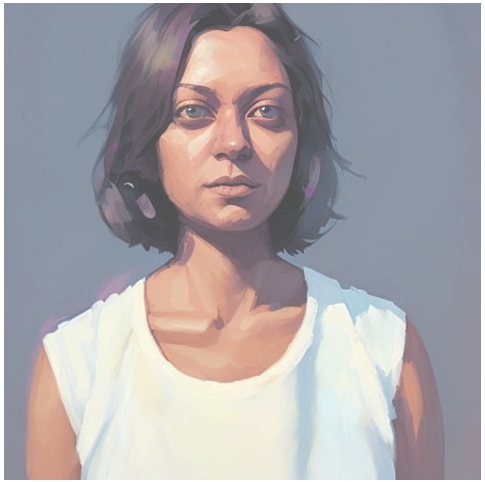
Tracking Bumblebee’s Development

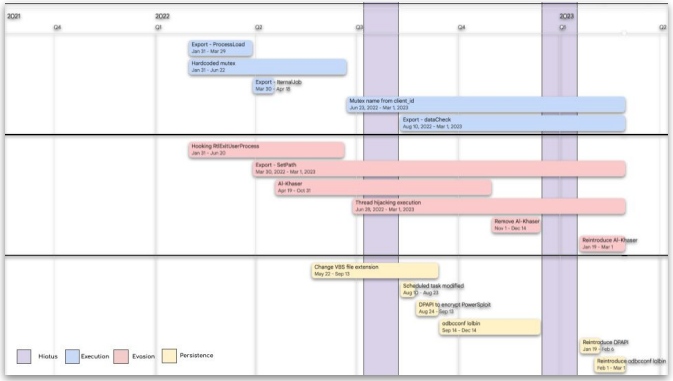


Suweera De Souza

Senior Security Researcher at
TAC eCrime CrowdStrike

Presentation Outline

- *Bumblebee*'s development timeline



- Endpoint Detection and Response (EDR) Evasion Techniques

Handles | DLLs | Threads

Stack | Module | Terminate

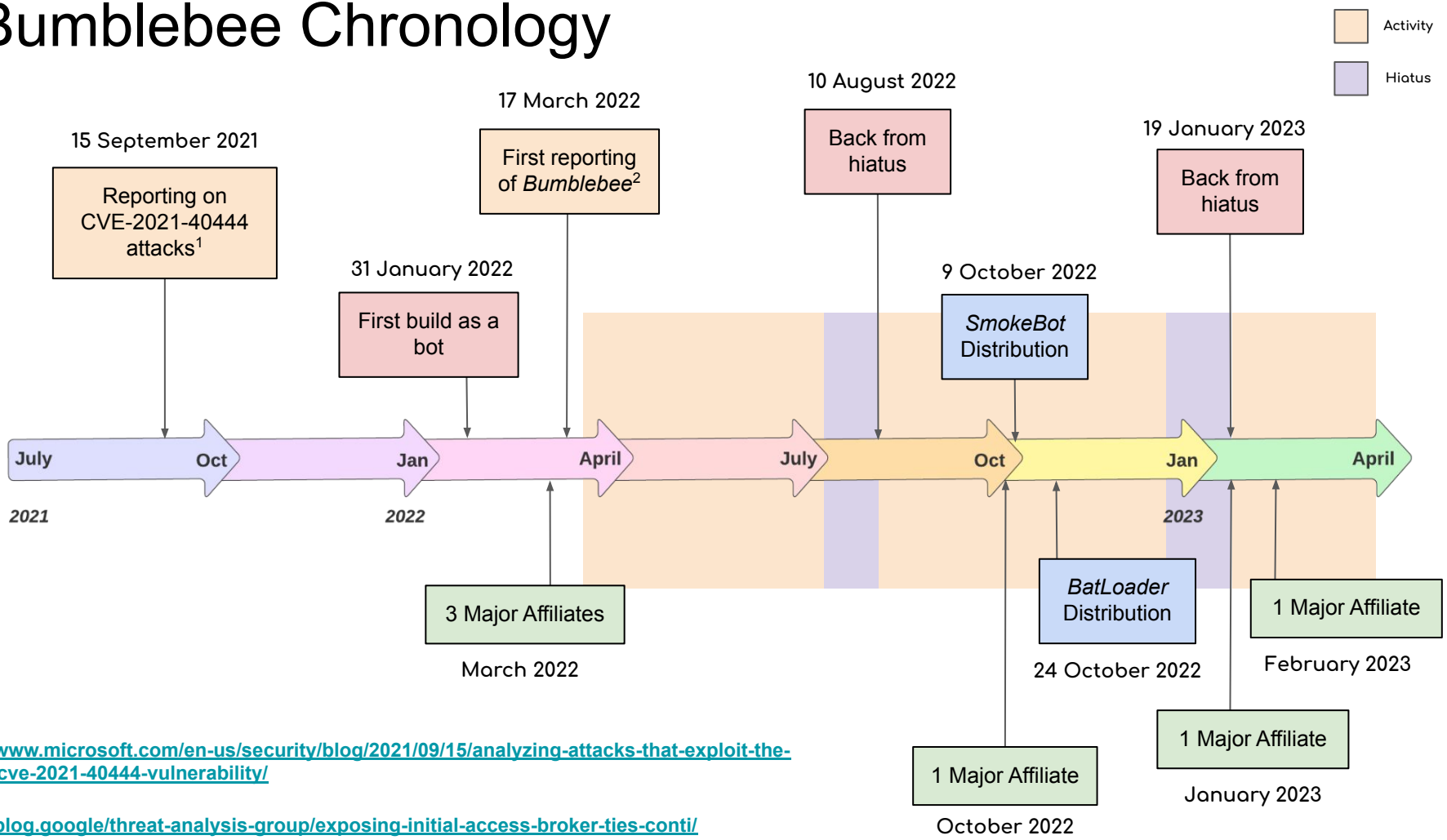
State	Wait Reason	TID	User Time	Kernel Time	CPU	CPU Time	Start Time	Start Address
Waiting	WrLpcReply	6584	00:00:00	00:00:00	0.78	00:00:00	02/10/23 13:04:...	ntdll.dll!RtlNewSecurityObjectWithMultipleInheritance
Waiting	UserRequest	3420	00:00:00	00:00:00		00:00:00	02/10/23 13:10:...	combase.dll!Ordinal122+0x9a0
Waiting	WrQueue	8944	00:00:00	00:00:00		00:00:00	02/10/23 13:14:...	ntdll.dll!LdrAccessResource+0x12c0
Waiting	UserRequest	60	00:00:00	00:00:00		00:00:00	02/10/23 13:04:...	dimsroam.dll!setPath+0xa1e94

- Assessment about the developers

```
0   ntoskrnl.exe+0x3fac26
1   ntoskrnl.exe+0x2e4540
2   ntoskrnl.exe+0x2e3a6f
3   ntoskrnl.exe+0x2e3313
4   ntoskrnl.exe+0x20cd8d
5   ntoskrnl.exe+0x2e6b09
6   ntoskrnl.exe+0x2e4867
7   ntoskrnl.exe+0x2e3a6f
8   ntoskrnl.exe+0x2c2c52
9   ntoskrnl.exe+0x6a156f
10  ntoskrnl.exe+0x4058b5
11  ntdll.dll!ZwDelayExecution+0x14
12  KERNELBASE.dll!SleepEx+0x9e
13  wabmig.exe+0x3e7f
14  KERNEL32.DLL!BaseThreadInitThunk+0x14
15  ntdll.dll!RtlUserThreadStart+0x21
```

