Loop

Best weapon for hackers

Thin Ba Shane

http://location-href.com

Contents

- Introduction
- Automation
- Loop
- Looping in Bruteforce, Fuzzing
- Bug hunting with loops
- Researching with loops
- Filter Bypassing with loops
- Non-Alphanumeric with loops
- Conclusion

Introduction

ဒီဆောင်းပါးလေးမှာ Loop တွေနဲ့ Hacking ဘယ်လိုသက်ဆိုင်နေလဲ ၊ ကျနော်တို့ရဲ့ရှေ့ကလူတွေ ဘယ်လိုသုံးကြတယ် ၊ ဘယ်လောက်ထိအကျိုးရှိတယ် စတာတွေကို ရေးသားထားတာဖြစ်ပါတယ်။ Programming နဲ့ Hacking ကြားက တခြားဆက်စပ်နေတဲ့ အချက်တစ်ခုဖြစ်သလို Tool တွေဖန်တီးရမှာလည်း အင်မတန်အသုံးဝင်တယ်ဆိုတာကို သိစေချင်တဲ့ ရည်ရွယ်ချက်နဲ့ရေးသားထားတာဖြစ်ပါတယ်။

ဒီနေရာမှာ တစ်ခုသိထားစေချင်တာလေးတစ်ခုတော့ရှိပါတယ်။ ဘယ် programming language ပဲဖြစ်ပါစေ ၊ အဲ့ဒါက ကျနော်ရွေးချယ်ရမှာမဟုတ်ပါဘူး၊ အသုံးပြုမယ့်သူအပေါ်မှာပဲ မူတည်ပါတယ်။ ဒီဆောင်းပါးကတော့ Beginner level ထက်ပိုချင်ပိုနေပါလိမ့်မယ်။ ဘာကြောင့်လဲဆိုရင် ကျနော့်အနေနဲ့ အကြံဉာဏ်လေးတွေနဲ့ ရှေ့ကလူတွေဘယ်လိုလုပ်တယ်ဆိုတာကိုပါ သိစေချင်တာဖြစ်လို့ စစချင်းလေ့လာနေတဲ့သူတွေ အတွက်တော့ ဟုတ်ချင်မှဟုတ်ပါလိမ့်မယ်။

Automation

Hacking မှာ Intermediate level လို့ဆိုလာပြီဆိုတာနဲ့ Tools တွေဖန်တီးချင်လာကြပါတယ်။ အဓိကရည်ရွယ်ချက်ကတော့ အရမ်းရိုးရှင်းတယ်။ ကျနော်တို့တွေ ဖြစ်စဉ်တစ်ခုကို သိပြီးသားလည်းဖြစ်တယ် ၊ ဒီဖြစ်စဉ်တစ်ခုကိုပဲ အကြိမ်ကြိမ်လုပ်ဆောင်ချင်တဲ့အခါမျိုးတွေမှာ လူကိုယ်တိုင်ထိုင်လုပ်ရတာမျိုးတွေပေါ့။ ဒီလိုအခါမျိုးမှာ ထိုင်လုပ်နေခြင်းဟာ ကြိုးစားတယ် လို့ခေါ်မလား ၊ ဒါမှမဟုတ် ကျနော်တို့ထက် ညက်ရည်ပိုမြင့်တဲ့သူတွေက အလိုလျောက်ဆောင်ရွက်ခြင်း (Automation) ကိုသုံးပြီး လုပ်နေတယ်ဆိုရင် ကျနော်တို့က ဗန်းစကားထဲကလို **ပိန်းတယ်** လို့ပဲပြောရမလားဖြစ်နေတာပေါ့။

