APPENDIX

.1 Explanation of Defined Labels in Table 2

$PS1$: If a process has network connection, we label the process with $PS1$, for we can't trust the data from network.

$PS2$: If a process has accessed high-value data flow nodes, we label the process with $PS2$, for the process holds sensitive data.

$PS3$: If a process has data from the network, we label the process with $PS3$, e.g., the process reads the file downloaded from network.

$PS4$: If a process has loaded or read the uploaded file from network, which can cause Webshell attack, we label the process with $PS4$ for the uploaded file can't be trusted.

$PS5$: If a process has interacted with non-existent files, which can cause Living-off-the-land attack, we label the process with $PS5$ for accessing to non-existent files is a feature of Living-off-the-land attack.

$PS6$: If a process has read a file that contains sensitive information such as $/etc/passwd$, we label the process with $PS6$ for it may cause sensitive information leakage.

$PS7$: If a process has read a file that saves historical commands such as $.bash\_history$, we label the process with $PS7$ for the traces may be exploited.

$PB1$: If a process has executed a file from network, we label the process with $PB1$, for the file can't be trusted.

$PB2$: If a process has executed a sensitive file, we label the process with $PB2$, for the file may cause potential attack.

$PB3$: If a process has executed a sensitive command, we label the process with $PB3$, for the command may cause potential attack.

$PB4$: If a process has executed any commands without being allowed, we label the process with $PB4$ (for Webshell only).

$PB5$: If a process has executed shell command, we label the process with $PB5$, for the process may reverse a shell to others.

$PB6$: If a process has modified the file that control scheduled tasks such as $/etc/crontab$, we label the process with $PB6$, for it may cause persistent control.

$PB7$: If a process has modified a file that control permissions such as $/etc/sudoers$, we label the process with $PB7$, for it may cause privilege escalation.

$PB8$: If a process has read high-value information, we label the process with $PB8$, for it may cause data exfiltration.

.2 Explanation of Defined Label in Table 3

$FU1$: If a file is uploaded, we label the file with $FU1$, for it can be untrustworthy.

$FU2$: If a file contains data from the network, we label the file with $FU2$, for the untrusted data may cause an attack later (code execution).

$FU3$: If a file does not exist, we label it with $FU3$. In a Living-off-the-land attack, accessing to a non-existent file is marked as access to a $(null)$ file.

$FU4$: If a file is written by the Webshell attack, we label the file with $FU4$, in order to capture the attack chain with webshell as the entry point.

$FU5$: If a file is written by the RAT attack, we label the file with $FU5$, in order to capture the attack chain with RAT as the entry point.

$FU6$: If a file is written by the Living-off-the-land attack, we label the file with $FU6$, in order to capture the attack chain with Living-off-the-land as the entry point.

$FH1$: If a file can control scheduled tasks such as $/etc/crontab$, we label the file with $FH1$.

$FH2$: If a file can control permissions such as $/etc/sudoers$, we label the file with $FH2$.

$FH3$: If a file holds sensitive information such as $/etc/passwd$, we label the file with $FH3$.

$FH4$: If a file saves historical commands such as $.bash\_history$, we label the file with $FH4$.

$FH5$: If a file is written by the process that have read sensitive information, we label the file with $FH5$, for it may cause data exfiltration.

.3 Event Used in APTSHIELD and Explanation

TABLE 12
a selective list of events used in APTSHIELD.

Events	Description
E0	File Read
E1	File Write
E2	Fork
E3	Execute
E4	LoadLibrary
E5	File Delete
E6	File Rename
E7	File Create
E8	File Property
E9	Exit
E10	LoadElf
E11	File Open
E12	File Close
E13	Vfork
E14	File Open with close-on-exec mark
E15	File Mmap

.4 Explanation of Defined Transfer Rules in Table 4

$PS1$-E1-$FU2$-D: If a process with label $PS1$ writes a file, we label the file with $FU2$.

$PS3$-E0-$FU2$-R: If a file with label $FU2$ is accessed by a process, we label the process with $PS3$.

$PS3$-E1-$FU2$-D: If a process with label $PS3$ writes a file, we label the file with $FU2$.

$PS4$-E0/E15-$FU1$-R: If a file with label $FU1$ is read or loaded by a process, we label the process with $PS4$.

$PS5$-E0/E10/E15-$FU3$-R: If a file with label $FU3$ is interacted by a process, we label the process with $PS5$.

