(lec 18) MALICIOUS SOFTWARE & BOTNETS

Botnet

A botnet is a no. of internet connected devices, each of which is running one or more bots. Botnets can be used to perform DDOS attack, steal data, send spam & allow attacker to access the device & its connection.

"Basic idea is ke logon ke sys attacks/infect krein & then take over the sys.

Botnet Threats

A network of compromised machines (bots) controlled by a botmaster. Responsible for:

① Launching DDOS attack ② SPAM sending ③ Click-fraud campaign ④ Info Theft

⑤ Large scale network probing activities (ie. scanning)

→ Shift from a fun-activity to a profit oriented business (attackers earn a lot by this & are getting better).

→ Torpig Botnet

Some researchers took over Torpig Bot. They learned that it was:

→ It was a Trojan horse

→ Distributed via Mebroot 'malware platform' as DLL

↳ malwares that facilitate other malwares in hiding & keep their existence undetected.

→ It used to inject itself at 29 different places in an app.

→ Used to steal sensitive info (credit card no, pwd) → HTTP injection for phishing

(jo bhi http pg. load hou rahay usmein extra form include krsaktay they)

→ uses 'encrypted' HTTP as 'command & control (C&C)' protocol — (to communicate with BE)

→ Mebroot

→ Sophisticated master boot record based Rootkit → Spreads via drive-by-downloads (unintended downloads)

PER 3/2.

> Data Collection Principles

- Principle 1 : Hijacked botnet should be operated so that any harm and/or damage to victims and targets of attacks would be minimized.
  - Never send new/blank config file
  - Respond with ok msg.
- Principle 2: The sinkholed botnet (not managed by good people) should collect enough info to enable notification & remediation of affected parties.
  - Worked with law enforcement (FBI etc)
  - Worked with bank security)

After data collection, observations are:

→ weak pwd → credential reuse → attacks enable.

> Lessons learnt

① Most security threats stem from web

② A malicious web pg leverages a defect in a prog to gain arbitary code execution

③ They exploits downloads & installs a malware sample that infects the victim.

Malicious Software

Malware refers to any unwanted software & executable code that is used to perform an unauthorized, often harmful, action on a computing device. It includes all such harmful softwares:

Means of distribution	Requires host	Run independently
self spreading	Virus	Worm
Non-spreading	Rootkit Trojan horse	Dialer Spyware Keylogger

Dependency on host

Host → existing software routine to compromise & run
↓ works as a server that opens into resource, services to other users or host on network

Virus — self replicating, needs a host to infect

Worm — self replicating, spreads autonomously over network

Trojan horse — malicious prog disguised as a legitimate software

Rootkit — used to keep access to a compromised sys
usually hides file, process & network connections

> Fighting Malware

Goal: → Understand malware behavior → Identify + classify malware → automatically generate effective malware detection models

→ Collect malware samples → Analyze malware samples

Static Analysis
- studying prog's properties without executing it
- reverse engineering may be hindered (eg. encryption)

Dynamic Analysis
- Studying a prog's properties by executing it
- Environment limited analysis

→ Extract (and generalize) malicious behavior → • Host • Network

→ Generate & deploy detection models

→ Hard Problems
→ lack of general definition of malicious behavior → Victims often (unwittingly) help attackers
→ Attackers have much freedom

Botnets

Bot → Autonomous progs performing tasks → More recent trend in malicious dev.

→ First bots were progs used for Internet Relay Chat (IRC) → offers useful services.

> Creation

> Botnets are created by infection + spreading

> Command & control channel (C&C), robustness features (eg. fast, flux, P2P)

→ Host infected by one of:
→ Network worm (vulnerabilities)
→ Email attachments
→ Trojan version of progs
→ Drive by download

→ Specialized services
→ PPI, Exploit-as-a-Service

> Drive by Download

Malicious Scripts → • Injected into legitimate sites (eg. via SQL injection)
• Embedded into ads
• Hosted on malicious sites (URL distributed via spam)

Drive by Download → • Attacks against web browser/vulnerable plug ins
• Typically launched via client-side script (eg. JS)

Re-direction → • Landing pg redirects to malicious sites (via iframe)
(If browser is compromised)
• Makes management easier
• customize exploits (browser version), serve each IP only once.