အချိန်တွေကုန်တယ် ၊ ပျင်းဖို့ကောင်းတယ် စတဲ့အချက်တွေဖြစ်နေတယ်ဆိုရင်ကို အလိုလျောက်လုပ်ဆောင်နိုင်ဖို့တစ်ခုခုတော့လိုအပ်လာပါလိမ့်မယ်။ သေချာပြန်တွက်ကြည့်မယ်ဆိုရင်လဲ တစ်ခါအပင်ပန်းခံလေ့လာလိုက်တာက အချိန်တွေ၊ လုပ်ဆောင်နိုင်စွမ်းတွေ အများကြီးလျော့သွားတာကိုတွေ့ရမှာပါ။ ဒါကြောင့်လဲ programming ဆိုတာက အခုလိုခေတ်ကာလမှာ အရေးပါ အရာရောက်နေတယ်ဆိုတာ စာဖတ်သူတွေလည်း သိပြီးသားဖြစ်ချင်ဖြစ်ပါလိမ့်မယ်။

Loop

အထက်မှာရေးခဲ့တဲ့ automation မှာဖော်ပြခဲ့သလိုပဲ ကျနော်တို့ ထပ်တလဲလဲလုပ်ဆောင်ရမယ့် အရာတွေကို လုပ်ဆောင်ဖို့အတွက် Programming တွေမှာ Loop လို့ခေါ်တဲ့ Flow control တစ်ခုပါဝင်ပါတယ်။ Programming Language တွေထဲမှာမှ ကိုယ်ဘယ်ဟာကိုကြိုက်လဲ၊ ကိုယ်ဘာကိုရေးတတ်လဲဆိုတဲ့အပေါ် မူတည်ပြီး အမျိုးမျိုးကွဲပြားသွားနိုင်ပေမယ့် ကျနော်ကတော့ ဥပမာပေးလို့ကောင်းမယ့်အပေါ် မူတည်ပြီးသာရေးသွားမှာဖြစ်ပါတယ်။ အကြံပြုချက်အနေနဲ့ဆိုရင်တော့ python လို programming မျိုးကိုပိုမြန်တယ်ဆိုတဲ့ အထောက်ထားမျိုးတွေရှိတာကြောင့် Fuzzing လို မြန်မြန်လုပ်ဆောင်စေချင်တဲ့ အခြေအနေမျိုးမှာတော့ ကျနော်တို့အနေနဲ့ python ကိုရွေးချယ်သင့်တာပေါ့။ ဒါကို ဘယ်လိုသိနိုင်မလဲဆိုရင်တော့ အောက်မှာပြထားတဲ့ လင်ခ့်လိုမျိုး ဆောင်းပါးမျိုးတွေကနေသိနိုင်ပါတယ်။ ကိုယ်တိုင်လည်း စမ်းကြည့်လို့ရပါတယ်။

https://labs.mwrinfosecurity.com/blog/faster-fuzzing-with-python/

Looping in Bruteforce, Fuzzing

ပထမဦးဆုံး ဥပမာအနေနဲ့ Bruteforce ကနေစပြီးဖော်ပြချင်ပါတယ်။ Bruteforce တိုက်တယ် bruteforce တိုက်တယ်ဆိုတာကြားဖူးပြီးသားဖြစ်မှာပါ။ ဒီလိုဆိုရင် bruteforce တိုက်တယ်ဆိုတာကိုအရင်ပြောမယ်။ Bruteforce / Fuzzing ကို အများအားဖြင့် အောက်ကလို ပုံစံ ၂ ခုကိုတွေ့နိုင်ပါတယ်။

- Matching
- Unmatching

Matching

ကျနော်တို့မှာ database သို့မဟုတ် wordlist တစ်ခုရှိမယ် ၊ User Input တစ်ခုလဲရှိမယ်။ db (သို့) list ထဲက data ကို user input ကနေတစ်ဆင့် ထည့်သွင်းပြီး ရလာတဲ့ result မှာ ကိုယ်မြင်ချင်တဲ့ string ကိုရှာတဲ့ ပုံစံဖြစ်ပါတယ်။

Unmatching

ကျနော်တို့မှာ database သို့မဟုတ် wordlist တစ်ခုရှိမယ် ၊ User Input တစ်ခုလဲရှိမယ်။ db (သို့) list ထဲက data ကို user input ကနေတစ်ဆင့် ထည့်သွင်းပြီး ရလာတဲ့ result မှာ ကိုယ်မြင်ချင်တဲ့ result နဲ့ကွဲလွဲနေတာတွေကိုရှာတဲ့ ပုံစံဖြစ်ပါတယ်။