$PB1$-E3-$FU2$-R: If a file with label $FU2$ is executed by a process, we label the process with $PB1$.

$PB1$-E10-$FU2$-R: If a file with label $FU2$ is loaded by a process, we label the process with $PB1$.

$PB1$-E1-$FU2$-D: If a process with label $PB1$ writes a file, we label the file with $FU2$.

$PB4$-E1-$FU4$-D: If a process with label $PB4$ writes a file, we label the file with $FU4$.

$PS4$-E0/E15-$FU4$-R: If a file with label $FU4$ is read or loaded by a process, we label the process with $PS4$.

$PB1$-E1-$FU5$-D: If a process with label $PB1$ writes a file, we label the file with $FU5$.

$PB1$-E0-$FU5$-R: If a file with label $FU5$ is accessed by a process, we label the process with $PB1$.

$PS5$-E1-$FU6$-D: If a process with label $PS5$ writes a file, we label the file with $FU6$.

$PS5$-E0-$FU6$-R: If a file with label $FU6$ is accessed by a process, we label the process with $PS5$.

$PB6$-E1-$FH1$-R: If a file with $FH1$ is written by a process, we label the process with $PB6$.

$PB7$-E1-$FH2$-R: If a file with $FH2$ is written by a process, we label the process with $PB7$.

$PS6$-E0-$FH3$-R: If a file with $FH3$ is accessed by a process, we label the process with $PS6$.

$PS7$-E0-$FH4$-R: If a file with $FH4$ is accessed by a process, we label the process with $PS7$.

$PB6-7|PS6-7$-E1-$FH5$-D: If a process with any one of labels $PS1\ 4$ writes a file, we label the file with $FH5$.

$PB8$-E0-$FH5$-R: If a file with label $FH5$ is accessed by a process, we label the process with $PB8$.

.5 Explanation of Defined Judge Rules in Table 5

Download&Execution: If a process has network connections and downloads a file from the network, executing the file indicates that a threat of Download&Execution may have occurred. The label $PB1$ represents that a process has the semantics of Download&Execution.

Webshell: If a process reads/loads an uploaded file and executes any commands without being allowed. It indicates that a Webshell attack may have occurred. Labels $PS4$&$PB4$ represent that a process has the semantics of the Webshell attack.

RAT: If a process executes a file downloaded from the network and then executes a shell command which reverses a shell to others. It indicates that a RAT attack may have occurred. Labels $PB1$&$PB5$ represent that a process has the semantics of the RAT attack.

Living-off-the-land: If a process interacts with a file that does not exist, and then executes a shell command to reverse the shell to others. It indicates that a Living-off-the-land attack may have occurred. Labels $PS5$&$PB5$ represent that a process has the semantics of the Living-off-the-land attack.

Suspicious Behavior: If a process reads/writes a file that controls permissions or holds sensitive information, it indicates that suspicious Behaviors (i.e., persistent stronghold, privilege escalation, credential access, information collection, etc.) may have occurred. The label $PB6|PB7|PS6|PS7$ represents that a process has the semantics of suspicious Behaviors.

Data Exfiltraion: If a process causes suspicious behavior and sends data out, it may cause the threat of data leakage. The label $PB8$ represents that a process has the semantics of data exfiltraion.

APT: If it meets any one of Webshell attacks, RAT attacks, and Living-off-the-land attacks. At the same time, there is data exfiltraion. It indicates that there is an APT attack. Labels $((PS4\,\&\,PB4)|(PB1\,\&\,PB5)|(PS5\,\&\,PB5))\,\&\,PB8$ represent that a process has the semantics of APTs.

.6 Details of Attacks from Laboratory

L-1: Webshell attack from the backdoored Apache. In this attack, port 80 is opened on the host to provide web services, and the website has file upload vulnerabilities while any file can be uploaded and accessed. The attacker successfully obtains the Webshell by uploading the Trojan horse file and accessing and connecting through Ant Sword. But the Webshell only has the www-data user permission. In the subsequent penetration, it is found that in the /etc/crontab file, the administrator uses the script $cleanup.sh$ to clean up the folder where uploaded files are stored once an hour. By viewing the script $cleanup.sh$, the attacker finds that anyone has the permission to modify it, so the attacker adds an attack command in $cleanup.sh$ to get the reverse shell. Then the attacker starts to monitor and wait for the script to be executed by the crontab. Soon the attacker obtains the permission of the administrator. By using id command, the attacker finds that the administrator belongs to the root group and has all permissions required by the attacker.