- CrowdStrike Name: *Shindig*

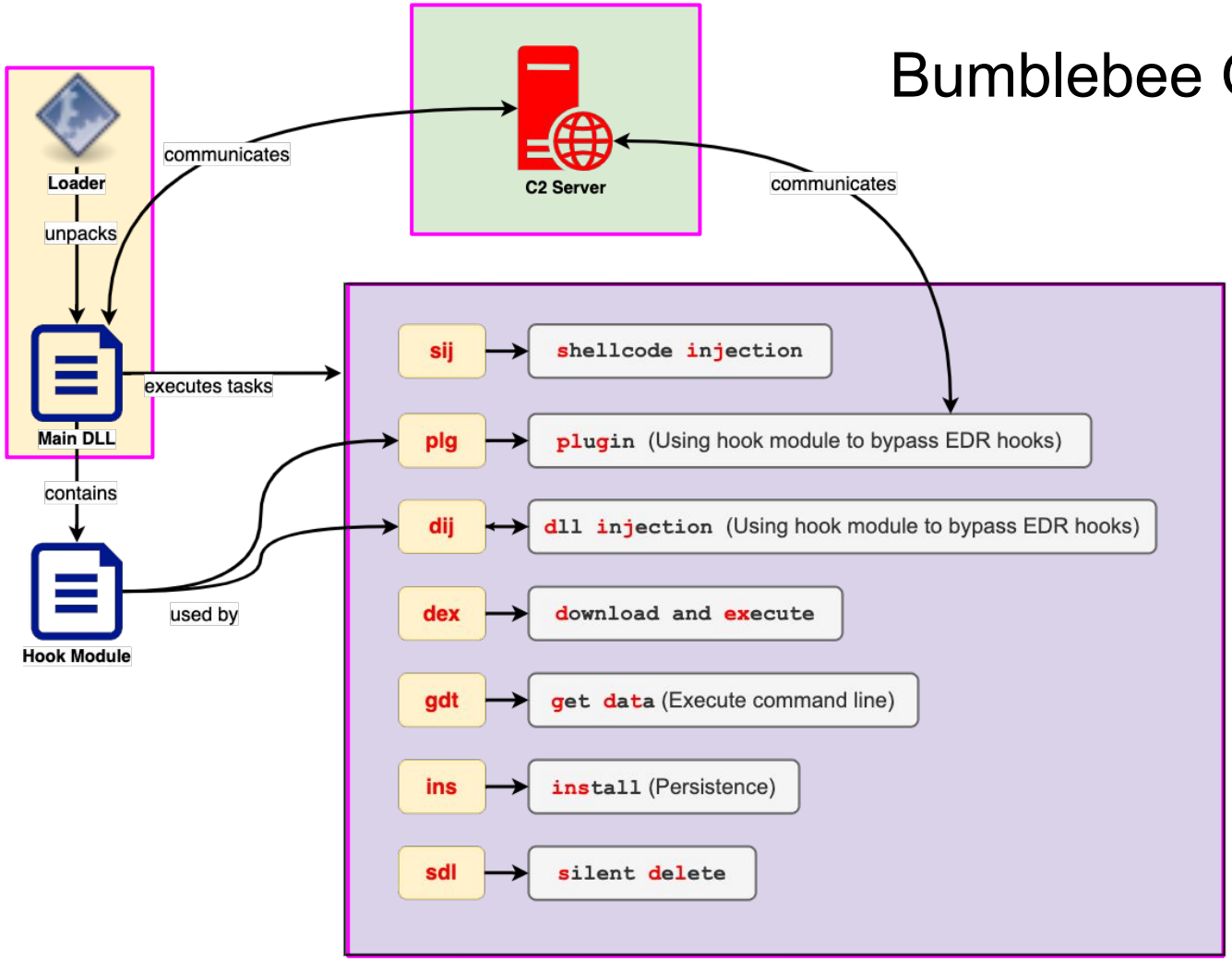
Bumblebee Chronology



[1] https://www.microsoft.com/en-us/security/blog/2021/09/15/analyzing-attacks-that-exploit-the-mshtml-cve-2021-40444-vulnerability/