ဥပမာ ဆိုကြပါစို့၊ ကျနော်တို့ Web Application တစ်ခုကို Bruteforce တိုက်ချင်တယ်ဆိုပါတော့။ ကျနော်တို့လိုအပ်မယ့်အရာတွေကို အောက်ကလို list လေးတစ်ခုလုပ်ကြည့်မယ်

- Wordlists - ကြိုတင်ခန့်မှန်းထားတဲ့ username,password တွေစုထားတဲ့ txt file တစ်ခုလိုမယ်
- File Handling – wordlist file ကို အသုံးပြုဖို့အတွက် file handling လုပ်နိုင်မယ့်တစ်ခုခုလိုမယ်
- Loop – wordlist ထဲကဟာတွေ အကုန်လုံးကို အကြိမ်ကြိမ်အခါခါ အလုပ်လုပ်ဖို့အတွက် loop တွေက မဖြစ်မနေလိုပါတယ်
- Matching/Unmatching String - ထွက်လာတဲ့ result မှာ ဘာကိုလိုချင်လဲဆိုတာက matching ၊ ထွက်လာတဲ့ result မှာ ဘာနဲ့မတူတဲ့ဟာဆိုတာက unmatching
- Conditional - ဘာနဲ့မတူလျှင် ၊ ဘာနဲ့တူလျှင် ဆိုတဲ့ အခြေအနေကိုဆုံးဖြတ်ဖို့အတွက် ကျနော်တို့ if လိုမျိုး conditional တစ်ခုလိုမယ်
- Request Handling – Web Application Bruteforce ဖြစ်တဲ့အတွက် Web Server ဆီကို http request ပို့နိုင်မယ့်အရာတစ်ခုလည်းလိုပါတယ်

အောက်မှာ ဒီ list လေးအတိုင်းရေးထားတဲ့ bruteforcer လေးတစ်ခုကို ဖော်ပြထားပါတယ်

Simple Bruteforcer using python

```
import requests

url='http://localhost:4000/login'


file1 = open("C:\Users\ASUS\Desktop\username.txt","r").readlines()
file2 = open("C:\Users\ASUS\Desktop\pass.txt","r").readlines()


file1L=len(file1)
file2L=len(file2)

for i in range(0,file1L):
        for j in range(0,file2L):
                username=file1[i].strip('\n')
                password=file2[j].strip('\n')
                data = {'userName':username,'password':password}
                r=requests.post(url,data)

                results=r.text.find('Invalid')
                if results==-1:
                        print "Found "+username+" "+password
```

မှတ်ချက်။ ။ Burp Suite မှာပါဝင်တဲ့ intruder ကိုသုံးလို့ရပါတယ်။
အပြောင်းအလဲလေးဖြစ်အောင် အပေါ်က code လေးနဲ့ ကျနော် Node Goat
ကိုဖြေတုန်းကရေးထားတာလေးဖြစ်ပါတယ်။ ပိုကောင်းအောင်လုပ်ချင်ရင်တော့ Multi
Processing, Multi Threading, Queue တို့ကိုပါထည့်သုံးလို့ရပါတယ်။ Concept
လေးသိယုံလောက်သာဖြစ်ပါတယ်

http://location-href.com/nodegoat-walkthrough/

Bug Hunting with loops

Software Bug တွေရှာတဲ့အခါမှာလဲ fuzzing loops တွေကိုသုံးကြပါတယ်။
နာမည်အကြီးဆုံးကတော့ American Fuzzy Loop (AFL) ပေါ့။ Production မှာတွေ့ရတဲ့
Software တွေ Web Application တွေမှာ ကျနော်တို့လေ့ကျင့်ခဲ့တဲ့ lab တွေလို မလွယ်ပါဘူး ၊
source code ဆိုရင်လဲအများကြီးဖြစ်နိုင်တယ်၊ Buffer Overflow ဆိုရင် လေ့လာတုန်းကလို
buffer size က 24 လောက်မဟုတ်တော့ဘူး 20000 လဲဖြစ်ချင်ဖြစ်မယ်။ ဒါပေမဲ့ သိထားတဲ့