FEB 3/12.

Once a pc is compromised, There are high chances that it is a part of a zombie network that and is being controlled with C&C or a botnet and has a communication with Pc.

Command & Control (C&C)

→ basic idea of how C&C manage Things

① → We mean a **Centralized Control** (a centralized place) jiske Through jo bots hain wo hundreds & millions of infections to rakhay control krta hai.

- Most bots prefer ke HTTP ke Through info exchange krein - (filtering se bach jata hai)
- HTTP — commands published in web pgs.
- IRC — commands published in IRC channels.

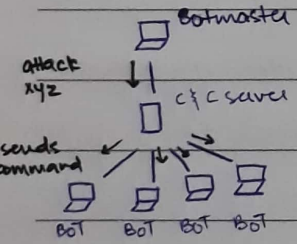
② can be **Distributed Control** (can add redundancy)

- P2P — commands and/or 'commander' addressed published in P2P network (more resilient to detection)

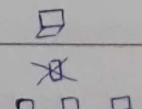
③ Have **Push & Pull** mechanism

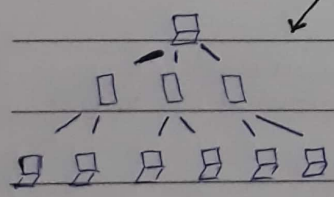
- Push — The bot silently waits for command from the 'commander' (wait for instructions)
- Pull — The bot repeatedly queries the 'commander' to see if there is new work to do (seek for instruction)

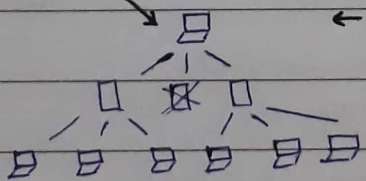
- **IRC based botnet**



- Bots continuously (re)connect to C&C server
- Botmaster sends a command to C&C server
- C&C server forwards the command to all connected bots
- If C&C server is isolated, botmaster loses control of all bots
- It could be very difficult to identify bot master (good hiding)
- Multiple C&C servers can be used simultaneously
- When C&C server is so isolated, the bots automatically connects to others







← • Use of multiple C&C servers increases the lifetime of botnet

- **HTTP based botnets**
  - → Very similar to botnet based on IRC
  - → C & C trace is difficult to identify (b/c packets are encrypted)
  - → Difficult to block at network level (eg. firewall)
  - → Difficult to block at DNS level (eg. domain blacklisting)

>Dynamic rendezvous points

- A meeting at a pre-arranged time & place.
- How does a bot locate C&C server
  - Hardcoded IP address → Bono ko ek IP dedein ke iske through communicate krna hai. (Bot & C&C)
    Prob → after a bit of research by people, IP can be identified
    Even if IP isn't in legal registration, it can easily be filtered out.
    But they use Fastflux & domainflux which makes them resilient
  - Fastflux → Hardcoded FQDN or dynamically generated FQDN (1 FQDN → 1 or more IP addresses)
  - Dynamicflux → Hardcoded URL or dynamically generated URLs (generated different domains as time passes by)

^both fluxes make bots intelligent that they don't communicate & connect using same domain.

↓

They create new ~~algo~~ domains & URLs using algo. So if you're analysing traffic, tou traffic aik jagah jatay way nai dikhe gi. So it ~~becomes hard to detect~~.

↓

Algo runs at both bots & C&C. (Domain = URL)

Domainflux

→ Botmaster ne jo bhi algo use kiya hotta hai ans us se domains generate hottay hein, it tries to get them registered jitney possible

After registration, it sets C&C in such a way that they map on C&C acc to days, months & weeks.