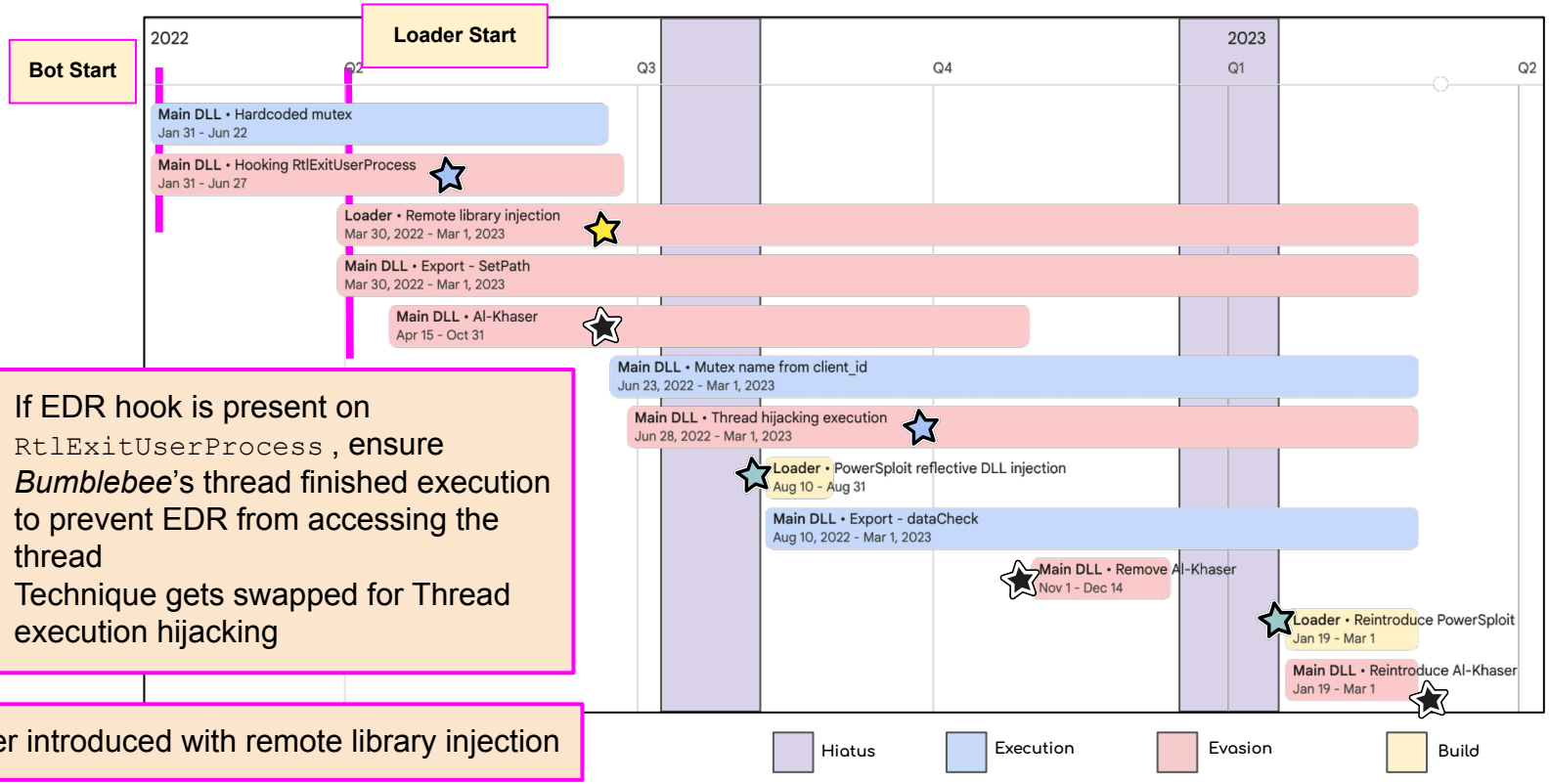
[2] https://blog.google/threat-analysis-group/exposing-initial-access-broker-ties-conti/

Bumblebee Overview



Loader / Main DLL Development

Note: *Bumblebee's* Main DLL and plugins use the Boost C++ library in many of its implementations.