အသိတစ်ခုတော့ရှိတယ်။ Buffer Overflow သာဖြစ်သွားရင် Program ဟာ crash ဖြစ်သွားတယ်ဆိုတာပေါ့။ ဒီတော့ ထပ်ခါထပ်ခါ တန်ဖိုးတွေတိုးတိုးသွားပြီး program crash တဲ့ထိ input ကိုထည့်သွားမယ်။ ဒါကို Crash point လို့ခေါ်တယ်။ fuzzer တွေဟာ ဒီလိုမျိုးအလုပ်ကိုလုပ်ပါတယ်။ Crash သွားပြီဆိုမှ crash analysis လုပ်တာကတော့ တစ်ပိုင်းပေါ့။

အကောင်းဆုံးဥပမာအနေနဲ့ Heart Bleed ကို AFL (American Fuzzy Loop) သုံးပြီး ဘယ်လိုရှာတွေ့ခဲ့တယ်ဆိုတာကို အောက်က link မှာဖတ်ကြည့်စေချင်ပါတယ်

https://blog.hboeck.de/archives/868-How-Heartbleed-couldve-been-found.html

Web Application တွေမှာရှာတဲ့အခါလည်း ဒီလိုပါပဲ။ ကျနော်တို့သိထားတဲ့ အသိတစ်ခုလိုတယ်။ ဥပမာ SQL Injection ပဲဆိုပါတော့

SELECT * from users where uid='[user_input]'

PHP ရဲ့သဘောတရားအရ single quote ၃ ခုဖြစ်သွားပြီဆို error တက်တယ်ဆိုတာကိုသိထားရမယ်။ ဒါတင်မကပဲ error ကို ပြမယ်ဆိုရင် error မှာ sql function ပါပြန်ပါလာတယ်ဆိုတာကိုပါသိထားရမယ်။ ဒီတော့ single quote သုံးထားမယ်ဆိုရင် single quote, double quote သုံထားမယ်ဆိုရင် double quote ဆိုတဲ့ data တစ်ခုလိုတယ်။

ဥပမာ

- ` single quote
- " double quote
- ` backtrick
- ') single quote & closed parenthesis
- ") double quote and closed parenthesis

ဒီလို data တွေကိုတစ်ခုပြီးတစ်ခုထည့်ကြည့်မယ်။ ထွက်လာတဲ့ result အပေါ်မူတည်ပြီး SQL Injection ပေါက်တယ်၊ မပေါက်ဘူးစသဖြင့် ဆုံးဖြတ်ကြတယ်။ Tool က တစ်ခါတည်းဆုံးဖြတ်တယ်ဆိုရင် Scanner ၊ လူကပဲဆုံးဖြတ်ရတယ်ဆိုရင်တော့ Fuzzer ပေါ့။ Scanner တွေကတော့ false positive များနိုင်ပါတယ်။

Fuzzing လုပ်ဖို့ဆိုရင် Fuzz-db ကတော့ လူသုံးများတဲ့ resource တစ်ခုဖြစ်ပါတယ်။

https://github.com/fuzzdb-project/fuzzdb/

fuzzer အနေနဲ့ဆိုရင်တော့ wfuzz ရှိပါတယ်။ ဒါပေမဲ့ Burp Suite Intruder မှာ professional က Multithread ရတာကြောင့် GUI လဲဖြစ်တော့ ပိုသုံးလို့ကောင်းပါလိမ့်မယ်

https://github.com/xmendez/wfuzz

Researching with loops

Loops တွေနဲ့ကျနော်တို့ research တောင်လုပ်လို့ရတယ်ဆိုတာကိုဖော်ပြချင်ပါတယ်။ Web Application Obfuscation စာအုပ်ထဲမှာပါတဲ့ ဥပမာတစ်ခုနဲ့အရင်ဆုံးဖော်ပြပေးမယ်။

