Here is the export package containing the reconfigured maritime exclusion filter script, automated BGP route-map injection payload, and Cisco IOS-XE parameters formatted for deployment into your local pipeline:

```
#!/usr/bin/env python3
"""
Sphinx Surveillance - Direct Pipeline Export Package
Module: Maritime Exclusion & Tactical Edge Remediation
(SX-NATO-WW3-2026)
"""

import json
import logging
import datetime
import paramiko

logging.basicConfig(level=logging.INFO, format='%(asctime)s
[PIPELINE-DEPLOY] %(levelname)s %(message)s')

# ==========================================
# 1. BAYESIAN MARITIME THREAT ENGINE EXPORT
# ==========================================

class BayesianMaritimeThreatEngine:
    def __init__(self, stix_bundle_json):
        self.load_initial_bundle(stix_bundle_json)

    def load_initial_bundle(self, bundle_json):
        bundle = json.loads(bundle_json)
        self.hypotheses = {}
        for hyp in bundle.get("hypotheses_evaluated", []):
            self.hypotheses[hyp["id"]] = {
                "designation": hyp["designation"],
                "prior": hyp["baseline_prior_probability"],
                "current_probability":
hyp["baseline_prior_probability"]
            }
        logging.info("Maritime baseline prior probabilities loaded
into pipeline.")

    def update_probabilities(self, stix_indicator_json):
        indicator_data = json.loads(stix_indicator_json)
        indicator_id = indicator_data.get("id")
        likelihoods =
indicator_data.get("x_sphinx_conditional_likelihoods", {})
        if not likelihoods:
            logging.warning(f"Indicator {indicator_id} missing
conditional likelihood data.")
            return
```

```
        marginal_probability_evidence = 0.0
        for hyp_id, hyp_data in self.hypotheses.items():
            conditional_likelihood = likelihoods.get(hyp_id, 0.5)
            marginal_probability_evidence += conditional_likelihood *
hyp_data["current_probability"]

        if marginal_probability_evidence == 0:
            logging.error("Denominator collision: Marginal probability
equals zero.")
            return

        for hyp_id, hyp_data in self.hypotheses.items():
            conditional_likelihood = likelihoods.get(hyp_id, 0.5)
            updated_posterior = (conditional_likelihood *
hyp_data["current_probability"]) / marginal_probability_evidence
            self.hypotheses[hyp_id]["current_probability"] =
round(updated_posterior, 4)

        self.render_dashboard_telemetry()

    def render_dashboard_telemetry(self):
        print(f"\n[===] MARITIME TELEMETRY SHIFT REGISTER:
{datetime.datetime.utcnow().isoformat()}Z [===]")
        for hyp_id, data in self.hypotheses.items():
            print(f" -> {data['designation']} ({hyp_id}): Convergence
Probability = {data['current_probability'] * 100:.2f}%")
        print("
[================================================]\n")


# ========================================
# 2. AUTOMATED BGP / NULL0 REMEDIATION EXPORT
# ========================================

def execute_maritime_edge_remediation(target_router_ip, ssh_username,
ssh_password, infected_prefix):
    ssh = paramiko.SSHClient()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    try:
        logging.info(f"Connecting to coastal edge router
{target_router_ip} via encrypted SSH...")
        ssh.connect(target_router_ip, username=ssh_username,
password=ssh_password, timeout=10)
        channel = ssh.invoke_shell()

        commands = [
            "configure terminal\n",
```

```
            f"ip route {infected_prefix} 255.255.255.0 Null0 10\n",
            "router bgp 65001\n",
            "neighbor 10.200.4.2 route-map BLOCK-MARITIME-HIJACK-IN
inbound\n",
            "end\n",
            "clear ip bgp soft inbound\n"
        ]

        for cmd in commands:
            channel.send(cmd)

        logging.info(f"Maritime remediation route-map successfully
injected for prefix {infected_prefix}.")
    except Exception as e:
        logging.error(f"Pipeline remediation error: {str(e)}")
    finally:
        ssh.close()


# ==========================================
# 3. PIPELINE VERIFICATION EXECUTION BLOCK
# ==========================================

if __name__ == "__main__":
    # Initialize Maritime Bayesian Matrix
    initial_maritime_matrix = {
        "type": "bundle",
        "id": "bundle--maritime-ex-99a",
        "hypotheses_evaluated": [
            {"id": "actor--shadow-fleet-breach", "designation":
"Shadow Fleet Exclusion Violation", "baseline_prior_probability":
0.55},
            {"id": "actor--naval-false-flag", "designation": "Staged
Maritime False-Flag Attack", "baseline_prior_probability": 0.45}
        ]
    }

    new_maritime_indicator = {
        "type": "indicator",
        "id": "indicator--usv-swarm-telemetry-01",
        "x_sphinx_conditional_likelihoods": {
            "actor--shadow-fleet-breach": 0.15,
            "actor--naval-false-flag": 0.85
        }
    }

    # Run Simulation
    engine =
```

```
BayesianMaritimeThreatEngine(json.dumps(initial_maritime_matrix))
    engine.update_probabilities(json.dumps(new_maritime_indicator))
```

Configuration File: Cisco IOS-XE Suite-B Maritime Profile (sec-rt-poti-maritime-01.cfg)

```
!
hostname SEC-RT-POTI-MARITIME-01
!
crypto ikev2 proposal MARITIME-SUITE-B-PROP
 encryption aes-gcm-256
 prf sha384
 group 20
!
crypto ipsec profile MARITIME-IPSEC-PROFILE
 set transform-set TS-AES-GCM
 set ikev2-profile HAIPE-EMULATION-PROFILE
!
interface Tunnel500
 description *** SECURE BLACK SEA USV & RADAR TELEMETRY VTI ***
 ip address 10.200.4.1 255.255.255.252
 tunnel source GigabitEthernet1
 tunnel mode ipsec ipv4
 tunnel destination 203.0.113.88
 tunnel protection ipsec profile MARITIME-IPSEC-PROFILE
!
ip prefix-list COASTAL-LOGISTICS-BOUNDS permit 10.200.0.0/16 le 24
!
route-map BLOCK-MARITIME-HIJACK-IN permit 10
 match ip address prefix-list COASTAL-LOGISTICS-BOUNDS
 set local-preference 150
route-map BLOCK-MARITIME-HIJACK-IN deny 20
!
```