- If EDR hook is present on `RtlExitUserProcess`, ensure *Bumblebee*'s thread finished execution to prevent EDR from accessing the thread
- Technique gets swapped for Thread execution hijacking

Loader introduced with remote library injection

Two loader variations of a loader are used

[1] https://github.com/LordNoteworthy/al-khaser

Remote Library Injection

- Masquerades *Bumblebee*'s main DLL as a legitimate DLL
- Hooks APIs used by NTDLL for mapping and loading DLLs
  - Emulates their operations against a memory region
- POC released in 2004[1]
- Observed used by *Ramnit*[2]

Threads

Terminate

son	TID	User Time	Kernel Time	CPU	CPU Time	Start Time	Start Address
ply	6584	00:00:00	00:00:00	0.78	00:00:00	02/10/23 13:04:...	ntdll.dll!RtlNewSecurityObjectWithMultipleInheritance
uest	3420	00:00:00	00:00:00		00:00:00	02/10/23 13:10:...	combase.dll!Ordinal122+0x9a0
e	8944	00:00:00	00:00:00		00:00:00	02/10/23 13:14:...	ntdll.dll!LdrAccessResource+0x12c0
uest	60	00:00:00	00:00:00		00:00:00	02/10/23 13:04:...	dimsroam.dll!setPath+0xa1e94

- Loader uses unique name **dimsroam.dll**
- setPath export function belongs to Bumblebee

[1] http://www.hick.org/code/skape/papers/remote-library-injection.pdf
[2] https://securityintelligence.com/posts/from-ramnit-to-bumblebee-via-neverquest/

Thread Execution Hijacking (Variation)

- Masquerades *Bumblebee* under a decoy start routine
  - Start routine is hidden under `RtlNewSecurityObjectWithMultipleInheritance`
- Swaps the start routine in the suspended thread's context
- Typically used for process injection[1]
- Observed used by COZY BEAR *C2-Client Dropbox Loader*[2]

Threads

Terminate

Reason	TID	User Time	Kernel Time	CPU	CPU Time	Start Time	Start Address
cReply	6584	00:00:00	00:00:00	0.78	00:00:00	02/10/23 13:04:...	ntdll.dll!RtlNewSecurityObjectWithMultipleInheritance
Request	3420	00:00:00	00:00:00		00:00:00	02/10/23 13:10:...	combase.dll!Ordinal122+0x9a0
eue	8944	00:00:00	00:00:00		00:00:00	02/10/23 13:14:...	ntdll.dll!LdrAccessResource+0x12c0
Request	60	00:00:00	00:00:00		00:00:00	02/10/23 13:04:...	dimsroam.dll!setPath+0xa1e94

Thread start address shows RtlNewSecurityObjectWithMultipleInheritance instead of Bumblebee's main DLL offset

[1] https://attack.mitre.org/techniques/T1055/003/
[2] https://github.com/Dump-GUY/Malware-analysis-and-Reverse-engineering/blob/main/APT29_C2-Client_Dropbox_Loader/APT29-DropboxLoader_analysis.md

C2 Communication

- WebSocket protocol
- Message is stored as JSON
- Message is RC4 encrypted
- The `hit` message is crafted from the Al-Khaser techniques

Request | **Response**

Ping

```
{
  "client_id": "9adddcc...",
  "client_ping": "FORTHEEMPEROR"
}
```

```
{
  "client_id": "9adddcc...",
  "session_id": "48wc6Lb..."
  "hit": True
}
```

Hit

```
{
  "client_id": "9adddcc...",
  "sess": "48wc6Lb...",
  "uuid": "AAAAAAAA-AAAA-AA...",
  "user": "EXAMPLE",
  "check_xen": false,
  "psexp_running": false,
  "wine_exports": false,
  "wine_req": false,
  "vbox_req_val": false,
  ...
  "binary_db": "U1FMaXRlIGZv…"
}
```