HTML တစ်ခုရဲ့ structure ကို research လုပ်သွားတာကို အရမ်းသဘောကျပါတယ်။ အောက်ကလို markup တစ်ခုပေါ့။

```
<a href="#">Click</a>
```

ဒီ markup မှာ element ပါတယ်၊ attribute ပါတယ်၊ innerHTML ပါတယ်။ ဒါ့အပြင် anchor သာအလုပ်လုပ်မယ်ဆိုရင် Click ဆိုတဲ့ စာသားလေးက underline နဲ့ link အဖြစ်နဲ့ပေါ်နေမှာဖြစ်တယ်။

ဒီအချိန်မှာ researcher က သာမာန်လူနဲ့မတူတဲ့အတွေးတစ်ခုတွေးသွားတယ်။ အောက်ကလိုမျိုးဆိုရင်ရော ဘယ်ဟာကအလုပ်လုပ်မှာလဲပေါ့

- <[something]a href="#">Click
- <a[something]href="#">Click
- <a href[something]="#">Click
- Click
- Click
- Click
- [something]Click
- <a href="#">Click<[something]/a>

- <a href="#">Click</a[something]>

အခဲ့ဒီ နေရာတွေမှာ ဘယ် Character တွေထည့်ရင် HTML ကအလုပ်လုပ်သေးလဲဆိုတာကို သိနိုင်ဖို့အတွက် for loop ကိုအောက်ကလိုသုံးသွားတယ်။

```
<?php
for ($i=0; $i < 255; $i++) {
	$character=chr($i);
	echo '<div>'.$i.'-><a'.$character.'href="http://localhost">Click</a></div>';
}
?>
```

255 ဆိုတာကတော့ ASCII character တွေအကုန်လုံးထည့်ပစ်လိုက်တာပါ။ Unicode ဆိုရင်တော့ 0 to 65535 = 65536 သုံးရမှာဖြစ်ပါတယ်။

https://www.amazon.com/Web-Application-Obfuscation-Evasion-Filters/dp/1597496049

အထက်ပါအကြောင်းအရာအပေါ် အခြေခံပြီး ကျနော် real world ဥပမာတစ်ခုထပ်ပြီးဖော်ပြချင်ပါတယ်။ Orange Tsai ရဲ့ New era of SSRF ကိုကြည့်ဖူးမှာပေါ့။ အခဲ့ဒီမှာ URL parser ကို research လုပ်ထားတာတစ်ခုပါပါတယ်။ မကြည့်ရသေးရင်ကြည့်လိုက်ပါလို့အကြံပြုချင်ပါတယ်။

https://www.blackhat.com/docs/us-17/thursday/us-17-Tsai-A-New-Era-Of-SSRF-Exploiting-URL-Parser-In-Trending-Programming-Languages.pdf

Slide အရ ကျနော်တို့အောက်ကလိုသိရတယ်

```
http://foo@127.0.0.1 @google.com/
```

space လေးတစ်ခုထည့်လိုက်တာနဲ့ URL parser က ဘာတွေဖြစ်ကုန်လဲဆိုတာကို ကျနော်တို့လဲ အလားတူ fuzzing လုပ်လို့ရနိုင်ပါတယ်။ တစ်ခုထပ်သိရတာက platform အကုန်လုံးလိုလိုမှာ လိုက်စမ်းတာကိုလဲတွေ့ရပါလိမ့်မယ်။

ဥပမာ PHP ဆိုရင် ဒီလိုများစမ်းလားလို့ တွေးမိပါတယ် (ကျနော့်အထင်ပါ)

```
for ($i=0; $i < 255 ; $i++) {
        echo chr($i);
        $URL="https://127.0.0.1:80".chr($i)."@google.com";
        $url=parse_url($URL);
        echo $i."->".$URL."<br>";
        var_dump($url);
        echo "<br>";
}
```