Through the shell of the administrator, the attacker can modify /etc/crontab and /etc/sudoers to keep persistent and exploit benign users. Also, the attacker can check /etc/passwd and .bash_history to obtain other user's credentials and collect their information. After that, the attacker writes the valuable information into a file and finally sends it out.

L-2: Remote Access Trojan from the phishing website. In this attack, the attacker uses a phishing website to put the Trojan file into the target host. After the monitor is turned on and the Trojan file is executed, the permissions of benign users are obtained. Afterwards, same as the follow-up operation in the Webshell attack scenario, the timing script in the crontab is found, and the script is modified to obtain privilege escalation. After that, the sensitive file is modified to achieve the purpose of persistence control and escalation. The sensitive file and credentials of user are read, written into a file, and finally sent out.

L-3: Living-off-the-land attack from vulnerable service. In this attack, the attacker finds that a special service is running on port 29273 on the host, and the permission of users could be obtained through interaction of string overflow. The subsequent attack scenario is similar to L-1 and L-2.

.7 Details of Attacks from Darpa Engagement

E-1: Information gather and exfiltration. In this attack, the attacker uses the username and password of a target user to log into the system and collect sensitive information, include reading $/etc/shadow$, $/etc/passwd$ files, executing commands such as $ifconfig$, $tcpdump$, $psaux$, $groups$, $dirname$, $etc.$, and returning the results to display on $/dev/pts/1$.

E-1: Malicious file download and execute. In this attack, the $scp-t./ccleaner$ command is used to download a file named $ccleaner$ from an unknown address to connect to the external IP 64.95.25.213, and then modify $SrcSinkObject$. The file $ccleaner$ is executed to obtain permissions and collect sensitive information from the system. The process also forks a $dbusdaemon$ process, which reads files such as $/proc/filesystem$, $/proc/mount$, and $/etc/passwd$. After

that, the $ccleaner$ and the files generated by $ccleaner$ are moved laterally to another host in the intranet through scp.

Next, a file named hc is copied from the external IP $108.25.125.188$ to the target host, and then the file is executed. During this execution, it loads a large number of binary and library files, and reads a large number of $/proc/directory$ files including $/proc/net/directory$ files, $/etc/passwd$, and other sensitive files. After that, the file is connected to the external IP, and a file named $/tmp/ext96481$ is generated. The file is subsequently executed and its privilege escalated, the malicious process also reads the $/etc/passwd$ file.

E-2: In-memory attack with firefox. In this attack, there is a writable and executable memory space in the $firefox$ process. The process also modifies the malicious device file $/dev/glx\_alsa\_675$ and increases its permissions. Next, when the $firefox$ process communicates with the external IP $86.129.31.201$, a malicious library file named $/tmp/libnet.so$ is dropped. Then the $sshd$ process loads the malicious file to connect with the same IP $86.129.31.201$. Subsequently, the malicious $sshd$ process extracts multiple files from the target host and transfers these files to the attacker's host. Finally, the process creates an executable file named $/home/admin/files/docs/audiobackup$ and executes it.

.8 Data Storage

The data compaction method proposed in this paper serves for real-time attack detection. When used in the real-world scenario, the enterprise will transmit compacted data (or original data) to another analysis or storage server for further investigation. To this end, combined with the existing work [63, 64], we discuss a storage-based data compaction method to provide guidance for security analysts from the following two perspectives. First, **content-based storage compaction** [63]. (1) Repeat keywords. Encode and Simplify repeated items such as uuid, time, suject, object, and event type. (2) High frequency words/strings. For example, when a web service such as tomcat7 is deployed on a Linux server, all web files that need to be accessed will be placed in the directory /var/lib/tomcat7/webapps by default, the redundancy path can be encoded and compacted. (3) Timestamps. Timestamps generated by the system are usually long and chronologically arranged, storage can be reduced by retaining the initial timestamp and using an offset to represent subsequent timestamps. Second, **deep learning-based compaction** [64]. One can also use a trained encoder and arithmetic encoding to compact a data file. Note that redundant semantics skipping, non-viable entity pruning, content-based storage compaction, and deep learning-based compaction can be used simultaneously to minimize storage overhead without affecting the accuracy of forensic analysis.