Beaconing-style communication - messages are sent in a loop

`binary_db` contains browsing history from Chrome and MSEdge

Task

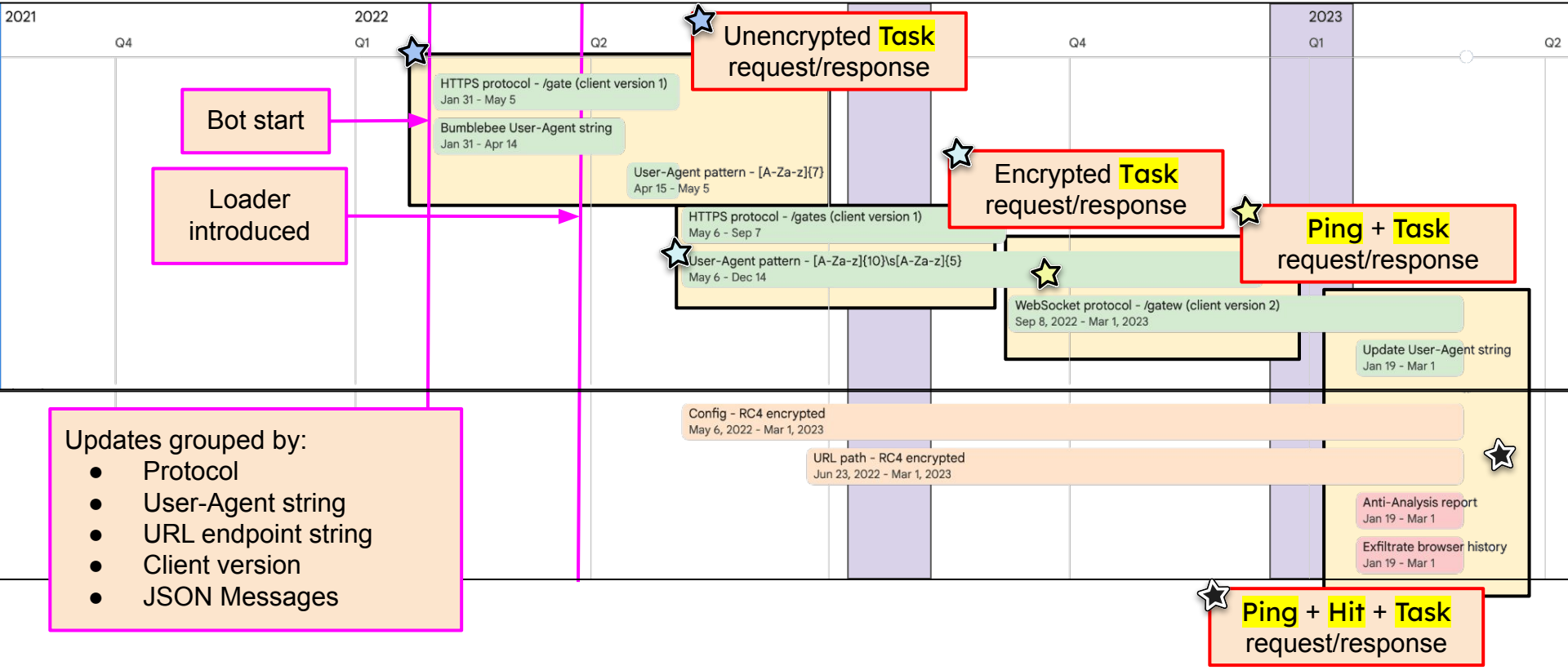
```
{
  "client_id": "9adddcc...",
  "group_name": "21maca",
  "sys_version": "Microsoft W...",
  "client_version": 2,
  "session_id": "48wc6Lb..."
}
```

```
{
  "response_status": 1,
  "tasks": [
    {
      "tasks": "dex",
      "task_id": null,
      "task_data": "TVqQAAMAAAAEA...",
      "file_entry_point": null
    }
  ]
}
```

Task Result Request

```
{
  "client_id": "9adddcc...",
  "group_name": "21maca",
  "sys_version": "Microsoft W...",
  "client_version": 2,
  "session_id": "48wc6Lb..."
  "tasks": [
    {
      "task_state": 1,
      "task_id": null,
      "task_result": ""
    }
  ]
}
```