Filter Bypassing with loops

ဥပမာအနေနဲ့ filter bypass လုပ်တဲ့အခါ loop တွေကိုဘယ်လိုသုံးမလဲဆိုတာကိုဖော်ပြချင်ပါတယ်။ သူတို့ဘယ်လိုသိကြလဲ၊ ကျနော်တို့လဲ သူတို့လိုသိအောင်ဘယ်လိုလုပ်ကြမလဲဆိုတာကို ကျနော်မြင်တဲ့ပုံစံနဲ့ပဲဖော်ပြထားတာဖြစ်ပါတယ်။ ဟုတ်ချင်မှလဲ ဟုတ်ပါလိမ့်မယ်။ ဒါပေမဲ့ ကျနော့်မှာ POC ရှိပါတယ်။ အောက်က Link မှာကြည့်ပါ

https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Remote%20commands%20execution

Command Execution မှာ wh^oa^mi ကအလုပ်လုပ်တယ်လို့ဆိုပါတယ်။ ဒါကိုသူများပြောလို့သိတာထက် ကိုယ့်ဘာကိုလဲသိအောင်လုပ်လို့ရပါတယ်။ PHP မှာဆိုရင် system() က Command ကို execute လုပ်ပေးနိုင်တယ်။ ဒါကြောင့် system("whoami"); က အလုပ်လုပ်တယ်ဆိုရင် ကျနော်တို့ကို www-data သို့မဟုတ် asus/blahblah စသည်ဖြင့် ပြန်ပြရမယ်။ ကျနော်တို့ Loop ပတ်လို့ရပါတယ်။

ဥပမာ

```
for ($i=0; $i < 255; $i++) {
        system("who".$chr($i)."ami");
        echo "<br>";
 }
```

ဒီလိုဆိုရင် အောက်ကလို ဘယ် character တွေထည့်လို့ရတယ်ဆိုတာ သိသွားတာပေါ့

who%0aami

who"ami

who^ami

တခြား ဥပမာအနေနဲ့ File System Attacks ကိုကြည့်စေချင်ပါတယ်။

http://www.ush.it/2009/02/08/php-filesystem-attack-vectors/

Non-Alphanumeric with loops

JS, PHP တို့ကို non-alphanumeric တွေဖန်တီးကြတာတွေ့ဖူးမှာပေါ့။ ဥပမာလေးတစ်ခုပြချင်ပါတယ်

```
php -r "for ($i=0; $i < 255; $i++) { print chr($i);} > alpha_1.txt
```

ASCII character တွေကိုရမှာဖြစ်ပါတယ်။ ဒီ character တွေကို Operators တွေနဲ့တွဲကြည့်မယ်။

http://php.net/manual/en/language.operators.php

~ ကိုသုံးတယ်ဆိုပါစို့

```
php -r "for ($i=0; $i < 255; $i++) { $test=chr($i);print ~$test;}" > alpha_2.txt
```

A မပါပဲနဲ့ A ကိုလိုချင်တယ်ဆိုရင် ဘယ်လိုလုပ်မလဲ?

```
php -r "print ~'¾';"
```

ဒီနည်းနဲ့ ~'¾'; ဟာ =A ဖြစ်နေတာကိုတွေ့ရပါမယ်။ ဒါပေမဲ့လိုချင်တာက A ပဲ၊ =A မဟုတ်ဘူး။့

```
php -r "print ~'¾'{1};"
```

သေသေချာချာလေ့လာချင်ရင်တော့ အောက်ကလို Blogpost တွေကနေလေ့လာနိုင်ပါတယ်

http://www.thespanner.co.uk/2011/09/22/non-alphanumeric-code-in-php/

Conclusion

ဒီဆောင်းပါးဟာ ကျနော့်ရဲ့ ဒုတိယဆောင်းပါးဖြစ်ပါတယ်။ အမှားတွေရှိနိုင်သလို အကောင်းဆုံး တင်ဆက်မှုတော့ဟုတ်ချင်မှဟုတ်ပါလိမ့်မယ်။ ပိုကောင်းမယ့်အရေးအသား ဖြစ်ဖြစ် ၊ အသုံးအနှုံးပြင်လိုတာပဲဖြစ်ဖြစ် တစ်စုံတစ်ရာရှိခဲ့လျှင် ကျနော့်ကိုအသိပေးစေလိုပါတယ်

Thanks

21 / 12 /2018