→ can be broken if someone decodes the algo & see if there are any unregistered domains & sets its own server there (b/c not all domains get registered easily)

FastFlux

→ Ak fully qualified domain name register krta hai, and peechay se IP addresses change krta hai.

→ IP addresses change: If you're ~~authoritative domain~~ server, so you can assign different IP at different time

IP is never hardcode (same nai rehtay)

* → (Problematic agents are replaced with others; botnet is composed of millions of agents; identity of code components of infras is well protected

- Search keys in P2P network → multiple domains are used by same bo

- Preventing the rendezvous
  - Network level ACLs
  - DNS ACL
  - HTTP ACL

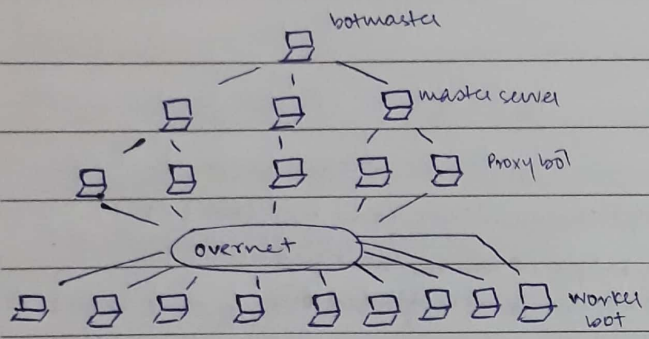
  → Different mechanisms can be applied to block such networks.
  - Once they're sure ke xyz domain ya URL block nai, they tend to block those list
  - They tend to improvise and make themselves resilient

- It is possible to perform clustering to determine that a domain is generated by a Domainflux

REB 3/2.

• P2P based botnet (Storm)

→ Added layer of security 'Overnet' which makes bot more distributed & makes them v. hard to compromise. as compared to traditional centralized or distributed botnet.



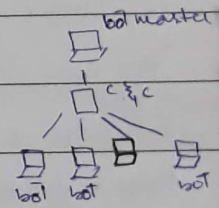
- Search for keys in P2P network to locate proxies
- connect to proxy & wait for command (each key is associated with IP and port of a new proxy)
- Worker botnet connects to proxy & authenticates itself & waits for command
- Proxy forwards command from master to workers & vice versa.
- Master servers are controlled directed by botmaster & are hosted on bullet-proof hosts.
- Workers with best resources are elected to proxies.

>Infiltrate a botnet (HTTP C&C)

→ The infiltrate connects to C&C server as normal bot but can see all bots connected

→ One way:

When you know ~~your~~ your sys ~~is compromised by a bot, ap analyze kr ke usse kiya traffic~~ hai aur kiya activity perform kr rahi hai etc. And when attack comes, you can see ke kiya command aayi hai aur sys se kaisa traffic jaa raha hai



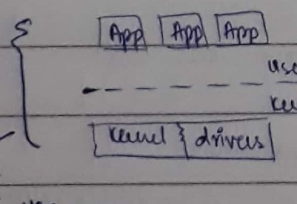
Prob → you'll be able to learn limited info. Baqi bots ke baray mein nai pata hoga.

But if server side infiltrate instead, can see other bots & send them commands (you're in total control)

Rootkits (type of malware)

A rootkit is a tool used to hide the presence of a malware on system from system administrator.

What to hide? → Files, registry keys, services, Network connections, Processes etc.



User space vs kernel space

Hijacking

There are different ways in which rootkit puts hooks & execution to hijack krein

Hooking → Hijack the flow of execution by modifying a code pointer.

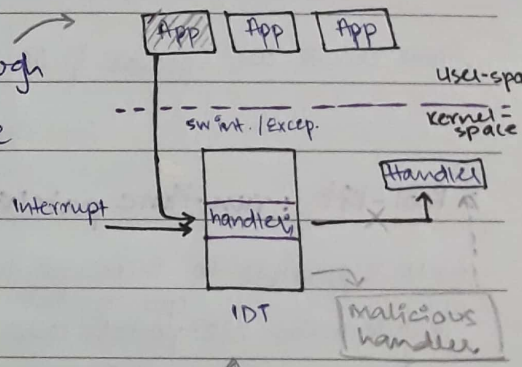
Eg. User space – IAT ; Kernel space – IDT, MSR, SSDT

↓

IAT : Import Address Table Equivalent to process linkage table and tells which external function this process is going to use.