C2 Communication Development



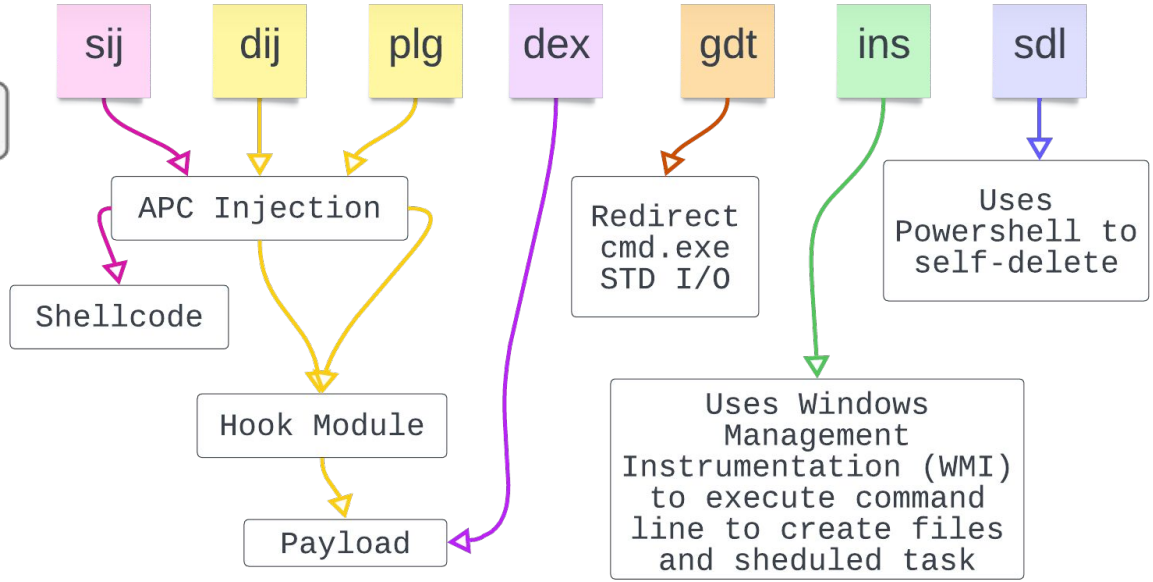
Updates grouped by:

- Protocol
- User-Agent string
- URL endpoint string
- Client version
- JSON Messages

```
Mozilla/5.0 (compatible, MSIE 11, Windows NT 6.3; Trident/7.0; rv
```

Tasks

- **sij** → shellcode injection
- **dij** → dll injection (Using hook module to bypass EDR hooks)
- **plg** → plugin (Using hook module to bypass EDR hooks)
- **dex** → download and execute
- **gdt** → get data (Execute command line)
- **ins** → install (Persistence)
- **sdl** → silent delete



Tasks Development

Loader introduced

Switch from using Windows API to WMI

Bot start

2021 Q4 | 2022 Q1 | Q2 | Q3 | Q4 | 2023 Q1 | Q2

- ins • Use API to create persistence — Jan 31 - May 5
- gdt • Introduced — May 6 - May 21
- ins • Use WMI to create persistence — May 6, 2022 - Mar 1, 2023
- gdt • Use Boost.Asio — May 22, 2022 - Mar 1, 2023
- ins • VBS file extension — May 22 - Sep 13
- ins • Scheduled task modified — Aug 10 - Aug 23
- ins • DPAPI to encrypt PowerSploit — Aug 24 - Sep 13
- ins • odbcconf lolbin — Sep 14 - Dec 14
- plg • Introduced with RPC — Sep 23, 2022 - Mar 1, 2023
- ins • Reintroduce DPAPI — Jan 19 - Feb 6
- sij • Rename injected process — Jan 28 - Mar 1
- dij • Rename injected process — Jan 28 - Mar 1
- plg • Rename injected process — Jan 28 - Mar 1
- dex • Rename executable — Jan 28 - Mar 1
- ins • Reintroduce odbcconf lolbin — Feb 1 - Mar 1

Hiatus

Persistence is different based on the loader used:

