např. Okteta, PSPad
  - `hexdump -C binarniSoubor`

Analýza malware

- Binární soubor můžeme otevřít v textovém editoru
  - Část dat můžeme „nějak“ interpretovat
  - Většina však „rozsypaný čaj“ (ukázka)
- Trochu lépe na tom budeme pomocí hexaeditoru (ukázka)
  - Např. Okteta, PSPad
  - `hexdump -C binarniSoubor`
- Spustitelný binární soubor je *objektový soubor*
  - $\Rightarrow$ Můžeme zkoumat jeho strukturu
  - `objdump -s soubor` – můžeme vidět jednotlivé sekce

Analýza malware

- Binární soubor můžeme otevřít v textovém editoru
  - Část dat můžeme „nějak“ interpretovat
  - Většina však „rozsypaný čaj“ (ukázka)
- Trochu lépe na tom budeme pomocí hexaeditoru (ukázka)
  - Např. Okteta, PSPad
  - `hexdump -C binarniSoubor`
- Spustitelný binární soubor je *objektový soubor*
  - $\Rightarrow$ Můžeme zkoumat jeho strukturu
  - `objdump -s soubor` – můžeme vidět jednotlivé sekce
- Můžeme se podívat i na vygenerované instrukce a odhadnout co program dělá
  - `objdump -d -M intel soubor`

Analýza malware

- Binární soubor můžeme otevřít v textovém editoru
  - Část dat můžeme „nějak“ interpretovat
  - Většina však „rozsypaný čaj“ (ukázka)
- Trochu lépe na tom budeme pomocí hexaeditoru (ukázka)
  - Např. Okteta, PSPad
  - `hexdump -C binarniSoubor`
- Spustitelný binární soubor je *objektový soubor*
  - $\Rightarrow$ Můžeme zkoumat jeho strukturu
  - `objdump -s soubor` – můžeme vidět jednotlivé sekce
- Můžeme se podívat i na vygenerované instrukce a odhadnout co program dělá
  - `objdump -d -M intel soubor`
- Více v kurzu Operační systémy 1 od dr. Krajči
  - `https://phoenix.inf.upol.cz/~krajcap/courses/2024LS/OS1/tutorial02.pdf`

Obrana malware

- Malware se často brání snadné analýze
  - Odmázání debug symbolů
  - Zbytečné instrukce, které se nikdy nevykonají
  - Důležité řetězce rozděleny a poskládány „bajt po bajtu“

Obrana malware

- Malware se často brání snadné analýze
  - Odmázání debug symbolů
  - Zbytečné instrukce, které se nikdy nevykonají
  - Důležité řetězce rozděleny a poskládány „bajt po bajtu“
- Často modifikují svůj zdrojový kód
  - Morfování
  - Zamezení snadné globální identifikaci (signatura antivirů)

Obrana malware

- Malware se často brání snadné analýze
  - Odmázání debug symbolů
  - Zbytečné instrukce, které se nikdy nevykonají
  - Důležité řetězce rozděleny a poskládány „bajt po bajtu“
- Často modifikují svůj zdrojový kód
  - Morfování
  - Zamezení snadné globální identifikaci (signatura antivirů)
- Obfuskace zdrojových kódů

Obfuskace zdrojových kódu

- = Snaha o znesnadnění čitelnosti zdrojových kódů
  - Jak zdrojových kódů
  - Tak výsledných binárních souborů

Obfuskace zdrojových kódu

- = Snaha o znesnadnění čitelnosti zdrojových kódů
  - Jak zdrojových kódů
  - Tak výsledných binárních souborů
- Ruční i automatizovaný postup
- V praxi se používá např. pro JavaScript, jehož kódy jsou z principu veřejné
  - $\Rightarrow$ snaha znepříjemnit „ukradení skriptů“

Obfuskace příklad

- Mějte funkci v JavaScriptu:

```
1  async function getJsonFromResponse(response) {
2    try {
3      return await response.json();
4    } catch (error) {
5      // console.log("Error in getJsonFromResponse: " + error);
6      return {
7        error: error,
8      };
9    }
10 }
```

Obfuskovaná verze

```
(function(_0x4e81a8,_0x362bb5){var _0x336e98=_0x497a,_0x16dd4c=_0x4e81a8
    ();while(!![]){try{var _0x1f5add=parseInt(_0x336e98(0x1e7))/0x1+
    parseInt(_0x336e98(0x1ee))/0x2+parseInt(_0x336e98(0x1ea))/0x3+-
    parseInt(_0x336e98(0x1ed))/0x4+-parseInt(_0x336e98(0x1ef))/0x5+-
    parseInt(_0x336e98(0x1ec))/0x6+-parseInt(_0x336e98(0x1e8))/0x7*(
    parseInt(_0x336e98(0x1e9))/0x8);if(_0x1f5add===_0x362bb5)break;else
    _0x16dd4c['push'](_0x16dd4c['shift']());}catch(_0x2411eb){_0x16dd4c[
    'push'](_0x16dd4c['shift']());}}}(_0x2897,0x2e502));function _0x2897
    (){var _0x1ad7b1=['json','687816wsdjBC','407972mloVRS','4253641ZTlKR
    ','127085gzVUrW','124255tprjeM','886571RSJXgl','8esKnFB','664380
    gxnMqL'];_0x2897=function(){return _0x1ad7b1;};return _0x2897();}
    function _0x497a(_0x45f9de,_0x5b4fe6){var _0x2897a0=_0x2897();return
     _0x497a=function(_0x497a98,_0x48f08a){_0x497a98=_0x497a98-0x1e7;var
     _0xc31a8d=_0x2897a0[_0x497a98];return _0xc31a8d;},_0x497a(_0x45f9de
    ,_0x5b4fe6);}async function getJsonFromResponse(_0x20607b){var
    _0x198fb4=_0x497a;try{return await _0x20607b[_0x198fb4(0x1eb)]();}
    catch(_0x3d0bff){return{'error':_0x3d0bff};}}
```

Úkol

- Co dělá následující funkce v JavaScriptu? (vlastními silami)

```
function MysteriousFunction(_0x32ad5b,_0x58440c,_0x439511){const
    _0x2c02fc=_0x58440c*_0x58440c-0x4*_0x32ad5b*_0x439511;if(
    _0x2c02fc>0x0){const _0x48d57c=(-_0x58440c+Math['sqrt'](
    _0x2c02fc))/(0x2*_0x32ad5b);const _0x4361ed=(-_0x58440c-Math['
    sqrt'](_0x2c02fc))/(0x2*_0x32ad5b);return[_0x48d57c,_0x4361ed
    ];}else if(_0x2c02fc===0x0){const _0x487850=-_0x58440c/(0x2*
    _0x32ad5b);return[_0x487850];}}
```

- Stáhněte si binární soubor
  `https://apollo.inf.upol.cz/~janostik/slides/bezit/ransom`
  - Zkuste přijít na to, co zhruba může dělat
  - Jaké je heslo pro šifrování?
  - Je tento ransomware „čestný“ a umožní i dešifrovat?

- POZOR! Spouštění pouze na vlastní nebezpečí, můžete přijít o data!!!
  - ⇒ Virtuální stroj s linuxem