> IDT Hooking (Interrupt Descripton Table)

Koi bhi app hoti hai & usmein interrupt ata hai so it moves through IDT ~~handler~~ (which is in kernel space & doing execution) & usmein se it further sent to handler & then delt with.



→ Interrupts & exception dispatching
→ Hijacking of IA handlers

• How can Rootkit sabotage it?

→ They can create malicious entries — app jaisay hi interrupt execute kre ga, it will call malicious handler rather than desired original handler.

→ Tou jo bhi handler call kiya gaya hai wo execute tou hoga par jo flow hai wo hijack hojaye ga.

• Execution hidden hi rahi.

• Problems

→ not possible to do filtering → Sys calls cannot be intercepted if sysenter/syscall used

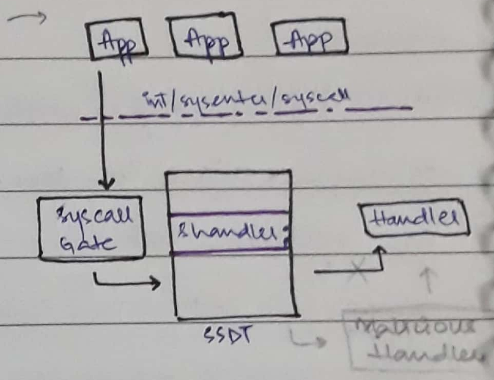
→ Easy to detect → (jab handler call kre tou check laga dein ke wo us service ka handler hai bhi ya nahi)

> SSDT Hooking (Service System Descripton Table)

→ Sys calls dispatching

→ Hijacking of SSDT descriptors

→ User jab bhi koi operation perform kre ga, so it'll go to syscall & SSDT mein dekhe ga ke OS ki konsi routine handle kr rahi & wahan jaa ke op. execute hojaye ga.



→ Here too, rootkits can enter a malicious handler & you're redirected to it & then handler redirect to the actual op. being executed
↓
flow is hijacked this way & remains undetected.

• Problems
→ Easily detected; put a check on routine & check if its a known address or not.

Some events are special & OS does keep a check on it eg. Data deletion.

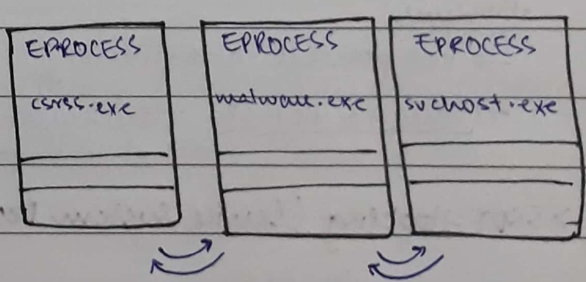
> Root-kit : run-time patching

→ It is possible to intercept the execution in multiple points.
• code mein aik (piece of code) 'patch' laga dein which will redirect the follow to malicious handler & then after malicious handler is done doing the activity, it'll go back to program ~~and real command will be detected~~

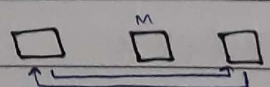
→ It is more difficult to detect (no common hooking point — code se hei ~~ds~~ redirect horahe) (whereas in previous, malicious handler was a different prog)
↓
more sophisticated, can't be detected using simple conditional checks.

> Root-kit : Direct Kernel Obj Manipulation (DKOM)

→ In memory alteration of a kernel structure.
→ No hook or patch necessary.
• Jaisay task managers shows all processes' list so malware.exe disappears from list of running processes



→ Scheduling is thread based
What rootkit does is ke linking ko free krdetay hain
so malware disappears from list



REB3/2.