- *PowerSploit*
  - DPAPI for system encryption[1]
  - Schedule task executes a script that uses DPAPI to decrypt the PS file
- Packed DLL
  - Scheduled task uses LOLBin odbcconf to execute the DLL

Initially a simple implementation of gdt was being tested[2]

```
\Device\NamedPipe\boost_process_auto_pipe_9024
\Device\NamedPipe\boost_process_auto_pipe_9024
\Device\NamedPipe\boost_process_auto_pipe_9024
```

Remote Procedure Call (RPC) with named pipes is used to communicate with the plugins[3]

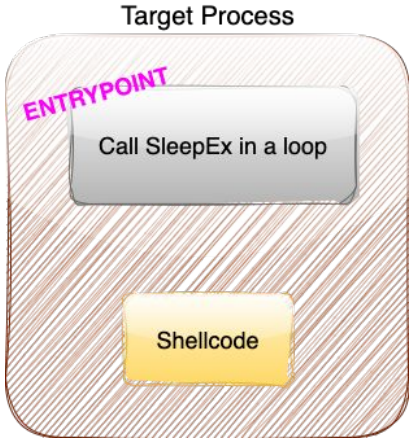
[1] https://learn.microsoft.com/en-us/dotnet/api/system.security.cryptography.protecteddata?view=windowsdesktop-6.0

[2] https://learn.microsoft.com/en-us/windows/win32/procthread/creating-a-child-process-with-redirected-input-and-output

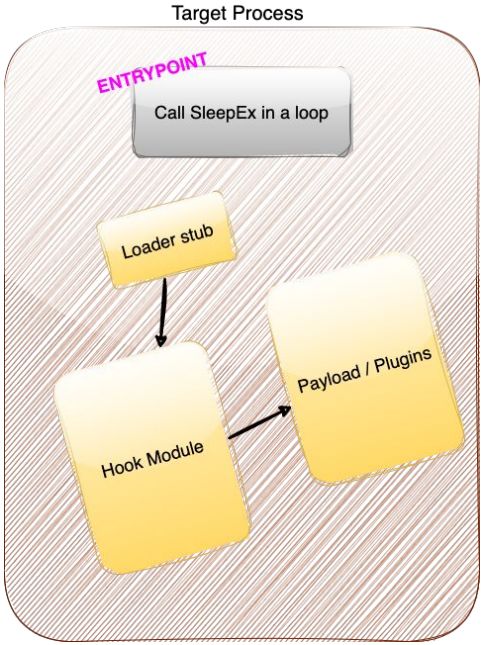
[3] https://learn.microsoft.com/en-us/windows/win32/rpc/asynchronous-rpc-over-the-named-pipe-protocol

Asynchronous Procedure Call (APC) Queue Code Injection (with WMI)

- Injected code is added to the target process thread's APC queue which gets executed when the thread enters an alterable state[1]
- *Bumblebee* injects its payloads into processes created with WMI



Shellcode Injection



DLL Injection

[1] https://attack.mitre.org/techniques/T1055/004/

Hook Module

- Removes EDR hooks on APIs
  - Compares the API's instructions in memory to that of the physical file
  - Compares instructions with a Length disassembler
  - Copy API's instructions from physical file to memory
- Uses Remote Library Injection to load the payload DLL as a legitimate DLL
- Implementation matches **libsplice**[1]
  - Commonly observed:
    - *Ramnit, TrickBot, BokBot*
    - *Game cheats*

[1] https://github.com/vol4ok/libsplice

Conclusion

- Mapping activity to the software development lifecycle
  - Agile methodology
    - First release is a minimal viable product (MVP) - **31 January 2022 - 31 March 2022**
    - Phase 2 introduced more EDR evasion - **31 March 2022** onwards
  - C2 infrastructure worked on during “hiatus”
- Stepping out of the norm
  - No API hashing or string obfuscation
    - Likely a result of using EDR evasion during execution
- Mature dev practices:
    - Boost
      - C2 communication
      - Command execution
    - libsplice - for Splicing

Thank You!